ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 37

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

DETECTION AND ENFORCEMENT OF NON-LINEAR CORRELATIONS FOR FAIR AND ROBUST MACHINE LEARNING APPLICATIONS

Presentata da: Luca Giuliani

Coordinatore Dottorato

Ilaria Bartolini

Supervisore

Michele Lombardi

Esame finale anno 2025

Abstract

Detecting correlations is crucial in several Machine Learning tasks, such as the identification of patterns or the enforcement of certain relational constraints. In the realm of algorithmic fairness, correlations are particularly significant, as indicators typically quantify the degree of dependence between a sensitive input attribute and a target variable. Nonetheless, traditional measures have been focusing solely on categorical protected attributes due to technical limitations, thus neglecting continuous sensitive information like age, income, degree of disability, or other aggregated numerical variables. To overcome these limitations, recent research has suggested using the Hirschfeld–Gebelein–Rényi (HGR) correlation coefficient as a measure of fairness. HGR is an extension of Pearson's coefficient able to detect non-linear correlations by employing two mapping functions called copula transformations; in this dissertation, we present a novel computational approach for estimating it by means of user-defined kernel functions parameterized through a vector of mixing coefficients. Our approach is deterministic, offers increased robustness, improves interpretability compared to existing methods, and features other advantageous properties that make it more trustworthy for practical applications. We demonstrate its benefits over other computational techniques in both synthetic data and real-world benchmarks; then, following a minor variation of the HGR semantics, we introduce the Generalized Disparate Impact (GeDI) indicator, which broadens the legal notion of disparate impact to continuous input variables. Empirical findings confirm that this indicator can effectively reduce unfairness across three benchmark datasets, as well as in a practical use case involving long-term fairness in ranking systems; moreover, we show how both measures can be brought into a unified framework, and are equivalent up to a data-dependent scaling factor. To conclude, we discuss ongoing and future works regarding both methodological extensions of our Kernel-Based HGR method and potential applications in intersectional fairness and causal discovery. All our theoretical claims are supported by mathematical proofs and empirical evaluations, whose code can be accessed under the MIT License at the following public repository: `https://github.com/giuluck/non-linear-correlations`.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The influence of Artificial Intelligence (AI) on society has been widely acknowledged during the last decades. As data-driven technologies increasingly assume prominent roles in several social arenas, questions about their behavior began to be raised. The European Commission has already outlined principles for the development of ethical and human-centered applications [High-Level Expert Group on AI, 2019], which paved the way for the first legislative action on Artificial Intelligence, known as the AI Act [Madiega, 2021]. A major focus is placed on the trustworthiness of these systems, which encompasses not only safety requirements but also cross-cutting considerations such as adherence to anti-discrimination policies, interpretability of the yielded results, and system robustness against failures and external attacks.

A recurring task in data-driven applications is correlation discovery. Machine Learning (ML) algorithms, along with their even more opaque Deep Learning and Generative AI counterparts, primarily focus on recognizing and replicating common patterns by detecting correlations in the input data. However, these correlations do not consistently yield expected outcomes due to potential confounding factors, as well as historical and sampling biases in the data selection process. Without the integration of specific constraints, the sole reliance on detected patterns can be particularly precarious, especially in domains subject to legal and social standards where learning models have been shown to absorb and perpetuate systemic discrimination against marginalized groups [Angwin et al., 2022].

From a mathematical perspective, analyzing fairness characteristics in Machine

Learning is a challenging task that requires dealing with statistical distributions and subsymbolic non-linear models [Srivastava et al., 2019]. Despite this complexity, various strategies have been developed to assess and ensure alignment with fairness principles, thereby preventing trained models from discriminating against minority groups or individuals [Mehrabi et al., 2021]. These methodologies rely primarily on quantitative indicators that assess the level of correlation between a "protected attribute" and the outcome of the task. Although these metrics are arguably the most practical means for addressing fairness concerns, their support is innately limited to discrimination defined over categories, since assessing correlations on continuous variables involves more significant computational challenges. In fact, the most recognized correlation indicators suitable for numerical data are confined to particular forms of correlation, such as linear with Pearson's coefficient or monotonic with Spearman's or Kendall's rank, and are therefore inadequate to provide guarantees when dealing with non-categorical protected attributes.

An interesting exception is the Hirschfeld–Gebelein–Rényi (HGR) correlation coefficient [Rényi, 1959], which extends Pearson's coefficient by incorporating non-linear effects via two mapping functions, referred to as copula transformations. Various computational methods have been proposed over time to estimate HGR, given its theoretically uncomputable nature, all of which require a careful balance between bias and variance in estimation models in order to achieve a reliable approximation; moreover, recent research [Mary et al., 2019, Grari et al., 2020] has examined ways to employ HGR as a measure of unfairness when the protected attribute is continuous. However, the proposed methods face several limitations that might limit their applicability in real-world scenarios, as they can suffer from sampling noise, lack deterministic definitions, and their interpretation is often challenging for human evaluators. Therefore, driven by the necessity to enhance the robustness of measuring and enforcing non-linear correlations, particularly in the fairness domain, we explored this issue and propose several solutions that rely on deterministic and interpretable algorithms.

The result of our research additionally led to the development of a Python package aimed at facilitating the practical application of non-linear correlation indicators, which contains the implementation of our proposed solutions along with

various existing methods. The code is available under the MIT license and can be accessed at: `https://github.com/giuluck/maxcorr`, or installed via any package manager for Python. The remainder of this dissertation is structured as follows:

Chapter 2 provides the essential background needed to grasp both the technical and the social aspects referenced throughout the text. Specifically, we begin with an overview of fundamental Machine Learning concepts and methods for integrating external constraints into learning processes. We then proceed to examine the key elements of correlation and the available indicators to measure it. Finally, we introduce the readers to the topic of algorithmic fairness, examining its societal roots along with the various types of metrics and algorithms employed to ensure it.

Chapter 3 focuses on the Hirschfeld–Gebelein–Rényi (HGR) indicator, an extension of Pearson's coefficient that can intercept non-linear dependencies. We initially review existing algorithmic approaches from the literature to estimate its value, showing both their strengths and limitations. Later, we present two novel computational techniques for estimating HGR. We highlight their properties and show how they can address the shortcomings of existing state-of-the-art methods, confirming such hypotheses through formal proofs and experiments on synthetic datasets. The theoretical and empirical findings outlined in this chapter have been derived and refined from a research paper that is currently under review at a renowned Artificial Intelligence journal.

Chapter 4 addresses the role of correlation measures in promoting fairness when dealing with continuous protected attributes. First, we demonstrate how our method for HGR estimation can function as a loss regularizer within a constrained machine learning setting, a contribution that is drawn from the abovementioned research paper currently under review. Next, after identifying certain semantic limitations, we propose a new fairness metric termed Generalized Disparate Impact (GeDI), which extends the legal concept of disparate impact to encompass continuous variables. We validate its effectiveness with two additional experimental studies on three benchmark datasets, adapting results from [Giuliani et al., 2023] with minor alterations in

order to encompass new insights gained after the paper submission, although the conclusions remain consistent with our initial publication. Moreover, we show a practical case study on long-term fairness enforcement for ranking tasks in the education domain, as discussed in [Giuliani et al., 2024]. We conclude by discussing the similarities and differences between HGR and GeDI, showing how a unified formulation can link the two indicators up to a data-dependent scaling factor.

Chapter 5 outlines a series of potential future research directions, all of which are currently being explored, despite the lack of any definitive conclusion. Initially, we initially focus on the technical challenge of selecting an appropriate family of kernel expansions, as a way to mitigate numerical instability and semantic incompatibilities. Next, we present a multi-variate extension of HGR, which could be applied in the field of fairness to ensure intersectionality or within the domain of causal discovery to enhance the reliability of conditional independence tests. Lastly, building upon the theoretical insights from our paper [Maggio et al., 2023], we propose a new approach to redefine fairness through a causal perspective, employing HGR to decorrelate each input feature from the sensitive attribute.

Chapter 6 reports some conclusive remarks, summarizing the findings of this dissertation and presenting our future goals on how this research might stimulate new investigations in the domain of trustworthy machine learning.

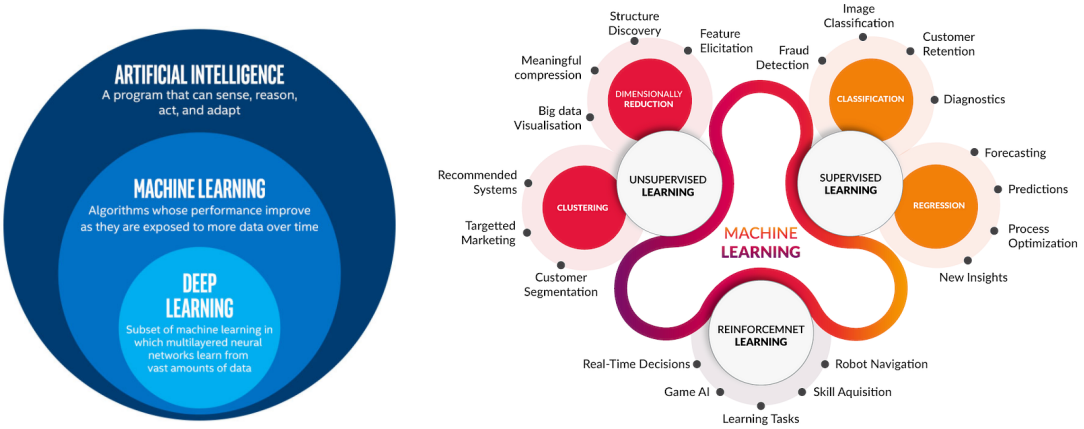
Chapter 2

Background

This chapter aims to ensure a foundational understanding of the key concepts that will be referenced throughout this dissertation. Specifically, in Section 2.1 we present the evolution of Machine Learning, from its statistical foundations to the latest hybrid methods that incorporate knowledge and constraints into data-driven learning. Section 2.2 discusses the theoretical principles of correlation along with the most important indicators used for its measurement. Lastly, we investigate the field of algorithmic fairness in Section 2.3, covering both its social and technical implications, up to the ongoing research challenges.

2.1 Machine Learning

Machine Learning is a branch of Artificial Intelligence that includes algorithms that are not explicitly programmed for a specific task, but rather fed with a set of data points from which patterns are extrapolated. Unlike conventional rule-based systems, which execute cognitive tasks using high-level procedures within a logical framework, machine learning algorithms are defined as subsymbolic since they employ statistical and optimization techniques to autonomously refine their internal state to model the distribution of the input data – see Figure 2.1(a). The recent growth of deep neural networks has stimulated a popularity rise of the field, which not only led to successful applications across various domains, but also opened new research directions aimed at addressing the inherent limitations of these models.



(a) Artificial Intelligence Taxonomy (b) Machine Learning Paradigms

Figure 2.1: Overview of the Artificial Intelligence and Machine Learning fields.
From:
(a) https://medium.com/swlh/artificial-intelligence-machine-learning-and-deep-learning-whats-the-real-difference-94fe7e528097
(b) https://resources.experfy.com/ai-ml/coding-deep-learning-for-beginners-types-of-machine-learning

Among all, we mention the interpretability of the results and the robustness against noisy data and intentional data pollution as two of the major concerns.

2.1.1 Learning Tasks & Algorithms

The most prevalent task in machine learning is Supervised Learning. Within this paradigm, we have access to a dataset $\mathcal{D} = \{X_i, y_i\}_{i=1}^n$, where X represents a matrix of input features and y is the vector of true output labels, also known as "ground truths". The goal is to train a machine learning model $\mathcal{M}$, parametrized over a vector of learnable coefficients θ, so to minimize a task-specific loss function $\mathcal{L}(y, \mathcal{M}(X;\theta))$. Once the training phase has finished, the obtained optimal parameters θ^* can be used to estimate the output $\hat{y}_{n+1}$ for a new data point X_{n+1}.

A common approach to solving this task involves likelihood maximization or iterative gradient-based techniques. There exist several supervised learning models, ranging from basic ones like Linear and Logistic Regression to more advanced ones such as Neural Networks. Another interesting category is that of ensemble models, such as Random Forests and Gradient Boosting, where the outputs of many independent smaller models are combined together. Throughout this dissertation, we will use some of them in our learning tasks, with appropriate modifications to

suit our specific use cases. We assume that the reader is already familiar with the fundamental concepts of such models; if not, we refer them to [James et al., 2023] or any other introductory text on statistical learning.

It is also worth noting that supervised learning is just one approach within the broader field of machine learning – see Figure 2.1(b). Other notable paradigms include Unsupervised Learning and Reinforcement Learning, where the ground truth vector y is not provided with the dataset. Specifically, unsupervised learning aims to organize the data into clusters of similar samples, whereas reinforcement learning updates the model parameters through a reward mechanism defined via the interaction between an agent and the environment. There exist some learning models which are specialized to these approaches; others, instead, cut across multiple paradigms. For instance, beyond their application in supervised learning, neural networks are prevalent both in deep reinforcement learning algorithms and in certain unsupervised learning tasks like anomaly detection, where autoencoder architectures are largely employed. Nonetheless, our research focuses exclusively on supervised learning, particularly within the framework of constrained supervised learning through loss regularizers for neural networks or projection-based declarative methods for gradient-free models. For this reason, we will not delve into these two or any other learning paradigms.

2.1.2 Supervised Learning as Optimization

From a statistical viewpoint, supervised learning algorithms are designed to learn the distribution of the process that generated the target data y, conditioned on the inputs X. This task is typically formulated as a maximum likelihood estimation

or, equivalently, as minimizing the log-likelihood. Namely, we want to solve:

$$\begin{aligned}
\arg\max_{\theta} P(y \mid X, \theta) &= \arg\min_{\theta} -\log P(y \mid X, \theta) \\
&= \arg\min_{\theta} -\log \left[\prod_{i=1}^{n} P(y_i \mid X_i, \theta) \right] \\
&= \arg\min_{\theta} -\sum_{i=1}^{n} \log \left[P(y_i \mid X_i, \theta) \right] \\
&= \arg\min_{\theta} -\sum_{i=1}^{n} \log \left[P(y_i \mid \mathcal{M}(X_i; \theta)) \right]
\end{aligned} \tag{2.1}$$

where the probability is eventually conditioned on the representations returned by the parametric learning model $\mathcal{M}(X; \theta)$.

When the target vector is categorical, specifically $y \in \{1, \ldots, k\}^n$, we label the problem as a classification task and employ the crossentropy loss as follows:

$$\begin{aligned}
\arg\max_{\theta} P(y \mid X, \theta) &= \arg\min_{\theta} -\sum_{i=1}^{n} \log \left[P(y_i \mid \mathcal{M}(X_i; \theta)) \right] \\
&= \arg\min_{\theta} -\sum_{i=1}^{n} \sum_{c=1}^{k} \log \left[P(y_i = c \mid \mathcal{M}(X_i; \theta)) \right] \\
&= \arg\min_{\theta} -\sum_{i=1}^{n} \sum_{c=1}^{k} \begin{cases} \log \left[\mathcal{M}(X_i; \theta) \right] & \text{if } y_i = c \\ 0 & \text{otherwise} \end{cases} \\
&= \arg\min_{\theta} -\sum_{i=1}^{n} \sum_{c=1}^{k} \mathbb{I}(y_i = c) \cdot \log \left[\mathcal{M}(X_i; \theta) \right]
\end{aligned} \tag{2.2}$$

where $\mathbb{I}(\pi)$ represents the indicator function, yielding 1 if the condition π is satisfied and 0 otherwise. In contrast, when $y \in \mathbb{R}^n$, we define it as a regression task and

obtain the following sum of squared error loss:

$$\begin{aligned}
\arg\max_{\theta} P(y \mid X, \theta) &= \arg\min_{\theta} -\sum_{i=1}^{n} \log\left[P(y_i \mid \mathcal{M}(X_i;\theta))\right] \\
&= \arg\min_{\theta} -\sum_{i=1}^{n} \log\left[\frac{1}{\sigma_{\mathcal{M}}\sqrt{2\pi}} e^{-\left(\frac{y_i-\mu_{\mathcal{M}}}{\sigma_{\mathcal{M}}}\right)^2}\right] \\
&= \arg\min_{\theta} -\sum_{i=1}^{n} \left[\log\left[\frac{1}{\sigma_{\mathcal{M}}\sqrt{2\pi}}\right] - \left(\frac{y_i-\mu_{\mathcal{M}}}{\sigma_{\mathcal{M}}}\right)^2\right] \\
&= \arg\min_{\theta} -\sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi}}\right] + \sum_{i=1}^{n} \left(y_i-\mu_{\mathcal{M}}\right)^2 \\
&= \arg\min_{\theta} \sum_{i=1}^{n} \left(y_i - \mathcal{M}(X_i;\theta)\right)^2
\end{aligned} \tag{2.3}$$

by assuming a normal distribution with mean $\mu_{\mathcal{M}} = \mu(\mathcal{M}(X;\theta)) = \mathcal{M}(X;\theta)$ and standard deviation $\sigma_{\mathcal{M}} = \sigma(\mathcal{M}(X;\theta)) = 1$.

The key takeaway here is that supervised learning tasks can be cast as optimization problems. Specifically, they represent unconstrained optimization problems, whose solutions are typically reached through iterative processes focused on minimizing the loss functions $\mathcal{L}$ described in Equations (2.2) and (2.3), or a surrogate thereof. As a result, if any additional constraint needs to be included, it cannot be accommodated within this framework unless it is inherently embedded into the structure of the learning model $\mathcal{M}$. Nevertheless, since incorporating non-trivial constraints is often unfeasible, this aspect significantly limits the plain application of machine learning in a wide range of real-world applications.

2.1.3 Constrained Supervised Learning

In contrast to subsymbolic approaches, there are symbolic ones. This category includes various algorithms that, despite their diversity, have the shared characteristic of lacking a hidden state. In fact, they leverage explicit high-level operations to find a solution to certain problem instances, relying on methodologies that come from the operational research area – such as constraint programming, mathematical programming, and global optimization – or from the domains of logic and inference.

A well-recognized benefit of symbolic methods is their ability to incorporate constraints inherently within the problem definition, conversely to machine learning techniques where linking the semantics of the constraints to the internal subsymbolic representation is generally impractical. Nonetheless, ensuring the reliability of the outputs yielded by a machine learning model is a valuable feature in various application domains. For example, one could use logical constraints to model the domain knowledge of an expert about, e.g., the taxonomy of the categories to be classified. Similarly, when learning models are used to predict attributes related to physical processes, we might want them to adhere to the relevant laws of physics, chemistry, or biology. Lastly, as in our use cases, when working in areas governed by specific anti-discrimination laws, it is crucial to enforce fairness constraints on the given outputs to guarantee that the system has an appropriate behavior toward minorities and other disadvantaged social groups.

In this regard, over the past decade there has been a significant increase in hybrid approaches that combine symbolic and subsymbolic methods. This integration serves various tasks and objectives. For example, the field of informed machine learning focuses on enhancing subsymbolic models by embedding external knowledge, thereby improving their accuracy and ability to generalize to unseen regions of the space – check [Von Rueden et al., 2021] for a more complete survey. In this dissertation, however, we will concentrate on the area of constrained machine learning, which prioritizes the satisfaction of a set of given constraints before focusing on accuracy. Practically, given a task loss $\mathcal{L}$ and a machine learning model $\mathcal{M}$, the objective is to solve the following constrained optimization problem:

$$\arg\min_{\theta} \mathcal{L}(y, \mathcal{M}(X;\theta)) \qquad \text{s.t.} \qquad \mathcal{M}(X;\theta) \in \mathcal{C} \tag{2.4}$$

where $\mathcal{C}$ represents the feasible region of the space. We will now provide a concise overview of the main types of approach, along with examples for each category.

Regularization-based Methods

The most straightforward way to integrate constraints into a learning process is through loss regularizers. This involves including an additional term in the loss

function that measures "how much" the desired property is being enforced. Some typical examples of regularization are the L1 and L2 norm regularizers in linear regression. To enforce the constraint $\mathcal{M}(X;\theta) \in \mathcal{C}$ shown in Equation (2.4), we can simply substitute the norms with a more complex function $\mathcal{R}(y, \mathcal{M}(X;\theta))$ measuring the deviation from compliance. Eventually, by associating it to a multiplier λ that governs its relative importance with respect to the task loss, we can reformulate the problem as:

$$\arg\min_{\theta} \mathcal{L}(y, \mathcal{M}(X;\theta)) + \lambda \cdot \mathcal{R}(y, \mathcal{M}(X;\theta)) \tag{2.5}$$

Here, the regularizer – sometimes referred to as a "penalizer" since it represents a penalty for not adhering to the constraint – can also be a vector that considers multiple distinct constraints, each associated with an individual multiplier λ_i.

This major benefit of these methods lies in their simplicity. However, it is not always straightforward to modify the loss function for certain machine learning models, which is why regularization techniques are mainly employed in iterative, gradient-based algorithms such as linear regression or neural networks. An interesting work on this topic is [Diligenti et al., 2017], which introduces a framework known as Semantic-based Regularization that translates constraints into fuzzy logic formulas to express prior general knowledge about the environment. Similarly, [Goh et al., 2016] embeds constraints into the learning process to approximately solve a series of non-convex constrained optimization problem. Finally, we mention [Fioretto et al., 2021], who introduces a framework that leverages the theory of Lagrangian duality to develop an iterative method to automatically adjust the λ multipliers during the training of neural networks.

Apart from the benefits, these methods also come with several drawbacks. Firstly, although loss regularizers are theoretically proven to converge after enough iterations, in practice they might fail to enforce constraints even on the training data due to time limitations. Additionally, when dealing with relational constraints, they might limit the application of mini-batch training, since they are typically defined over the entire dataset rather than individual samples. Lastly, in gradient-based learning algorithms, regularizers are required to be differentiable; this might be infeasible sometimes depending on the type of constraints, thus mandating the

use of approximation or surrogate functions which can affect performances due to inaccurate estimates or numerical instability. For a comprehensive discussion of the advantages and disadvantages of regularization-based methods, refer to [Lombardi et al., 2021].

Projection-based Methods

Another group of techniques involves projecting data into the feasible space either before, after, or even during the training process. In the realm of fairness constraints, [Kamiran and Calders, 2011] suggests three distinct pre-processing techniques to achieve feasibility within the training data, which is eventually fed to a machine learning model trained in a plain unconstrained fashion. Nevertheless, although this approach demonstrates reduced unfairness in the downstream task, it can still provide no guarantee that the constraints will be fulfilled during inference, as the model itself can inherently introduce biases during the learning phase. In a similar vein, [Detassis et al., 2021] expands this concept by presenting a framework called Moving Targets, which repeatedly projects data and learns from these refined projections. Although it cannot guarantee constraint satisfaction like the previous approach, this one yields greater consistency, enhancing both task accuracy and compliance. Alternatively, [Zemel et al., 2013] introduces a framework that projects the input variables into an unbiased latent space, and then uses these data for the learning task; while [Cotter et al., 2019], in a different direction, proposes a "proxy-lagrangian" formulation for tackling non-convex optimization problems, which can be addressed using methods such as projected gradient descent.

Typically, these approaches can be applied to a variety of tasks, especially since they can better handle relational constraints thanks to their efficient managing of projections that can consider the dataset as a whole. Moreover, if the constraints are defined on individual samples, a projection mechanism can be integrated at the end of the learning process to ensure full compliance with the requirements. Nonetheless, projection-based methods often demand more computational resources than regularization-based ones, and generally fail to offer theoretical guarantees for relational constraints that rely on statistical definitions.

Model-based Methods

A final category of approaches addresses constraints by directly incorporating them into the definition of the model. That is, instead of training $\mathcal{M}$, they leverage a custom model $\mathcal{M}_{\mathcal{C}}$ that ensures constraint satisfaction by design. In this regard, we can reformulate Equation (2.4) as:

$$\arg\min_{\theta} \mathcal{L}(y, \mathcal{M}_{\mathcal{C}}(X;\theta)) \qquad \text{s.t.} \qquad \mathcal{M}_{\mathcal{C}}(X;\theta) \in \mathcal{C}, \forall (X,\theta) \tag{2.6}$$

Some of these methods rely on logical frameworks in which the data-driven component is incorporated to derive an explicit model formulation that can be easily constrained with minimal effort. For instance, Logic Tensor Networks [Badreddine et al., 2022] aim to apply Real Logic within a neural network setup, whereas Deep-ProbLog [Manhaeve et al., 2018] leverages neural networks to provide probabilistic outputs which are then integrated into a first-order logic program to allow for symbolic reasoning. A different, yet notable class of models within that inherently manages constraints by design are Lattice Models [Garcia and Gupta, 2009]. Lattice models can be used to enforce partial monotonicities and other kinds of shape constraints, and have also been adapted for integration with deep neural networks as described in [You et al., 2017].

On the one hand, model-based methods prove very useful in certain settings due to their theoretical guarantees of constraint satisfaction, with applications in physics-informed machine learning and deontological fairness [Wang and Gupta, 2020]. On the other hand, their specificity confines their utility to a limited range of tasks, making them mostly inapplicable when relational constraints are involved. For this reason, we will not incorporate any model-based approach in our experiments.

2.2 The Role of Correlation

Correlation plays a crucial role in statistics and machine learning. Several tasks are based on the identification of correlations within multivariate datasets. For example, algorithms for data preprocessing such as feature importance and dimensionality reduction strongly depend on correlation measures, respectively to

quantify the mutual impact of input variables and to guide their projection onto lower-dimensional spaces. Likewise, learning models often leverage correlations between features to make predictions, as these patterns and relationships allow generalization to new data.

2.2.1 Correlations vs. Causation

A core element in understanding correlation is how it relates to the concept of causation. Although the latter always implies the former, the opposite does not necessarily hold. This means that the correlation between two or more variables can result from mediation effects, confounding effects, or even sampling noise – in which case we talk about spurious correlations.

Such implications have a strong effect on machine learning algorithms. In fact, a model can exploit the correlations found in the training set that do not represent causal connections. For instance, if two features are correlated due to a hidden confounder, the model could make precise predictions on the training data but perform poorly on new samples for which the confounding relationship is absent. This effect becomes even more pronounced as the input data shows interdependencies between pairs of features. Many machine learning algorithms, in fact, assume that the input data is independently and identically distributed, meaning that each input feature is generated by an independent phenomenon. This assumption is obviously incorrect in most real-world applications, and as a result many models are overspecified, i.e., they have infinite globally optimal solutions due to symmetries. Likewise, the existence of spurious correlations can lead to overfitting and lack of robustness, causing the model to capture noise instead of the actual underlying patterns.

That being said, it is crucial to acknowledge that correlation is nevertheless a strong hint of causality, and under specific assumptions it might be used to enforce certain causal structures. Similarly, independence tests – and, by extension, the assessment of correlations – are foundational elements in most algorithms designed for causal inference and discovery. For a more detailed discussion on this subject, readers are referred to [Nogueira et al., 2022].

2.2.2 Measuring Correlations

Formally, correlation quantifies the degree to which two random variables $A \in \mathcal{A}, B \in \mathcal{B}$ are interdependent. The most popular measure of correlation is the Mutual Information, which is defined as:

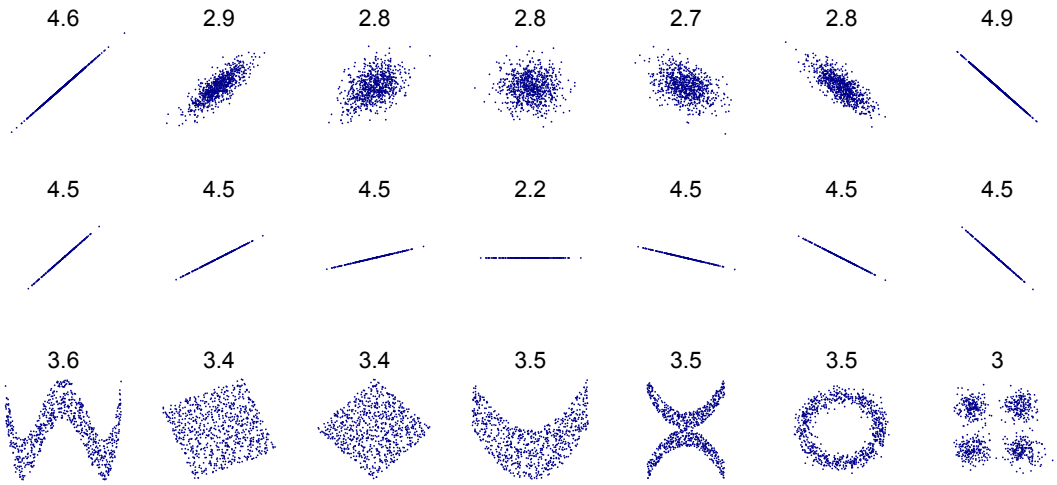
$$I(A, B) = \mathbb{E}\left[\log \frac{P(A, B)}{P(A) \cdot P(B)}\right] \tag{2.7}$$

Although the mutual information has remarkable capacity to capture the relationship between variables, it also has practical limitations that complicate its usage. For instance, while it is straightforward to estimate and optimize it in the discrete case, computing the mutual information on continuous data requires either binning it into discrete intervals – which introduces quantization errors and does not scale well with higher dimensions – or relying on some form of parametric estimation – e.g., variational methods, which in most cases encounter instability issues. In addition to that, mutual information is inherently lacking interpretability, as even its most reliable estimate cannot reveal *how* the variables depend on each other.

Another significant problem with mutual information is that it is defined only for data distributions. In most contexts, however, we do not have access to the original distributions but rather to a set of data points sampled from it. In these cases, computing the mutual information would require a density estimation step, possibly introducing additional errors to the process. To address these limitations, the correlation can be calculated directly from the dataset $(a, b) = \{a_i, b_i\}_{i=1}^{n}$, where each pair of elements $(a_i, b_i) \sim P(A, B)$ is independently sampled from the joint distribution of the corresponding random variables. In this case, we refer to the measure as sample correlation, and typically associate it with a level of significance.

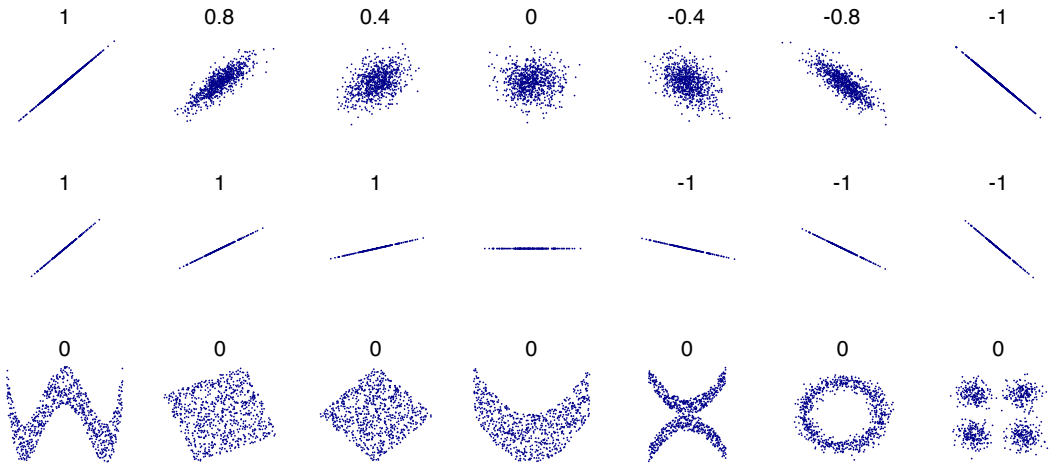
2.2.3 Correlation Coefficients

The most commonly used sample correlation coefficient is Pearson's ρ. It is defined as the covariance between the two vectors $a = \{a_i\}_{i=1}^{n}$ and $b = \{b_i\}_{i=1}^{n}$, normalized



(a) Mutual Information

From: https://commons.wikimedia.org/wiki/File:Mutual_Information_Examples.svg



(b) Pearson's Correlation

From: https://commons.wikimedia.org/wiki/File:Correlation_examples2.svg

Figure 2.2: Correlations computed on different sets of bivariate data.

by the product of their standard deviations, i.e.:

$$\rho(a,b) = \frac{\sum_{i=1}^{n}(a_i - \overline{a}) \cdot (b_i - \overline{b})}{\sqrt{\sum_{i=1}^{n}(a_i - \overline{a})^2} \cdot \sqrt{\sum_{i=1}^{n}(b_i - \overline{b})^2}} \tag{2.8}$$

where $\overline{a}$ and $\overline{b}$ denote the vector averages. On the one hand, Pearson's coefficient is straightforward to compute, as it requires just one pass through the data, and its simplicity becomes a further strength when it comes to interpreting the result. On the other hand, it has the same drawback as many other correlation coefficients, namely it is selective for a specific type of correlation only. In particular, it can only detect linear correlations, thus making it ineffective in capturing the full spectrum of nonlinear relationships between variables, and potentially irrelevant when dealing with categorical values. Figure 2.2 highlights the differences between Pearson's coefficient and Mutual Information. As observed, the latter reports high correlations even with data exhibiting strong non-linearities (last row), whereas Pearson's has the advantage of higher interpretability, especially since its values are bounded within the interval $[-1, 1]$.

Various other indicators designed to assess different types of correlations exist as well. A notable category among them is the rank coefficients, which are based on the concept of rank and are selective to monotonic correlations. The rank of a point x_i within a vector x represents its (ordinal) position after x has been sorted; consequently, the rank R of the entire vector can be obtained by replacing each element x_i with its corresponding rank – e.g., given the vector $x = \begin{bmatrix} 2.4 & 3.1 & 2.7 & 0.8 \end{bmatrix}$, its rank $\mathrm{R}[x]$ would be $\begin{bmatrix} 2 & 4 & 3 & 1 \end{bmatrix}$. The most renowned rank correlation coefficient is Spearman's ρ, which is defined as the Pearson's correlation applied to the rank vectors. Another well-known one, Kendall's τ, is computed as the normalized difference between the total number of concordant and discordant pairs, where a pair $[(x_i, y_i); (x_j, y_j)]$ is defined as concordant if either $x_i < x_j$ and $y_i < y_j$, or $x_i > x_j$ and $y_i > y_j$. Moreover, Spearman's and Kendall's rank correlations can be seen as specific instances of a broader indicator [Kendall, 1948]. Given two antisymmetric bivariate functions ϕ and ψ,

we can build two matrices $A, B \in \mathrm{R}^{n \times n}$ such that:

$$A_{ij} = \phi(a_i, a_j) = -\phi(a_j, a_i) = -A_{ji} \tag{2.9}$$

$$B_{ij} = \psi(b_i, b_j) = -\psi(b_j, b_i) = -B_{ji} \tag{2.10}$$

for all $(i, j) \in \{1, \ldots, n\}^2$. The General Correlation Coefficient is then defined as:

$$\Gamma(a, b) = \frac{\langle A, B \rangle_F}{\|A\|_F \cdot \|B\|_F} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} \cdot B_{ij}}{\sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}^2} \cdot \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} B_{ij}^2}} \tag{2.11}$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product and $\|\cdot\|_F$ is the Frobenius norm. Specifically, we can derive Spearman's ρ using $\phi(x_i, x_j) = \psi(x_i, x_j) = \mathrm{R}[x]_i - \mathrm{R}[x]_j$, whereas Kendall's τ involves $\phi(x_i, x_j) = \psi(x_i, x_j) = \mathrm{sign}(\mathrm{R}[x]_i - \mathrm{R}[x]_j)$. Even Pearson's correlation can be conceptualized within this framework by simply employing the same functions utilized for Spearman's correlation, with the omission of the rank operator, resulting in $\phi(x_i, x_j) = \psi(x_i, x_j) = x_i - x_j$.

Although Γ generalizes all sample-based correlation coefficients, it relies on predetermined mapping functions. This limits its ability to capture a wide range of correlation types, thus failing to meet the specific characteristic of Mutual Information for which a null correlation implies complete independence between the variables. To address this limitation, the Hirschfeld–Gebelein–Rényi (HGR) correlation coefficient was introduced [Rényi, 1959]. Also known as the maximal correlation coefficient, HGR is a non-linear extension of Pearson's obtained via two copula transformations, f and g, which map the input data into a space that maximizes their co-linearity. Formally, it is defined as:

$$\mathrm{HGR}(a, b) = \max_{f,g} \rho(f(a), g(b)) = \max_{f,g} \frac{\mathrm{cov}(f(a), g(b))}{\mathrm{std}(f(a)) \cdot \mathrm{std}(g(b))} \tag{2.12}$$

Unlike the General Correlation Coefficient, HGR optimizes the transformations rather than taking them as fixed. This allows to capture any form of dependency, ensuring complete independence between the two variables when the correlation is zero, although it makes its computation theoretically intractable – see Chapter 3 for a more detailed exploration, as it will focus on a new methodology to estimate HGR

in a configurable and robust manner. We also underline that HGR is potentially able to reduce to Spearman's ρ, since the copula transformations can approximately recreate the rank function R by converting the elements of vectors a and b into their ranks R[a] and R[b]. Nonetheless, this is the only rank correlation indicator that can be generalized using HGR, as we can derive HGR under the lens of the General Correlation Coefficient by setting the functions $\phi(x_i, x_j) = f(x_i) - f(x_j)$ and $\psi(x_i, x_j) = g(x_i) - g(x_j)$. The univariate nature of f and g, though, hinders the consideration of pairs of data points, except when they can be linearly decomposed as in Spearman's coefficient. Nothing prevents from further extending the HGR to include bivariate copula transformations $f, g : \mathbb{R}^2 \mapsto \mathbb{R}$, but such an analysis exceeds the scope of this dissertation, and we state that no similar formulation has been recorded in the literature to the best of our knowledge.

2.3 Fairness in Machine Learning

The field of algorithmic fairness encompasses a wide range of applications and is aimed at removing forms of social inequities and other discriminatory practices that are acquired and perpetrated by subsymbolic models. The emergence of large neural-based models has introduced new challenges, with significant biases related to gender and race emerging in areas such as machine translation and facial recognition, up to newer tasks such as text and image generation from prompts. Nonetheless, unfair behaviors of artificial intelligence systems were and are still present in many other domains, with notable examples such as the COMPAS software, a decision support tool used to assess the likelihood of recidivism that has been shown to incorporate and sustain historical racial biases [Angwin et al., 2022].

The foundations of algorithmic fairness extend back before the rise of artificial intelligence, as it has to be sought in the identification of unfair practices across various fields such as healthcare [Ueda et al., 2024], hiring [Fabris et al., 2023], and education [Kizilcec and Lee, 2022]. Several studies have indicated that extensive social and legal interventions must be combined with research efforts in order to be effective [Morley et al., 2021]; for this reason, research on algorithmic fairness remains active and applies to both technical and academic papers. For a

more comprehensive overview of the subject, refer to [D'Alessandro et al., 2017], [Hutchinson and Mitchell, 2019], and [Caton and Haas, 2020].

2.3.1 Social and Statistical Background

Unfair practices existed long before the rise of automated decision-making systems. Particularly in sectors like insurance and banking, human decision-makers have executed numerous forms of indirect or unintentional discrimination. Among the most well-known and researched is "redlining" [Locke et al., 2021], which refers to the discriminatory act of arbitrarily refusing or restricting services to certain marginalized groups, such as people of color or those with lower income, based on their belonging to a certain residential neighborhood. Other kinds of indirect discrimination include: (*i*) "discrimination on redundant encodings", where an individual with protected attribute Z is rejected based on a theoretically non-discriminatory proxy attribute S, which is correlated to Z but irrelevant to the outcome; (*ii*) "self-fulfilling prophecy", where the intentional selection of unsuitable candidates from the discriminated group results in a bad track record, later employed to support the dismissal of similar candidates; and (*iii*) "reverse tokenism", which involves rejecting a qualified applicant from the privileged group to justify the exclusion of other applicants from the discriminated group [Dwork et al., 2012].

Since machine learning models often learn from biased track records, they not only acquire these biases but can also amplify and automate them. Moreover, traditional methods such as "fairness through unawareness", which involve excluding protected attributes from input data, have proven ineffective as these discriminations tend to be perpetrated through proxies [Dwork et al., 2012]. This has led to the development of various indicators that measure the impact of decisions on protected groups, legally referred to as *disparate impact* [Feldman et al., 2015]. This term became well-known following the American court case of *Griggs v. Duke Power Co.*[1], where the examiners consistently rejected black applicants based on intelligence test results and the requirement of a high school diploma, both features that were specifically tailored for white people. Just as human evaluators were forbidden from practicing indirect discrimination, it is similarly necessary that any machine

[1] `https://supreme.justia.com/cases/federal/us/401/424/`

learning model employed in real-world applications ensures an equitable distribution of both favorable and unfavorable outcomes across all social groups.

Mathematically, this concept can be translated as the degree of statistical independence between the protected attribute Z – e.g., gender, ethnicity, etc. – and the predicted outcome $\hat{Y}$. This is commonly measured using quantitative indicators associated with the protected attribute; however, since ethical issues can hardly be implemented into simple formulas, a wide range of metrics have been introduced, whose selection is often delegated to domain experts and depends on the specific context. Notable metrics include Demographic Parity (DP) and Equalized Odds (EO), designed for binary protected attributes and outcomes and defined as follows:

$$\mathrm{DP}(Z, Y, \hat{Y}) = \left| P(\hat{Y} = +, Z = p) \qquad - P(\hat{Y} = +, Z = m) \right| \tag{2.13}$$

$$\mathrm{EO}(Z, Y, \hat{Y}) = \left| P(\hat{Y} = +, Y = +, Z = p) - P(\hat{Y} = +, Y = +, Z = m) \right| \tag{2.14}$$

where $+$ denotes the positive output class, and p, m represent the privileged and marginalized group, respectively. Demographic Parity essentially attempts to quantify the correlation between predictions $\hat{Y}$ and the protected attribute Z, as it achieves $\hat{Y} \perp\!\!\!\perp Z$ when its value is zero; conversely, Equalized Odds also takes into account the ground truth vector Y, leading to conditional independence $\hat{Y} \perp\!\!\!\perp Z \mid Y$ when its result is null. Besides these, numerous other metrics evaluate various dimensions of fairness in learning systems, one of which is the Disparate Impact Discrimination Index (DIDI) [Aghaei et al., 2019], which we will explore in both our theoretical and empirical studies. The DIDI assesses the degree of disparate impact in regression tasks using a categorical (not necessarily binary) protected attribute. Mathematically, it is formulated as:

$$\mathrm{DIDI}(Z, \hat{Y}) = \sum_{z \in \mathcal{Z}} \left| \frac{\sum_{i=1}^{n} \hat{Y}_i \cdot \mathbb{I}(Z_i = z)}{\sum_{i=1}^{n} \mathbb{I}(Z_i = z)} - \frac{1}{n} \sum_{i=1}^{n} \hat{Y}_i \right| \tag{2.15}$$

where $\mathcal{Z}$ is the support of Z, and $\mathbb{I}$ is the indicator function. A similar expression is also provided for multi-class classification contexts.

2.3.2 Fairness Enforcement Algorithms

While the assessment of fairness simply involves the application of a chosen indicator, enforcing fairness properties in machine learning models is significantly more complex. In particular, fairness constraints fall under the category of relational constraints, namely those imposed on a group of data points rather than on individual samples [Hardt et al., 2016, Fish et al., 2016]. This makes it difficult to handle them since they require access to the entire distribution and might demand the development of specialized algorithms. In certain applications such as fair ranking, there exists a specific taxonomy of these algorithms [Zehlike et al., 2022a, Zehlike et al., 2022b]; generally speaking, however, fairness enforcement methods are categorized into three primary groups based on the timing of the debiasing intervention [Mehrabi et al., 2021] .

Pre-processing Techniques

Pre-processing methods involve the modification of the input data *before* the start of the training process. Such modification is aimed at minimizing discrimination while maintaining the closest distance towards the original dataset in order to minimize information loss. A key advantage of pre-processing methods is their applicability, as they can be used whenever it is possible to change training data without altering the learning procedure. Nonetheless, the strongest drawback is that these methods cannot address potential bias introduced by the algorithms themselves during the learning phase.

An example of this category is presented in [Luong et al., 2011], where the authors employ a K-Nearest Neighbor model to identify significant treatment disparities among neighbors with different protected attributes, later using this information to adjust the sample label so that it better fits the surrounding neighborhood. Similarly, [Kamiran and Calders, 2011] proposes three methods to artificially equilibrate input data using resampling or reweighting techniques to adhere to the given fairness criteria, whereas [Calmon et al., 2017] solves a comparable task by introducing a convex optimization problem aimed at reducing the level of discrimination while minimizing distortions on individual samples. Finally, [Celis et al., 2020] shifts the input data distributions by applying the

principle of maximum entropy, selecting the one that most closely matches the given prior in terms of KL-divergence among all the unbiased distributions.

Post-processing Techniques

In contrast to the previous class, post-processing methods adjust the outputs of the model *after* the training is complete. Similarly to pre-processing methods, their implementation is relatively straightforward, but they can also consider the bias introduced during the training process, hence generally providing stronger guarantees. A significant limitation of these algorithms, though, is that they typically require processing data in batches because fairness metrics are computed across a group of samples rather than on individual data points; consequently, they carry a higher risk of overfitting, as promising candidates might be penalized when grouped with other optimal individuals, and vice versa.

A representative case of these techniques is presented in [Hardt et al., 2016], where the outputs of a classifier are locally adjusted to meet the fairness criteria by means of a linear program; likewise, [Xian et al., 2023] employs a linear program to derive a fair classifier by demonstrating that the minimization of classification error with respect to a desired distribution is equivalent to solving an optimal transport problem. Moreover, post-processing methods are frequently used for ranking tasks. For instance, [Zehlike et al., 2017] modifies the positions of the candidates in the final ranking to satisfy certain lower and upper bound requirements on the presence of subgroup members in the top-k ranking. Similarly, [Singh and Joachims, 2018] proposes a linear program that addresses visibility bias, as studies have shown that candidate exposure diminishes geometrically with their position in the rank. Ultimately, [Biega et al., 2018] considers ordered rankings as well and applies a strategy that dynamically adjusts the results for the same query to ensure equitable attention over time, thus integrating long-term effects into their approach in order to promote more balanced outcomes in the long run.

In-processing Techniques

Finally, in-processing techniques are those that integrate the fairness criteria *throughout* the training phase. These methods are the most challenging to develop

because they usually require modifying the core training process of the machine learning model, employing either loss regularizers or other constraint-imposing approaches. Nonetheless, they are extensively researched in the academic literature since they generally offer the best trade-off between meeting fairness constraints and maintaining task accuracy, despite their increased computational burden.

Many techniques for in-processing fairness enforcement involve minor alterations of standard algorithms, either to ensure certain properties or to guide the learning process toward fair outcomes. For example, [Woodworth et al., 2017] highlights the theoretical limitations of post-processing approaches and develops a simplified algorithm for creating unbiased binary predictors based on second-order moments. In [Kamiran et al., 2010], the authors propose a strategy to adjust the splitting criteria of Decision Trees to mitigate biases in the data, and similar methods are suggested in [Calders and Verwer, 2010] for Naive Bayes classifiers. Likewise, [Zafar et al., 2017] introduces a framework for training fair classifiers using a Disciplined Convex-Concave Program, which is efficiently solvable with known heuristics, while [Komiyama et al., 2018] presents a nonconvex approach that addresses unfairness across multiple protected attributes simultaneously, demonstrating that it can be reduced to exactly solvable convex optimizations. Again, [Donini et al., 2018] introduces the Fair Empirical Risk Minimization framework, extending Support Vector Machines with an extra regularization term to decrease unfairness; similarly, [Padala and Gujar, 2020] integrates a regularization term in the learning process of Neural Networks, demonstrating that they can handle non-convex constraints even with mini-batches, provided that the size of the batch is accurately tuned to avoid generalization issues.

On a final note, we mention some additional techniques that use specific learning models or paradigms to provide outcomes with stronger theoretical guarantees in relation to different criteria. For example, [Wang and Gupta, 2020] uses lattice models to enforce deontological fairness requirements in the form of monotonicity shape constraints, while [Greco et al., 2023] incorporates fairness by expressing constraints within the framework of logic tensor networks. Instead, [Ge et al., 2021] and [Yin et al., 2024] reformulate fairness requirements within the reinforcement learning paradigm, allowing them to better handle the dynamic nature of incoming

data and yield results that are more fair and stable over time.

2.3.3 Categorical vs. Continuous Protected Attributes

All the indicators mentioned in Section 2.3.1 are defined with respect to either binary or, at most, categorical protected attributes. This is due to two primary reasons. On the one hand, fairness definitions have traditionally been associated with social groups defined by categorical attributes such as gender, ethnicity, sexual orientation, religious belief, etc. Even when numerical targets can be considered, as illustrated by the DIDI definition which is applicable to regression tasks, they still require the sensitive information to be framed through categories. On the other hand, fairness is typically quantified using correlation measures, which are significantly easier to compute with categorical rather than continuous variables. As a result, inherently numerical features such as income, age, weight, or even aggregate data like the share of marginalized individuals in a population have consequently been excluded from any fairness application.

Theoretically, continuous protected attributes can be managed through discretization. Nonetheless, this method faces challenges in practical scenarios, as altering the number of bins or even just their boundaries can lead to unpredictable fluctuations in the results, making the approach less reliable and susceptible to manipulation. [Mary et al., 2019] was the first to suggest using the Hirschfeld–Gebelein–Rényi (HGR) Coefficient as an alternative solution to measure fairness under continuous protected attributes. As discussed in Section 2.2.3, HGR can virtually assess any form of correlation, but its exact calculation has been proven intractable. For this reason, the authors employ Kernel Density Estimation techniques to approximate the distributions which generated the data, and eventually compute an estimate of HGR using a theoretical approximation known as Witsenhausen's characterization [Witsenhausen, 1975]. A similar methodology leveraging Witsenhausen's characterization was later introduced in [Baharlouei et al., 2019], where the authors restrict their experiments to categorical protected attributes to avoid estimating the underlying distribution but, interestingly, propose a method to tackle fair clustering besides fair classification. Subsequently, the work by Mary et al. was expanded in [Grari et al., 2020], which introduced a new method

to calculate the indicator leveraging two neural networks to model the copula transformations of HGR. Alternatively, [Jiang et al., 2022] used similar techniques – i.e., discretization and Kernel Density Estimation – to extend the definition of Demographic Parity to the continuous case, rather than employing HGR as a measure of fairness.

We position our research precisely within this specialized area of study, where we aim to advance understanding through various contributions. Initially, we present a novel algorithm for estimating HGR, featuring notable properties and enhanced robustness. Furthermore, we demonstrate that the applicability of HGR when used as a fairness metric might be limited due to specific properties of the indicator which might conflict with certain requirements and tasks. To address this, we slightly alter the semantics of HGR in order to introduce and evaluate a new indicator that extends the notion of disparate impact to the continuous case.

Chapter 3

Non-Linear Correlations

This chapter is dedicated to introducing a novel algorithm for computing the Hirschfeld–Gebelein–Rényi (HGR) correlation coefficient [Rényi, 1959]. The coefficient is initially presented in Section 3.1, highlighting both its advantages and major drawbacks. Next, Section 3.2 examines the current computational approaches for estimating the value of HGR, distinguishing between methods that provide gradient information and those that do not. In Section 3.3, we present our approach based on polynomial expansions, while Section 3.4 shows how to derive an approximation to our method which can ensure faster results and a more stable gradient at the expense of a limitation in the expressive power. In Section 3.5, we delve deeper into the characteristics of our indicator, enumerating all its properties and explaining why they make it more robust and trustworthy than other state-of-the-art methods. Lastly, Section 3.6 showcase an empirical evaluation designed to validate the benefits offered by our method in terms of robustness and explainability. All the proofs, results, and discussions presented in this chapter are drawn from and reworked based on a research paper which is currently under review at a prestigious Artificial Intelligence journal. Certain material has been modified or extended to better fit the thesis format, although the conclusions remain unchanged.

3.1 The Hirschfeld–Gebelein–Rényi Coefficient

The HGR coefficient, also referred to as the maximal correlation coefficient, is a natural yet less known extension of the Pearson coefficient. It is defined as the maximal correlation that can be achieved by transforming random variables into non-linear domains through copula transformations. Formally, given two jointly distributed random variables $A \in \mathcal{A}, B \in \mathcal{B}$, we have:

$$\mathrm{HGR}(A, B) = \sup_{f:\mathcal{A}\mapsto\mathbb{R}, g:\mathcal{B}\mapsto\mathbb{R}} \rho(f(A), g(B)) \tag{3.1}$$

where $\rho(\cdot, \cdot)$ denotes the Pearson coefficient, and f and g are the copula transformations belonging to the Hilbert space of all possible mapping functions.

Practically, HGR seeks to determine the optimal pair of functions f and g that project the initial variables into a new space where their co-linearity is maximized. For instance, if the relationship between A and B is circular, then the copula transformations will be of the type $f : a \mapsto a^2$ and $g : b \mapsto b^2$ in order to capture the quadratic interaction from both sides that is typical of circumferences. Similarly, if the relationship is $B = \sin(A)$, we obtain $f : a \mapsto \sin(a)$ and $g : b \mapsto b$, resulting in a final correlation of $\rho(f(A), g(B)) = \rho(\sin(A), \sin(A)) = 1$.

Given Equation (3.1), there are infinite viable copula transformations for this problem. Indeed, as the Pearson correlation is invariant to both translation and scaling, any possible solution which retains the same shape is equally valid. In order to break these symmetries, we can impose additional conditions on the copula transformations and arrive at a unique[1] solution. More specifically, we require zero-centered copula transformations with unitary standard deviation, obtaining:

$$\mathrm{HGR}(A, B) = \sup_{\substack{f:\mathcal{A}\mapsto\mathbb{R}, g:\mathcal{B}\mapsto\mathbb{R} \\ \mathbb{E}[f(A)]=\mathbb{E}[g(B)]=0 \\ \mathbb{E}\left[f^2(A)\right]=\mathbb{E}\left[g^2(B)\right]=1}} \mathbb{E}\left[f(A) \cdot g(B)\right] \tag{3.2}$$

[1]To be more precise, two distinct solutions exist. In fact, given the optimal functions f^* and g^*, one can always define the functions $f'(x) = -f^*(x)$ and $g'(x) = -g^*(x)$, which produce identical outcomes since the Pearson correlation feature a product at the numerator. However, introducing a constraint to disrupt this symmetry is challenging and often does not offer further insight or computational benefit, since the two solutions are entirely equivalent, hence we will consider this approach as the most suitable one.

where the Pearson correlation can be replaced by the mean of the product, as both the mean and standard deviation terms no longer have an influence.

Notably, the definition of HGR allows to derive three important properties:

$$\begin{aligned} &\mathrm{HGR}(A,B) \in [0,1] \\ &\mathrm{HGR}(A,B) = 1 \iff \exists f,g \mid P(f(A) = g(B)) = 1 \\ &\mathrm{HGR}(A,B) = 0 \iff A \perp\!\!\!\perp B \end{aligned} \tag{3.3}$$

That is, the domain of HGR is bounded between 0 and 1. This makes it different from other known correlation metrics that are defined in $[-1, 1]$ as it is unable to determine the direction of the correlations, hence limiting to its strength in absolute value. Within this domain, higher values denote a stronger degree of dependence, while lower values reflect weaker dependence. In particular, HGR reaches its maximum value only when there exist two deterministic functions f and g making the random variables identical, while it reaches its minimum value only when A and B are independent. This last property is especially significant, as other measures of sample correlation do not guarantee it; for example, two variables might be dependent on each other even if their Pearson's correlation is zero, as their relationship might be exclusively non-linear.

3.1.1 The Issue of Uncomputability

As previously discussed, HGR is an effective tool for measuring the dependence between two variables. However, its calculation poses difficulties, as it requires to optimize over an infinite set of infinite-dimensional elements – that is, all potential f and g functions. Among the well-founded methods proposed to estimate this coefficient, one notable approach is by Witsenhausen [Witsenhausen, 1975], which uses the second singular value σ_2 of a carefully selected matrix:

$$Q = \frac{P(A,B)}{\sqrt{P(A)} \cdot \sqrt{P(B)}} \tag{3.4}$$

where $P(A,B)$ represents the joint distribution and $P(A), P(B)$ the marginal distributions, respectively.

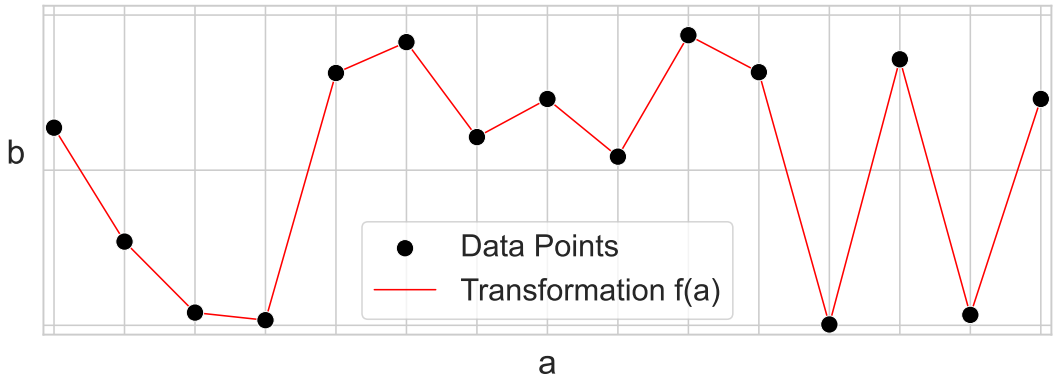


Figure 3.1: Example of overfitting in the computation of sample HGR.

It should be mentioned, however, that the exact value of $\sigma_2(Q)$ is theoretically uncomputable. An approximation for the lower bound based on the χ^2 value has been suggested by [Mary et al., 2019]. Nonetheless, in addition to the fundamental theoretical error, their method also requires to estimate the distributions of A and B from the available samples, thereby introducing a further layer of inaccuracy to the whole process.

3.1.2 Distribution vs. Samples

In principle, a sample version of HGR can be readily derived by substituting the theoretical value of the Pearson correlation in Equation (3.2) with its sample-based counterpart. The two copula transformations f and g need to be redefined to act on vectors as well, specifically:

$$f(a) = \{f(a_i)\}_{i=1}^{n} \qquad g(b) = \{g(b_i)\}_{i=1}^{n} \tag{3.5}$$

However, this characteristic makes the indicator prone to overfitting on any given sample $(a, b) \sim P(A, B)$, where the pairs can be seen as deterministic specifications of either a $a \mapsto b$ or $b \mapsto a$. For instance, consider the dataset presented in Figure 3.1, where the a points are equidistant and the b points are sampled randomly from a uniform distribution. Since it is possible to define the new transformations f and g based only on their values at the sampled points, an unrestricted model could return two copula transformations such as $f : a_i \mapsto b_i$ and $g : b_i \mapsto b_i$, resulting

in a maximal correlation $\text{HGR}(a, b) = \rho(f(a), g(b)) = \rho(\{b_i\}_{i=1}^n, \{b_i\}_{i=1}^n) = 1$, even though b is generated from a distribution independent of A.

In Figure 3.1, we show how this task can be easily accomplished using a piecewise-linear function that precisely fits the data, but we underline that this concept applies to any model capable of exact interpolation on a dataset, ranging from piecewise-constant to polynomial or spline interpolations. In real-world scenarios, tackling these issues requires the use of suboptimal computational methods that can balance the trade-off between bias and variance.

3.2 Computational Methods for HGR

In this section, we will present several algorithms that have been suggested for approximating HGR using samples. Various theoretical relaxations, computational approximations, and semantic modifications have been introduced for this task. The common thread in this comprehensive body of work is the combination of the expressive power provided by non-linear kernel methods with the well-established theoretical and practical benefits of linear algebra.

The major difference between these algorithms lies in their ability to provide gradient information at the end of the procedure. Although this feature is often not essential for applications where correlation measurement is sufficient, it becomes highly beneficial whenever such correlation must be constrained up to certain values using regularization-based methods – see Section 2.1.3 for a more thorough examination of this subject.

3.2.1 Gradient-Free Algorithms

The first approach proposed to estimate HGR was the Alternating Conditional Expectations (ACE) algorithm [Breiman and Friedman, 1985]. Although it can be generally used as a method for finding optimal transformations between the target and input variables in regression tasks, in the context of bivariate datasets it grounds to the computation of HGR. The aim of the algorithm is to identify the best pair of zero-centered transformations f and g which minimize the fraction of

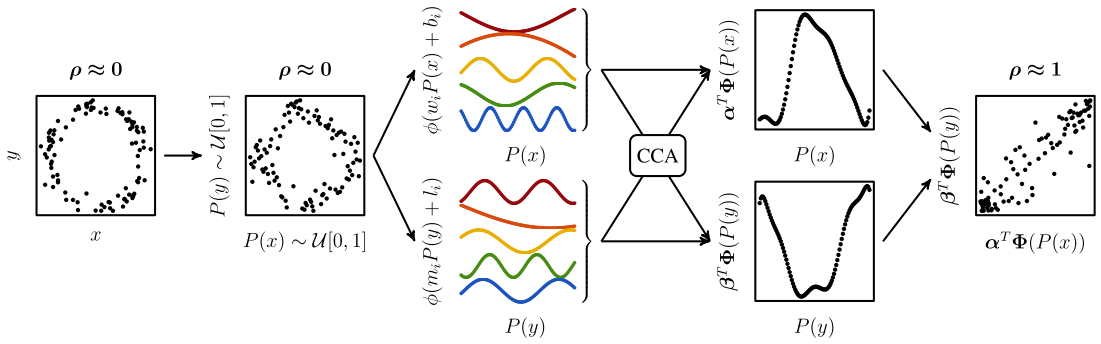


Figure 3.2: Pipeline of the Randomized Dependence Coefficient.
From: *The Randomized Dependence Coefficient* [Lopez-Paz et al., 2013]

unexplained variance, i.e.:

$$\underset{\substack{f:\mathcal{A}\mapsto\mathbb{R},g:\mathcal{B}\mapsto\mathbb{R} \\ \mathbb{E}[f(A)]=\mathbb{E}[g(B)]=0}}{\arg\min} \frac{\mathbb{E}\left[f(A) - g(B)\right]^2}{\mathbb{E}\left[g^2(B)\right]} \tag{3.6}$$

This process involves alternately fixing f and g while iteratively resolving the following subproblems until the fraction of unexplained variance converges within a specified tolerance:

$$g(B) = \mathbb{E}\left[f(A) \mid B\right] \tag{3.7}$$

$$f(A) = \mathbb{E}\left[g(B) \mid A\right] \tag{3.8}$$

Similar to other algorithms, ACE necessitates access to distributions $P(A)$, $P(B)$, and $P(A, B)$, which further restricts its applicability in practical scenarios.

Other interesting gradient-free techniques for the computation of a generalized correlation coefficient have been suggested over time. Noteworthy examples include the Distance and Brownian Correlation [Székely and Rizzo, 2009], Kernel Independent Component Analysis [Bach and Jordan, 2003], Kernel Canonical Correlation Analysis [Hardoon and Shawe-Taylor, 2008], and Hilbert-Schmidt Independence Criterion [Gretton et al., 2005, Póczos et al., 2012].

As a final mention, we reference the Randomized Dependence Coefficient (RDC) [Lopez-Paz et al., 2013]. The RDC estimates HGR by choosing the pair which yields the highest correlation among randomly-calibrated sinusoidal projec-

tions of the input variables. Specifically, as displayed in Figure 3.2, the algorithm utilizes k different sinusoidal functions obtained using a random scaling term ω, and eventually employs Canonical Correlation Analysis (CCA) to identify the pair with maximal correlation. This method is similar to ours but has two main shortcomings: (*i*) it relies on random projections, which limits its robustness and interpretability, and (*ii*) its procedure does not provide gradient information, making it unsuitable for enforcement scenarios. Nonetheless, due to these similarities, we opted to use it as a baseline in some of our experiments.

3.2.2 Kernel-Density Estimation

[Mary et al., 2019] was the first paper to devise a novel computational approach for HGR that ensures differentiability. In their approach, the authors use Kernel Density Estimation (KDE) techniques to approximate the input and output distributions, later using them to compute the χ^2-divergence, which they prove to be a reliable upper bound of the Witsenhausen's characterization of HGR [Witsenhausen, 1975]. Given that the χ^2-divergence provides a smooth approximation, it can be effectively used as a loss regularizer during the neural network training process. Remarkably, the paper also empirically demonstrates how this approach is effective when applied to mini-batches, making it suitable for deployment in deep learning frameworks.

Despite its distinct advantages, the proposed HGR-KDE method also has several limitations. Firstly, akin to our previous discussion on Mutual Information, correlation metrics based on kernel-density estimation methods tend to be largely uninterpretable, as they model distributions but offer no substantial insight into the structure of the copula transformations. Secondly, although the authors assert their approach is non-parametric, we acknowledge that KDE algorithms require a configuration that can significantly influence the quality of the resulting solution. Furthermore, this configuration is often opaque and challenging to determine a priori, as it lacks a direct correlation with observable outcomes.

3.2.3 Neural Networks

Another gradient-based method to calculate HGR is proposed in [Grari et al., 2020]. This is achieved through an adversarial framework where two neural networks are used to estimate copula transformations. A primary model (f) attempts to predict the protected variable mapping while a secondary one (g) predicts the output target one. Eventually, the system is trained to maximize the resulting correlation.

Compared to the KDE method, this HGR-NN approach is slower but significantly more effective. Moreover, the neural-based copula transformations offer a greater level of interpretability compared to kernel-density ones, as they enable visualization of the response function by scanning the input space, although this level of interpretability cannot be framed in closed-form due to the inherent complexity of neural networks. When it comes to configurability, neural networks require as well the definition of certain specifications, such as the hyperparameters of the layers and the optimizer. Nevertheless, the effects of these specifications are more predictable in advance since they are more directly correlated with the trade-off between bias and variance, hence allowing for a more straightforward calibration. A major drawback of neural networks is their intrinsic non-determinism, which can be particularly problematic in fairness applications where specific guarantees might be necessary. Finally, we mention that this method has been shown as well to work properly with mini-batches, and in particular it can be more efficient when applied to data sampled from the same process, as it is possible to apply fine-tuning techniques to the pre-trained transformations.

3.3 Kernel-Based HGR

The core idea of our approach is to represent f and g by means of finite-degree polynomials. Formally, let us consider two vectors (a, b) sampled from the joint distribution of A and B. Our finite variance models are expressed as weighted polynomial expansions $\mathbf{P}_x^d \cdot \omega$, where x refers to the input vector, d denotes the degree of the polynomial kernel, and ω is a d-dimensional vector of coefficients

assigned to each polynomial degree. Specifically:

$$\mathbf{P}_x^d \cdot \omega = \begin{bmatrix} x_1 & x_1^2 & \dots & x_1^d \\ x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_n^2 & \dots & x_n^d \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_d \end{bmatrix} \tag{3.9}$$

Accordingly, our kernel-based HGR variant is defined as:

$$\text{HGR-KB}(a, b; h, k) = \max_{\alpha,\beta} \rho(\mathbf{P}_a^h \cdot \alpha, \mathbf{P}_b^k \cdot \beta) \tag{3.10}$$

where the copula transformations $f(a)$ and $g(b)$ are replaced with $\mathbf{P}_a^h \cdot \alpha$ and $\mathbf{P}_b^k \cdot \beta$, respectively. Here, h and k are two positive integers that represent the order of polynomial expansions for both variables. As their specification is designated to the user, these hyperparameters provide a way to adjust the flexibility of the indicator, balancing both the trade-off between bias and variance as well as its expressiveness against computational complexity. The remainder of this section addresses the technical aspects concerning the calculation of the indicator, followed by a discussion on its properties and practical advantages.

3.3.1 Optimization Problem Formulation

Addressing Equation (3.10) poses a challenge due to the presence of several non-linearities in the definition of Pearson's coefficient. Nevertheless, we can observe that the sample Pearson correlation $\rho(a, b)$ can be obtained as follows by finding the unique solution of an unconstrained least-square optimization problem:

$$\arg\min_r \frac{1}{n} \left\| \frac{a - \mu(a)}{\sigma(a)} \cdot r \; - \frac{b - \mu(b)}{\sigma(b)} \right\|_2^2 \tag{3.11}$$

where μ and σ denote the mean and standard deviation operators, respectively – see proof in Appendix B.1. By substituting this into Equation (3.1), we get to the

subsequent bi-level optimization problem:

$$\max_{f,g} \arg\min_{r} \frac{1}{n} \left\| \frac{f(a) - \mu(f(a))}{\sigma(f(a))} \cdot r \; - \frac{g(b) - \mu(g(b))}{\sigma(g(b))} \right\|_2^2 \tag{3.12}$$

Given that the copula transformations exhibit finite and strictly positive variance, this represents an alternative yet equivalent sample version of HGR, from which the correlation coefficient can be obtained as the optimal value r^*, which is guaranteed unique thanks to the strict convexity of the underlying problem.

Nonetheless, having a bi-level optimization problem is cumbersome to handle from a computational viewpoint. Therefore, in Appendix B.2 we prove that the objectives of the outer and inner optimization problems are aligned, thus making it possible to simplify the formulation to:

$$\arg\min_{f,g,r} \frac{1}{n} \left\| \frac{f(a) - \mu(f(a))}{\sigma(f(a))} \cdot r \; - \frac{g(b) - \mu(g(b))}{\sigma(g(b))} \right\|_2^2 \tag{3.13}$$

3.3.2 Plugging Polynomial Kernels

Let us substitute our polynomial models in Equation (3.13). What we get is:

$$\arg\min_{\alpha,\beta,r} \left\| \frac{\mathbf{P}_a^h \cdot \alpha - \mu(\mathbf{P}_a^h \cdot \alpha)}{\sigma(\mathbf{P}_a^h \cdot \alpha)} \cdot r \; - \frac{\mathbf{P}_b^k \cdot \beta - \mu(\mathbf{P}_a^k \cdot \beta)}{\sigma(\mathbf{P}_a^k \cdot \beta)} \right\|_2^2 \tag{3.14}$$

where the scaling factor $1/n$ can be omitted without altering the solution.

Looking at this equation, we can notice that the $\mu(\cdot)$ terms may be easily removed since the mean operator is invariant to translation. Consequently, we can precompute the zero-centered the polynomial kernels $\widetilde{\mathbf{P}}_a^h$ and $\widetilde{\mathbf{P}}_b^k$ and use them in place of the numerators, leading to:

$$\arg\min_{\alpha,\beta,r} \left\| \frac{\widetilde{\mathbf{P}}_a^h \cdot \alpha}{\sigma(\mathbf{P}_a^h \cdot \alpha)} \cdot r \; - \frac{\widetilde{\mathbf{P}}_b^k \cdot \beta}{\sigma(\mathbf{P}_a^k \cdot \beta)} \right\|_2^2 \tag{3.15}$$

The same reasoning cannot be applied to the standard deviation operator, as its value is more complex to compute. Nonetheless, since r appears as the multiplicative factor of a scale-invariant term, we can eliminate the denominator

$\sigma(\mathbf{P}_a^h \cdot \alpha)$ by incorporating it into the coefficient vector along with r as follows:

$$\widetilde{\alpha} = \frac{\alpha}{\sigma(\mathbf{P}_a^h \cdot \alpha)} \cdot r \tag{3.16}$$

Finally, the only method to eliminate $\sigma(\mathbf{P}_a^k \cdot \beta)$ is to impose a constraint that sets its value to 1, resulting in the following optimization problem:

$$\arg\min_{\widetilde{\alpha},\beta} \left\| \widetilde{\mathbf{P}}_a^h \cdot \widetilde{\alpha} - \widetilde{\mathbf{P}}_b^k \cdot \beta \right\|_2^2 \qquad \text{s.t.} \qquad \sigma(\widetilde{\mathbf{P}}_b^k \cdot \beta)^2 = 1 \tag{3.17}$$

Specifically, we impose the constraint on the variance instead of the standard deviation and replace $\mathbf{P}_b^k$ with its zero-centered counterpart $\widetilde{\mathbf{P}}_b^k$. This approach retains the same solution, but simplifies its computational modeling by eliminating the need to calculate square-root terms and allowing the use of a single matrix.

3.3.3 Solution of the Problem

Although the quadratic objective may appear straightforward, finding the optimal solutions $\widetilde{\alpha}^*$ and β^* of Equation (3.17) is challenging due to the equality constraint imposed on the quadratic function $\sigma(\widetilde{\mathbf{P}}_b^k \cdot \beta)^2$. This prevents from the application of the Quadratically Constrained Quadratic Program (QCQP) framework, as it can only handle negative inequalities when dealing with positive-semidefinite problems like ours. However, in Appendix B.3 we show that the problem admits a convex formulation, thus ensuring the existence of a globally optimal solution. This allows the application of any global optimization method despite the lack of a closed-form solution and, specifically, we rely on the implementation of the Trust Region Method from [Conn et al., 2000] provided by the `scipy.optimize` package.

We underline that $\widetilde{\alpha}^*$ does not ensure a unitary standard deviation for the copula transformation, as it inherently embeds the term r^* and the standard deviation $\sigma(\widetilde{\mathbf{P}}_a^h \cdot \alpha)$. This prevents us from directly employing the product mean as in Equation (3.2) to compute the final correlation. However, in Appendix B.4 we demonstrate that the standard deviation of $\widetilde{\mathbf{P}}_a^h \cdot \widetilde{\alpha}^*$ corresponds exactly the optimal

correlation r^*, thus allowing us to determine the value of HGR-KB as:

$$\text{HGR-KB}(a, b; h, k) = \sigma(\widetilde{\mathbf{P}}_a^h \cdot \widetilde{\alpha}^*) \tag{3.18}$$

once the optimization problem has been solved. Otherwise, an alternative approach would be to directly compute the Pearson correlation. Given its invariance to both translation and scaling, additional transformations are unnecessary and we get:

$$\text{HGR-KB}(a, b; h, k) = \rho(\widetilde{\mathbf{P}}_a^h \cdot \widetilde{\alpha}^*, \widetilde{\mathbf{P}}_b^k \cdot \beta^*) \tag{3.19}$$

3.4 Single-Kernel HGR

Suppose that we only aim to measure a type of functional dependency as described in $a \overset{f}{\mapsto} b$, rather than allowing the algorithm to consider codependencies. In order to achieve this, it is sufficient to eliminate the copula transformation g, which in our scenario means selecting a first-order polynomial for the second kernel.

Our formulation simplifies to:

$$\arg\min_{\widetilde{\alpha},\beta} \left\| \widetilde{\mathbf{P}}_a^h \cdot \widetilde{\alpha} - \beta(b - \mu(b)) \right\|_2^2 \qquad \text{s.t.} \qquad \sigma(\beta b)^2 = 1 \tag{3.20}$$

where the constraint fully determines the value of β, making it equal to $1/\sigma(b)$.

Therefore, we can rewrite Equation (3.20) as:

$$\arg\min_{\widetilde{\alpha}} \left\| \widetilde{\mathbf{P}}_a^h \cdot \widetilde{\alpha} - \frac{b - \mu(b)}{\sigma(b)} \right\|_2^2 \tag{3.21}$$

which allows for more efficient solutions based on a wide range of ad-hoc algorithms as it represents an unconstrained least-squares problem.

3.4.1 A Further Approximation

Despite this setup being limited to quantifying correlations in functional form, its computational benefits are significant enough that we decided to use it as the basis for a simplified version of our indicator, which we called Single-Kernel HGR.

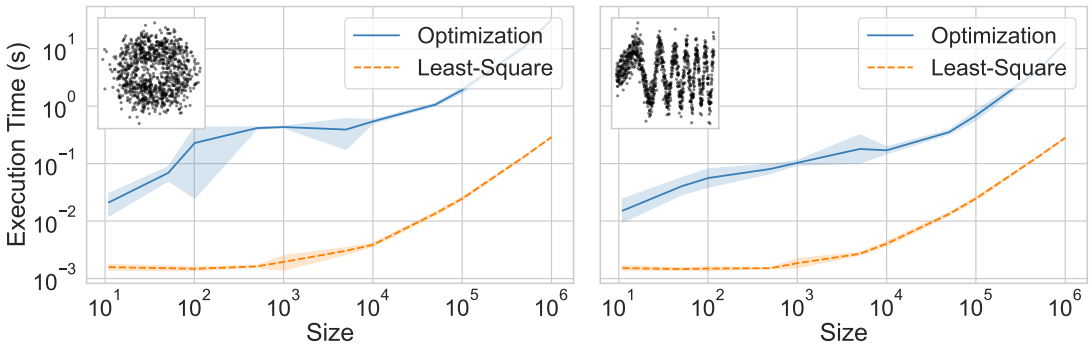


Figure 3.3: Time required to compute HGR-SK using Least-Square vs. Global Optimization algorithms.

HGR-SK is obtained by evaluating HGR-KB with orders $(d, 1)$ and $(1, d)$, then selecting the maximal outcome:

$$\text{HGR-SK}(a, b; d) = \max\left\{\text{HGR-KB}(a, b; d, 1), \text{HGR-KB}(a, b; 1, d)\right\} \tag{3.22}$$

In this variation, d controls the degree of both polynomial expansions, allowing the indicator to capture functional dependencies only, although in both directions.

In particular, we can address the optimization problems using Equation (3.21), and eventually compute the optimal value using the definition of Pearson's coefficient as in Equation (3.19), leading to:

$$\begin{aligned} \text{HGR-SK}(a, b; d) &= \max\left\{\rho(\widetilde{\mathbf{P}}_a^d \cdot \widetilde{\alpha}^*, b), \rho(a, \widetilde{\mathbf{P}}_b^d \cdot \widetilde{\beta}^*)\right\} \\ \widetilde{\alpha}^* &\in \arg\min_{\widetilde{\alpha}} \left\| \widetilde{\mathbf{P}}_a^d \cdot \widetilde{\alpha} - \frac{b - \mu(b)}{\sigma(b)} \right\|_2^2 \\ \widetilde{\beta}^* &\in \arg\min_{\widetilde{\beta}} \left\| \widetilde{\mathbf{P}}_b^d \cdot \widetilde{\beta} - \frac{a - \mu(a)}{\sigma(a)} \right\|_2^2 \end{aligned} \tag{3.23}$$

or, alternatively, using Equation (3.18) instead of Pearson's correlation.

3.4.2 Advantages and Disadvantages

The primary strength of the Single-Kernel formulation lies in its speed. Solving an unconstrained least-square problem is a well-understood task in linear algebra,

with highly-optimized computational routines available. Figure 3.3 demonstrates that employing least-square solvers offers an improvement of nearly two orders of magnitude over global optimization using trust region methods. Similar conclusions can be drawn from the experiments shown in Section 3.6. Furthermore, while one might argue that solving a least-square problem does not scale efficiently as data size increases, it must be considered that constraint enforcement through regularization techniques could be effective on mini-batches as well, hence the dimensionality of the data can be arbitrarily chosen to balance speed and accuracy during training.

Another interesting feature of HGR-SK is that its procedure is completely differentiable. Notably, automatic differentiation frameworks support a differentiable least-squares operator, such as `tf.linalg.lstsq` in TensorFlow and `torch.linalg.lstsq` in PyTorch. Consequently, the Single-Kernel variant has a well-defined gradient and, in addition to that, it can potentially allow for precise constraints on individual elements of the coefficient vectors $\widetilde{\alpha}^*$ and $\widetilde{\beta}^*$ – refer to Section 4.3.3 for a more detailed discussion on fine-grained constraints within the context of Generalized Disparate Impact.

In terms of limitations, the primary issue with HGR-SK stems from its inability to detect strong non-linear dependencies between the two vectors. An empirical example of this is presented in Section 3.6, particularly in the circular dataset which features a quadratic relationship between both vectors. However, experiments on real-world data shown in Section 4.1 indicate that this limitation has minimal effects on highly complex data.

3.5 Properties of Kernel-Based Methods

We claim that our indicator HGR-KB, along with its Single-Kernel variant, has several properties that make it significantly more suitable for practical implementations than other options. Table 3.1 provides a summarization of these properties, which are further examined in the following subsections.

Method	HGR-KB	HGR-SK	HGR-NN	HGR-KDE	RDC
Expressivity	f, g	f **or** g	f, g	f, g (distributions)	f, g
Interpretability	✓	✓	visualization only	×	✓
Configurability	✓	✓	architecture only	×	×
Differentiability	✓	✓	✓	✓	×
Determinism	✓	✓	×	✓	×

Table 3.1: Properties of HGR-KB and HGR-SK compared to three alternative techniques for computing HGR.

3.5.1 Expressivity

Regarding expressivity, both our method and the adversarial approach can be considered universal approximators. Theoretically, polynomial expansions are known for their higher numerical instability compared to Neural Networks; however, in practice, extremely expressive models may lead to overfitting, as demonstrated by our experiments, hence low degrees are usually more reliable and preferable. Conversely, the kernel-density estimation method depends on a bound, complicating the expressivity analysis, whereas the Randomized Dependence Coefficient makes use of a single sinusoidal function, thus being inherently limited in expressivity.

3.5.2 Interpretability

The use of polynomial kernels guarantees an easier interpretability of our method, on par with the Randomized Dependency Coefficient. Figure 3.4 shows an example of how the optimized kernels can be both displayed and analytically examined based on their interpretable coefficients. The original data, shown on the left, exhibit almost no (linear) correlation; in the central figures, we plot the learned copula transformations along with the corresponding coefficients for the polynomial terms; finally, the projected data on the right demonstrate a much stronger correlation. Notably, our method correlates the magnitude of each component with its respective degree, allowing in fact a mathematical examination of the outcome on top of the visual one. For instance, in the depicted scenario, the quadratic relationship between variables is made evident by the stronger magnitude of the second coefficient with respect to the first one, both in α and in β. This characteristic is missing in the

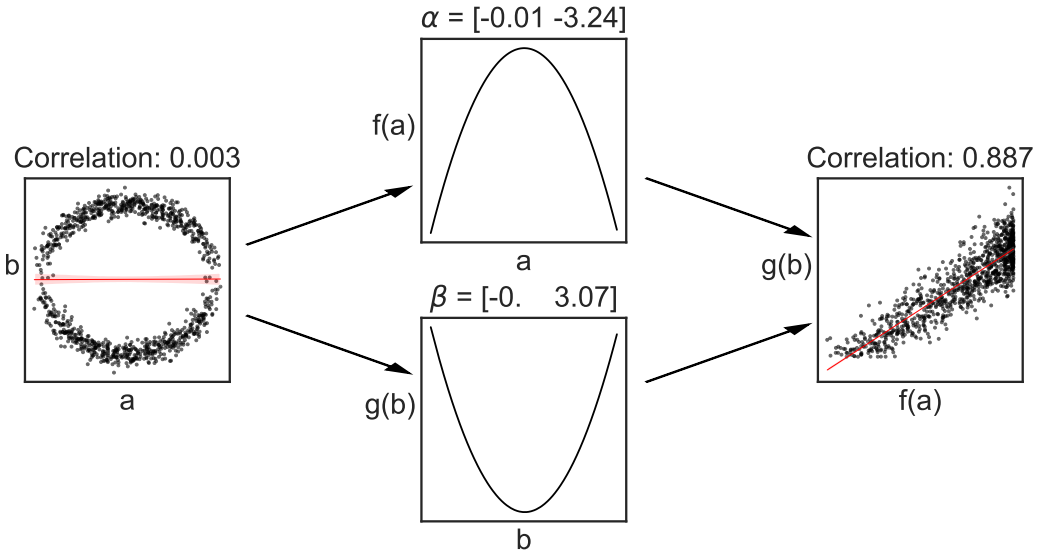


Figure 3.4: Example of kernel computation using HGR-KB.

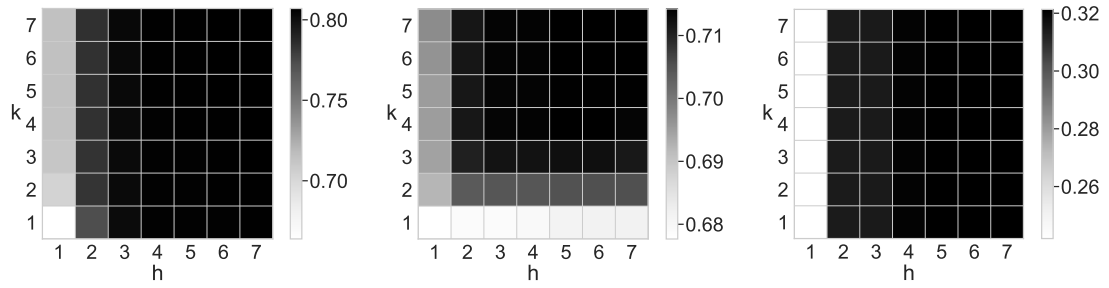
KDE approach, while the adversarial method HGR-NN can only offer visualization due to the inherent sub-symbolic nature of Neural Networks.

3.5.3 Configurability

A specific property which results from the definition of our method, is that HGR-KB increases monotonically with respect to the degrees of the kernel; i.e.:

$$\text{HGR-KB}(a, b; p, q) \geq \text{HGR-KB}(a, b; h, k), \forall p \geq h, q \geq k \tag{3.24}$$

This property, which is proven in Appendix B.5, improves the understanding of the balance between bias and variance and between expressiveness and computational costs by analyzing the potential improvements brought by higher degrees. An empirical analysis of this property is presented in Figure 3.5, where the correlation between a protected attribute (z) and the target (y) is measured across three different benchmark datasets. Specifically, we used *2015 US Census*, *Communities & Crimes*, and *Adult Census Income*, with the protected attributes and targets mentioned in Appendix A.2. In the figure, darker shades represent a higher correlation, thereby demonstrating the monotonically increasing trend of



(a) 2015 US Census (b) Communities & Crimes (c) Adult Census Income

Figure 3.5: Computation of HGR-KB$(a, b; h, k)$ with varying h and k on three benchmark datasets.

the results in both directions, as the color becomes darker towards higher degrees – i.e., the upper right corner. Interestingly, we also observe that the correlation computed in the *Adult Census Income* dataset remains stable with respect to the increments in the k degree, since the target variable is binary and, therefore, not affected by polynomial expansions.

The possibility of selecting the degrees of the kernel offers a clear and well-understood mathematical approach to manage the bias-variance trade-off, which is essential for the robustness of the method. Specifically, considering the mentioned monotonicity property, it is evident that lower degrees are often adequate to capture most non-linear correlations, and that in general we could locate the "optimal" configuration (h, k) using a procedure similar to the elbow method. For example, by visually inspecting the three grids in Figure 3.5, the points $(2, 3)$, $(3, 2)$, and $(2, 1)$ emerge as those beyond which there is little or no improvement in the obtained correlation value for each respective dataset. This conclusion is drawn from the observation that the colors no longer darken after those coordinates, indicating that greater degrees fail to capture more significant dependencies[2]. In contrast, the effects of the KDE parameters are less predictable, while only loose guidance can be provided to the RDC coefficient due to its inherent randomness. Conversely,

[2]In this context, we must also underline that the optimal value for measurement might not be the same when enforcing. In fact, when using any HGR measure to enforce a constraint at training time, the model might learn to move the correlations in a space where the algorithm cannot detect them rather than actually cancelling them. In our case, this corresponds to moving the correlations to higher degrees; for this reason we will use $h = k = 5$ throughout all our experiments at the cost of introducing a slight computational overload.

the adversarial approach offers substantial control, but through a less transparent mechanism because of the complexity of neural networks.

3.5.4 Differentiability

As mentioned in Section 2.1.3, several approaches for imposing constraints in machine learning leverage loss regularizers. In particular, in our scenario we would like to compute the correlation between a protected attribute vector z and a prediction vector $\hat{y}(\theta)$, which can be naturally differentiated with respect to the parameter vector θ of the selected machine learning model. Eventually, this correlation should be limited within a specific threshold τ, i.e.:

$$\mathrm{HGR}(z, \hat{y}(\theta)) \leq \tau \tag{3.25}$$

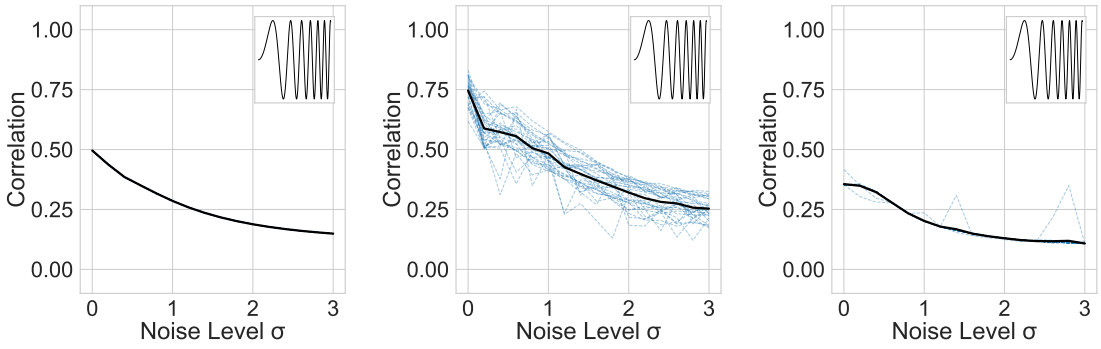
When used in conjunction with Gradient Descent, this constraint can be incorporated into the task objective, resulting in the following custom loss:

$$\mathcal{L}(y, \hat{y}(\theta)) + \lambda \cdot \max\{0, \mathrm{HGR}(z, \hat{y}(\theta)) - \tau\} \tag{3.26}$$

where $\mathcal{L}$ represents the task loss, y denotes the ground truths, and λ is a Lagrangian multiplier that can be either set to a fixed value or adjusted automatically.

However, this method requires the constraint to be differentiable in order to work properly. Unfortunately, for HGR-KB we cannot fully ensure this, since the derivation of $\widetilde{\alpha}^*$ and β^* from Equation (3.17) depends on a numerical optimization process, which inherently provides no gradient information. Nevertheless, both Equation (3.18) and Equation (3.19) are differentiable, thus we can use them to obtain a valid subgradient of HGR-KB$(z, \hat{y}(\theta); h, k)$. In the real-world experiments described in Section 4.1, we empirically demonstrate that this subgradient is useful and can effectively guide the learning process towards regions where the constraint is met. Moreover, as mentioned in Section 3.4.2, the computation procedure for HGR-SK$(a, b; d)$ is based on an unconstrained least-square problem, therefore it has a well-defined gradient since automatic differentiation frameworks support a differentiable least-squares operator.

Regarding alternative approaches, HGR-KDE is based on an inherently differ-



(a) HGR-KB (b) HGR-NN (c) RDC

Figure 3.6: Effects of algorithm stochasticity in three different techniques to estimate HGR.

entiable process, which allows for a straightforward regularization. HGR-NN relies instead on a subgradient, although empirical evaluations by [Grari et al., 2020] have demonstrated its efficacy. In contrast, the computational method of RDC does not provide any gradient information.

3.5.5 Determinism

We also remark that the use of exact optimization techniques ensures that our indicator remains completely deterministic once the kernel degrees are fixed. This characteristic is common with the KDE method but not with the RDC and NN methods, which rely on inherently random operations and/or local optimization mechanisms, such as random sinusoidal functions and Stochastic Gradient Descent. Figure 3.6 reports the impact of algorithmic stochasticity across three distinct HGR computation techniques, where the blue dashed lines represent single runs while the black solid line is the average. As we can observe, our method produces consistent results for each of the 30 seeds tested, whereas the outcomes of HGR-NN exhibit strong fluctuations while those of RDC exhibit few irregular peaks along with constant minor fluctuations. We reinforce that non-determinism can be a significant disadvantage in real-world applications, as it may either lead to confusion among decision-makers or require multiple evaluations to achieve a reliable measurement, thus impacting the computational complexity.

3.6 Empirical Evaluation

In this section, we will compare our method with the alternative approaches mentioned above in order to highlight its advantages and shortcomings when used to measure correlations. For these experiments, we will use synthetic data generated from known deterministic functions, optionally complemented with additive Gaussian noise. We denote our methods as HGR-KB and HGR-SK, with their respective hyperparameters set to $h = k = 5$ and $d = 5$ unless otherwise indicated. The comparative baselines include the adversarial technique (HGR-NN), the kernel density approach (HGR-KDE), the Randomized Dependence Coefficient (RDC), and the linear Pearson's correlation (PEARS). Further specifics regarding the implementation of both our approach and the baseline methods are available in Appendix A.4, while details about our hardware and software setup can be found in Appendix A.1.

3.6.1 Synthetic Data Generation

All datasets were synthetically generated based on a deterministic function f. Whenever the relationship is a function of one variable, we select 1001 equally spaced points from the interval $[-1, 1]$ and assign them to the independent variable. Subsequently, we determine the respective values of the dependent variable according to f. We consider six polynomial relationships along with five other non-linear functional forms. The polynomial functions include $y = x$, $y = x^2$, $y = x^3$, $x = y^2$, $x = y^3$, and $x^2 + y^2 = 1$ – this last one is also referred to as the *circular* relationship. In terms of the others, instead, we test $y = \max(0, x)$, $y = \text{sign}(x)$, $y = \tanh(x)$, $y = \sin(x)$, and $y = \sin(x^2)$.

In all scenarios, noise is drawn from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$ and added *solely* to the dependent variable after adjusting it proportionally to its standard deviation in order to ensure consistent results across datasets. For example, for the relationship $f : x \mapsto x^2$, we compute $y = f(x) + \sigma(f(x)) \cdot \mathcal{N}(0, \sigma^2)$, where $\sigma(\cdot)$ represents the standard deviation operator and σ is the noise level shown in Figures 3.7 and 3.8. We emphasize that, in the relationships $x = y^2$ and $x = y^3$, the independent variable is y, thus the noise is added to x. The only exception to

this process regards the circular relationship, where the variables are co-dependent. More specifically, we draw 501 samples of x from the standard interval $[-1, 1]$, after which we duplicate the sample size to 1002 data points. The vector y is then calculated as $y = \pm\sqrt{1 - x^2}$, applying a positive sign to the first half of the dataset and a negative sign to its duplicated counterpart. Given the absence of functional dependency between the two variables, proportional noise is incorporated into *both* variables. We assert that this distinct method of managing noise is what contributes more to the faster decline in computed correlation compared to other functions.

3.6.2 Measurement Experiments

As an initial experiment, we compute correlations using each discussed indicator, along with an ORACLE approach representing the Pearson correlation determined using optimal copula transformations – e.g., $f(a) = a^2$ and $g(b) = -b^2$ for circular data. For each deterministic relationship, we test sixteen distinct noise levels σ, ranging linearly from 0.0 to 3.0 in increments of 0.2. Next, we create ten different datasets by sampling the noise vector with a respective noise seed `ns`, and we perform the evaluation ten times using different algorithm seeds `as` to account for stochasticity in both the method and the data. The total number of runs is $123200 = 6 + 1$ (indicators) $\times$ 11 (functions) $\times$ 16 (noise levels) $\times$ 10 (noise seeds) $\times$ 10 (algorithm seeds).

The results are presented in Figure 3.7. The shaded bands around each line plot represent the standard deviation of the outcomes, stemming from both randomness in noise sampling and, only for HGR-NN and RDC, stochasticity in the solving process. Average execution times across all datasets, noises, and seeds are reported in the upper-left subplot, accompanied by a bar indicating the standard deviation.

The first thing we can observe is that HGR-SK is the fastest approach, although it fails to provide a good estimate of the correlation for certain relationships due to its inherent limitations to functional dependencies, especially in the circular dataset as it features a strong non-linear co-dependency of the variables. Concerning RDC, its results are comparable to HGR-KB and HGR-NN, with execution times that are one to two orders of magnitude lower. However, as previously shown, this approach exhibits significantly higher variability due to the presence of random kernels. In

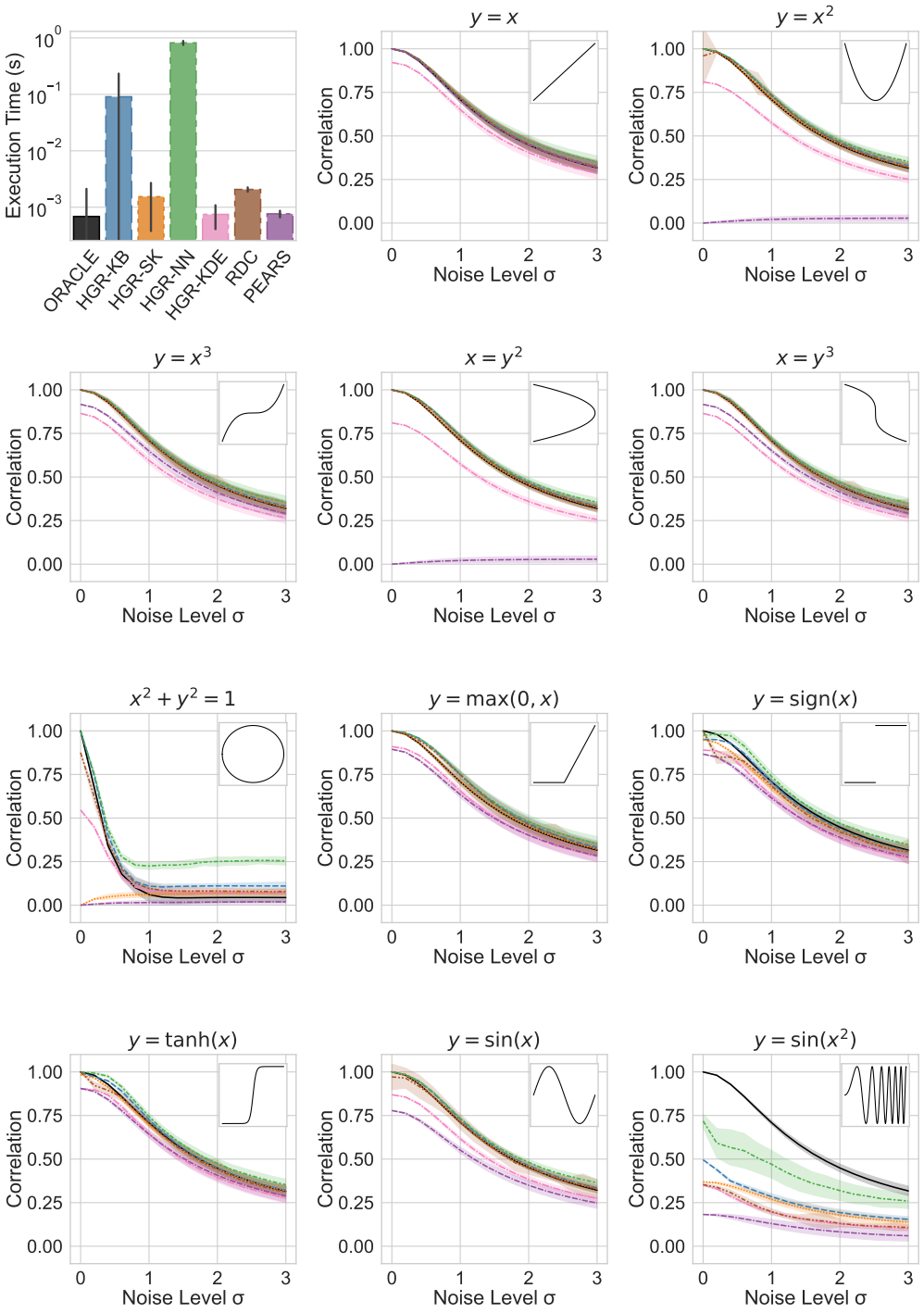


Figure 3.7: Execution times and correlations calculated using various HGR indicators across different deterministic relationships.

conclusion, we observe that HGR-KDE consistently yields lower correlation values even without noise, while HGR-NN, on the contrary, tends to generate higher estimates, sometimes even outperforming the ORACLE solution as in the case of circular relationships. We suggest that this is due to overfitting in the adversarial training procedure and support our hypothesis with additional experiments.

3.6.3 Test Distribution Experiments

To further examine the impact of overfitting in HGR-NN, we conduct an additional experiment using test distributions. Specifically, we initially collect the learned copula transformations from previous "training" runs and later create nine unique "test" datasets by sampling points from the same relationship with different noise seeds within the range of $\{0, \ldots, 9\}$, excluding the `ns` seed used to generate the "training" data. Finally, we calculate the correlation on these "test" datasets by applying the gathered copula transformations without undergoing further training. During this process, we exclude HGR-KDE and RDC as they either lack accessible transformations or do not allow easy access, thus obtaining a total of $792000 = 4 + 1$ (indicators) $\times\, 11$ (functions) $\times\, 16$ (noise levels) $\times\, 10$ (noise seeds) $\times$ 10 (algorithm seeds) $\times\, 9$ (test seeds) test cases.

The results of this experiment are reported in Figure 3.8. Its goal is to emphasize that the ability of HGR-NN to capture higher correlations in training data is likely a consequence of overfitting, since the outcomes on test distributions are similar to the other indicators, with the exception of $y = \sin(x^2)$ where it still provides a better estimate although in a smaller proportion with respect to "training" data.

To further support this claim, Figure 3.9 shows that the mappings produced by HGR-NN are significantly more unstable than those generated by our kernel-based methods. More specifically, we consider the circular dependency with noise $\sigma = 1.0$ and examine the copula transformations generated by: (i) our method HGR-KB with default hyperparameters $h = k = 5$, (ii) a variant of our method HGR-KB (2) with hyperparameters $h = k = 2$, and (iii) the adversarial method HGR-NN. Looking at the left (f) and right (g) plots, we observe how the neural transformations significantly overfit in certain regions, whereas our method tends to produce instability almost only at the borders. Additionally, since our method

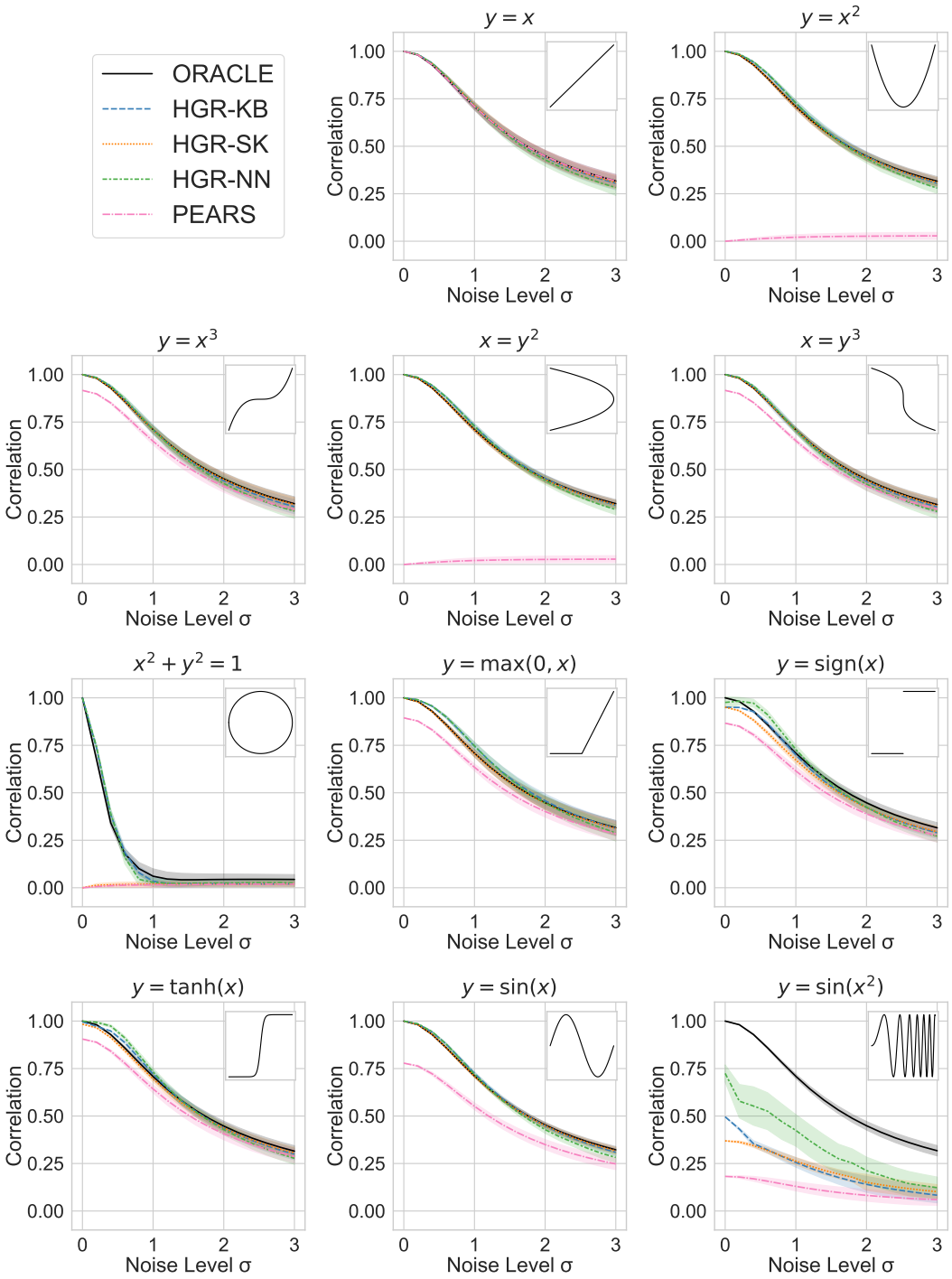


Figure 3.8: Test correlations computed for HGR indicators providing explicit access to the copula transformations.

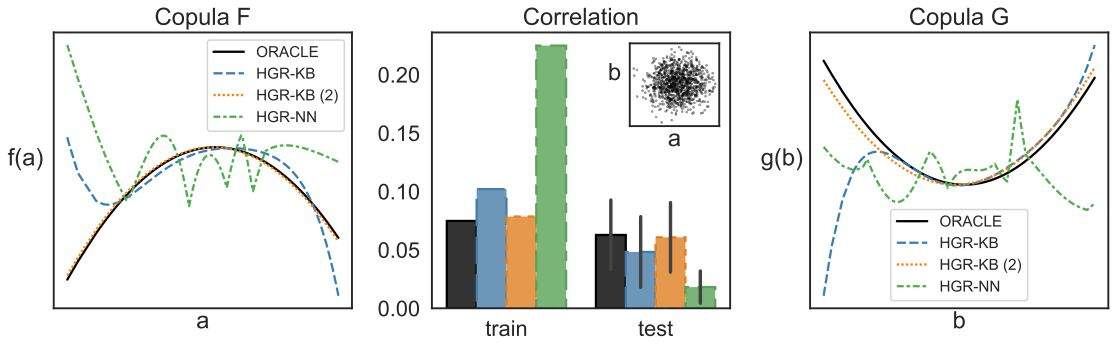
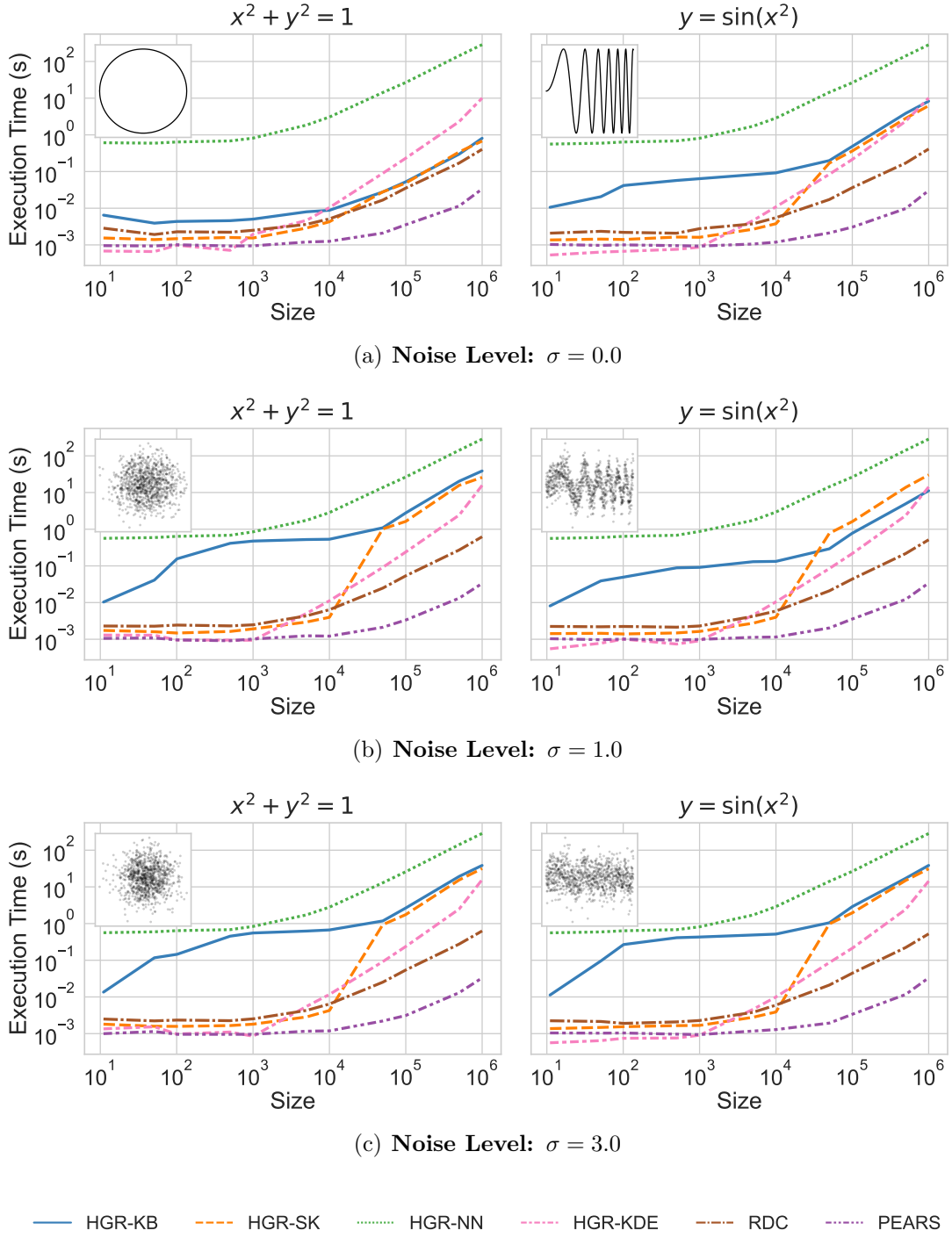


Figure 3.9: Kernel inspection of three HGR indicators on circular dataset.

allows to reduce the complexity of the transformations based on domain knowledge or experimental analysis, we could find that polynomial kernels of order 2 yield an even more stable result for this particular dataset, something that aligns in fact with the underlying deterministic relationship. The effects of this instability are also evident in the computed "test" correlation, shown in the central plot. Here, we can see that the seemingly optimal performance of HGR-NN in the training data does not translate to optimal performances in the test split, where all the methods provide similar results.

3.6.4 Scalability Experiments

As our last experiment, we examine the computational efficiency of each estimation algorithm. We have already assessed in Figure 3.3 the advantages of a least-square formulation compared to a global optimization strategy, which is accordingly reflected in the gap between the execution times of our kernel-based and single-kernel methods, as reported in Figure 3.7. Here, we aim at expanding such comparison by measuring the runtime of all the algorithms that we previously tested, respectively to a progressive increment in the input cardinalities. For each indicator, we compute the correlation on two distinct datasets across three noise levels – 0.0, 1.0, and 3.0, as shown in Figures 3.10(a) to 3.10(c). Each computation is repeated five times using different noise seeds, and timed over 11 data sizes ranging geometrically from 11 to 1000001, for a total of 6 (indicators) $\times$ 11 (sizes) $\times$ 2 (functions) $\times$ 3 (noise levels) $\times$ 5 (noise seeds) = 1980 runs.



(a) **Noise Level:** $\sigma = 0.0$

(b) **Noise Level:** $\sigma = 1.0$

(c) **Noise Level:** $\sigma = 3.0$

Figure 3.10: Average time required to estimate HGR using different algorithms in two synthetic datasets with growing cardinalities.

Figure 3.10 reports execution times averaged over the five seeds. We decided not to include variances in order to favor the interpretability of the plots, but these results are sufficient to support the partial observations obtained in Figures 3.3 and 3.7. HGR-NN confirms itself as the slowest approach, while the Randomized Dependence Coefficient is the quickest, following closely after Pearson's correlation straightforward calculation. Notably, HGR-KDE is the most affected by the growth in data size, becoming slower than HGR-KB in certain scenarios, but there is a general tendency of all the baselines to start demanding increased computational resources between 1000 and 10000 samples. Among all, our two methods are the least regular regarding execution times; this can be attributed to their reliance on optimization processes, which are highly sensitive to the structures of the input data. Specifically, since HGR-KB is based on iterative global optimization routines, it is able to converge faster when the relationship is directly representable by the kernel and relatively unaffected by noise. This also explains the greater variance in the times reported in Figure 3.7 for HGR-KB and HGR-SK, which is a consequence of their ability to respond more rapidly in simpler situations.

3.7 Final Discussion

In this chapter, we introduced a novel methodology for calculating the Hirschfeld-Gebelein-Rényi (HGR) correlation coefficient using two polynomial kernels as copula transformations. In addition to that, we also presented a less expressive but significantly faster approximation that is fully-differentiable and relies on the more established and well-studied least-square problem. Our technique offers distinct advantages in terms of robustness, explainability, and determinism, making it a more viable choice for real-world scenarios compared to existing techniques. We validated our claims using synthetic data generated from known deterministic functions, which demonstrated these benefits and highlighted the strengths of our approach. The findings validated our theoretical analysis of the indicator's properties, demonstrating its enhanced robustness against under- and overfitting. Additionally, it proved that our approximation is significantly more efficient, being two orders of magnitude faster than the most competitive baseline.

Chapter 4

Fairness with Continuous Protected Attributes

In this chapter, we will explore the application of fairness measures in learning tasks involving continuous protected attributes, where all traditional indicators are inapplicable. We start by introducing in Section 4.1 the use of HGR as a fairness metric, a concept that has already been examined in the literature, and conduct an empirical evaluation to show the advantages provided by our two implementations, HGR-KB and HGR-SK. Subsequently, in Section 4.2, we outline some limitations that may render HGR unsuitable as a fairness metric in certain contexts. We then propose a remedy in Section 4.3, presenting a novel indicator named Generalized Disparate Impact (GeDI) which extends the concept of Disparate Impact to continuous attributes, accompanied by our experimental analysis in Section 4.4 where this indicator is applied as a fairness constraint. In Section 4.5, we demonstrate a practical application of GeDI within an algorithm aimed at ensuring long-term fairness in ranking applications, and eventually conclude this chapter with a discussion about similarities and differences between GeDI and HGR in Section 4.6 which culminates into the definition of a unified formulation applicable to all non-linear correlation indicators. As for the previous chapter, all the proofs, results, and discussions reported here are adapted from [Giuliani et al., 2023], [Giuliani et al., 2024], and from another research paper currently under review at a prestigious Artificial Intelligence journal. Some content has been altered or

expanded to guarantee a more straightforward exposition in this thesis, but this does not modify the conclusions we obtained.

4.1 HGR as a Fairness Measure

As highlighted in Section 3.2, [Mary et al., 2019] was the first to propose the introduction of HGR as a measure of fairness, particularly with their estimation method HGR-KDE. Further advances in this domain were made by [Grari et al., 2020], who suggested the HGR-NN indicator for a similar purpose. In our work detailed in Chapter 3, our focus was restricted to evaluating correlation using our methods HGR-KB and HGR-SK and benchmarking them against the existing standards. Nevertheless, we did not explore neither theoretical nor experimental implications of employing them either as constraints or as fairness metrics.

Our next experiment aims to fill this gap by examining the effectiveness of our HGR estimation techniques in addressing unfair penalization under continuous protected attributes. For this analysis, we replicate the experimental procedure of both [Mary et al., 2019] and [Grari et al., 2020], thus performing evaluations on three widely-recognized benchmark datasets, namely the US 2015 Census (*Census*), Communities & Crimes (*Communities*), and Adult Census Income (*Adult*). We restrict our fairness regularizers to the methods HGR-KB, HGR-SK, and HGR-NN, and adopt the Lagrangian dual approach from [Fioretto et al., 2021] to ensure that the measured correlation remains below a custom threshold τ which we define for each dataset. More details on the datasets, on the learning algorithms we use, and on our implementation can be found in Appendix A.

4.1.1 Identifying Protected Attributes

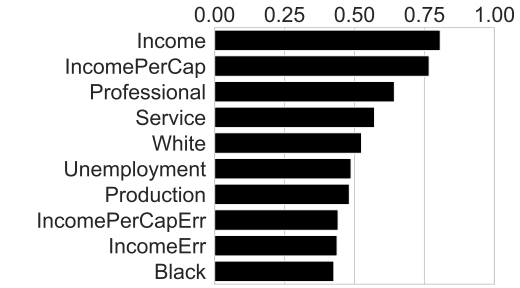
Before starting our learning experiments, we identify one continuous protected attribute in each of our benchmark datasets. To achieve this, we use HGR-KB to measure the level of correlation between the target variable y and all the input variables in the dataset. As a result, we pinpoint the protected attribute z as the most correlated continuous feature that can be considered a potential source of unfairness. For the *Census* dataset, this attribute is `Income`; for *Communities*, it

is `pctWhite`; and for *Adult*, it is `age`.

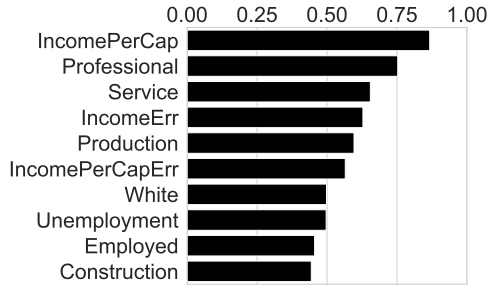
The left column of Figure 4.1 reports the outcomes of this feature importance step. Specifically, the x-axis denotes the level of correlation, while the y-axis lists the feature names. The selected feature is not always at the top either because it does not represent any sensitive information or because it is not continuous. For example, the attribute `pctKids2Par` in the *Communities* dataset denotes the proportion of children in family housing with two parents, which we did not classify as sensitive information. Conversely, the `Married-civ-spouse` attribute in the *Adult* dataset is binary, hence it was deemed not useful for our experimental analysis, despite its potential for discriminatory implications. Nonetheless, we can notice that measuring correlations can suggest potential unfair causes as expected, since income, ethnicity, and age – together with gender, which also appears in the analysis of the *Adult* dataset – are typically identified as the primary sources of discrimination in these benchmarks.

Subsequently, we repeat the procedure to assess the correlation between the protected attribute and each remaining input variable, again using HGR-KB as a metric. Our goal is to identify a surrogate protected attribute s that: (i) shows a high correlation with the continuous protected attribute z, (ii) represents different sensitive information, and (iii) is binary. This strategy allows us to pair the continuous protected attribute with a binary surrogate to observe how constraints on the former affect the latter. More specifically, since HGR is not widely studied as a fairness metric, our objective is to investigate whether the decrease in the correlation between the target y and the protected attribute z is mirrored in the surrogate s. The surrogate is deliberately chosen to be binary to enable the use of a more established fairness metric such as the Disparate Impact Discrimination Index (DIDI). The results of this second feature importance analysis are illustrated in the right column of Figure 4.1. The identified surrogate attributes are `Unemployment` for the *Census* dataset[1], `race` for the *Communities* one, and `Never-married` for *Adult*. Once again, the chosen surrogates might not be the most correlated since

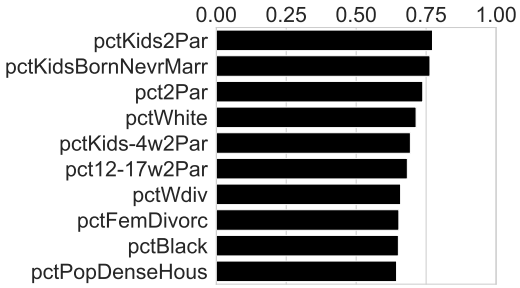
[1]The *Census* dataset contains continuous attributes only, so we first identified `Unemployment` as the most correlated surrogate and then binarized it using its mean value as a threshold. The binarization process results in some loss of information, which explains why the `White` feature, though unselected, ranks slightly higher.



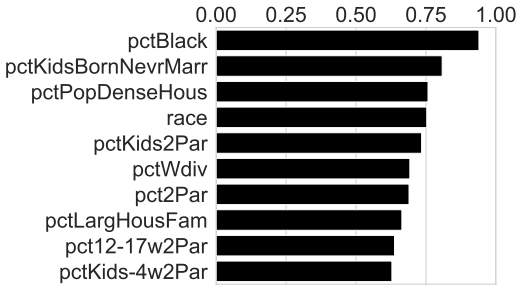
(a) *Census* – `ChildPoverty`



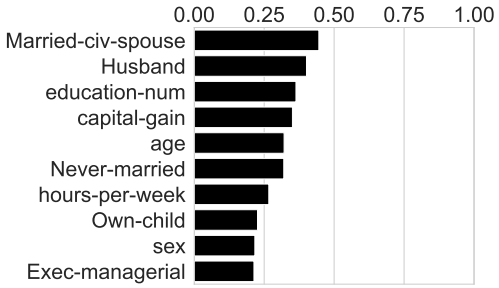
(b) *Census* – `Income`



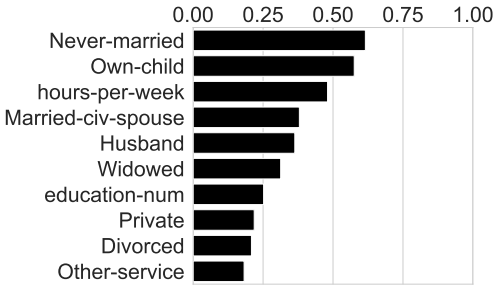
(c) *Communities* – `violentPerPop`



(d) *Communities* – `pctWhite`



(e) *Adult* – `income`



(f) *Adult* – `age`

Figure 4.1: Top 10 highly correlated features with respect to output target (left) and continuous protected attribute (right) computed on the three benchmark datasets using HGR-KB.

they do not respect our experimental criteria; however, it is evident that both the continuous protected attribute and the surrogate are interlinked not only from a statistical but also from a semantic viewpoint, as they represent similar categories and discriminated features.

4.1.2 Training Details

With our salient features selected, we can now proceed to define our training task. In particular, we want to solve the following learning problem:

$$\arg\min_{\theta} \mathcal{L}(y, \hat{y}(\theta)) \qquad \text{s.t.} \qquad \text{HGR}(z, \hat{y}(\theta)) \leq \tau, \quad \tau \in \mathbb{R}^{+} \tag{4.1}$$

where $\hat{y}(\theta)$ denotes the model's predictions, y stands for the ground truths, z is the continuous protected attribute, $\mathcal{L}$ represents the original task loss, and τ is a non-negative threshold which serves as an upper bound for our desired level of correlation. Specifically, we employ Mean-Squared Error (MSE) as loss for regression tasks and Binary Crossentropy (BCE) for classification tasks. Furthermore, we link each dataset to a corresponding score function, i.e., the R^2 score and the Area Under Curve (AUC), respectively for regression and classification tasks.

Theoretically, one can address Equation (4.1) employing either declarative projection methods or loss regularization techniques. However, the constraints required for computing HGR-KB and HGR-SK make it infeasible to process the computation in a reasonable time frame, while alternative methods for HGR estimation lack any declarative structure. For this reason, our experiments will be conducted using a custom training loss defined as:

$$\mathcal{L}(y, \hat{y}(\theta)) + \lambda \cdot \max\{0, \text{HGR}(z, \hat{y}(\theta)) - \tau\} \tag{4.2}$$

In order to enhance efficiency and offer a more fair comparison of results, we decided to take advantage of the Lagrangian dual framework proposed in [Fioretto et al., 2021] to satisfy the regularization term included in Equation (4.2), as it dynamically adjusts the multiplier $\lambda \in \mathbb{R}^{+}$ through a gradient ascent step. This removes the need for a separate tuning phase for λ, thus allowing

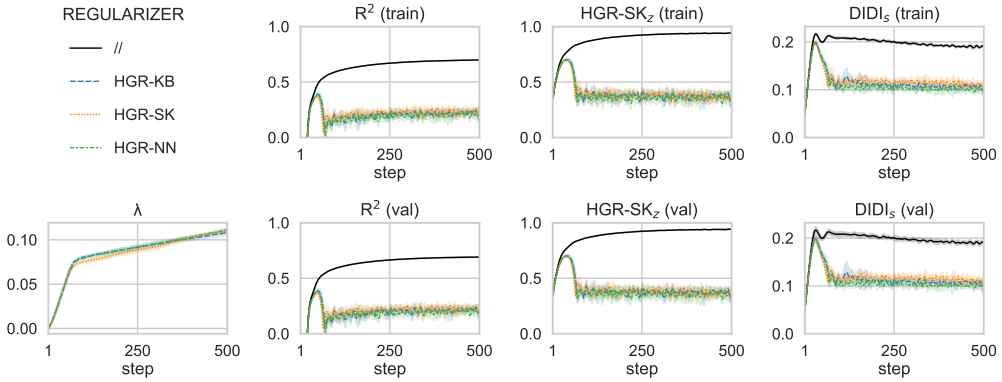
Dataset	*Census*	*Communities*	*Adult*
Output Target	`ChildPoverty`	`violentPerPop`	`income`
Protected Attribute	`Income`	`pctWhite`	`age`
Surrogate Attribute	`Unemployement`	`race`	`Never-married`
Loss Function	MSE	MSE	BCE
Score Function	R^2	R^2	AUC
Hidden Units	[32]	$[256, 256]$	$[32, 32]$
Batch Size	2048	Full-Batch	2048
Steps	500	500	500
τ	0.4	0.3	0.2

Table 4.1: Description of salient input and output features, along with network hyperparameters and penalty threshold for the three benchmarks.

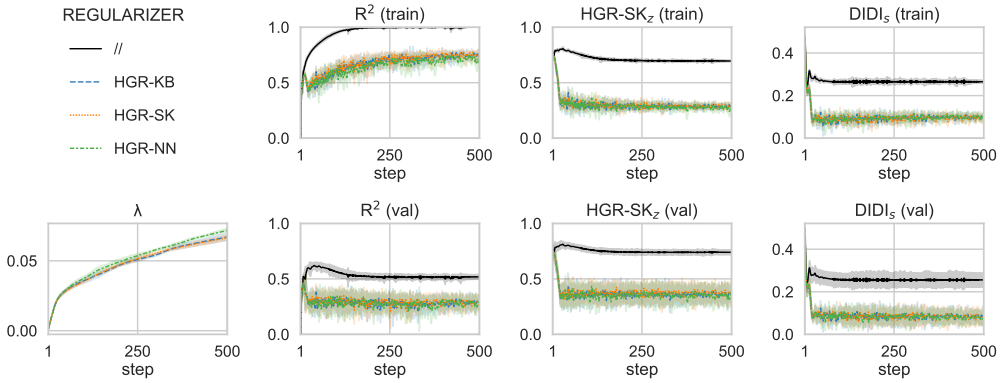
our training mechanism to penalize any correlation that exceeds a predefined threshold τ without the need to specify further hyperparameters – for additional details, see Appendix A.4.2. We manually select a custom threshold for each benchmark, based on the initial correlation levels between y and z as reported in Figure 4.1. Having different thresholds also serves the dual purpose of testing different experimental setups and emulating real-world scenarios where fairness constraints might need to be adjusted based on external requirements or conflicts with predictive accuracy. Finally, concerning our neural model architectures, we undergo a hyperparameter tuning phase which is further detailed in Appendix A.3. We report the optimal configuration of the learning models for each dataset in Table 4.1, paired with a summary of the key input and output features along with the selected thresholds τ.

4.1.3 Experimental Results

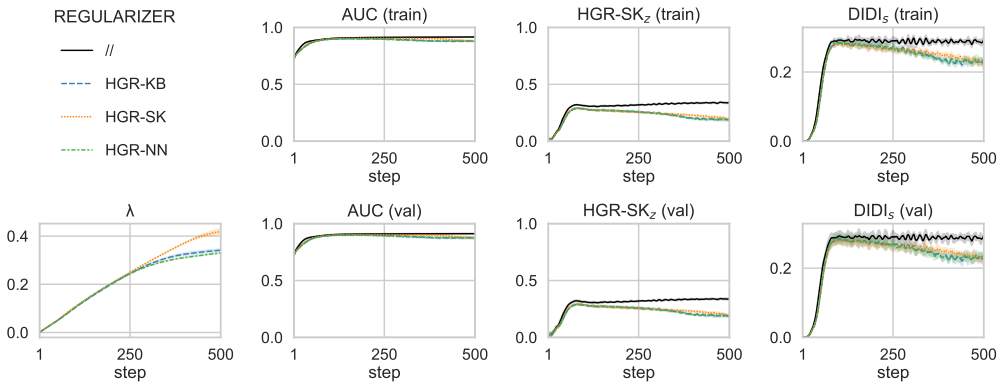
We conduct our experiments by splitting each dataset into 5 folds. For each of them, we train a neural network with the configuration discussed above and use HGR-KB, HGR-SK, and HGR-NN methods to calculate the correlation between the predictions $\hat{y}(\theta)$ and the protected attribute z as described in Equation (4.2). Furthermore, we train an unconstrained model, denoted as //, where we ignore the



(a) *Census* – $\tau = 0.4$



(b) *Communities* – $\tau = 0.3$



(c) *Adult* – $\tau = 0.2$

Figure 4.2: Training histories on all the benchmark datasets for neural networks with HGR-based loss regularizers.

Regularizer	Score ($\times 10^2$)		Constraint$_z$($\times 10^2$)		DIDI$_s$($\times 10^2$)		Time (s)
	train	val	train	val	train	val	
CENSUS ($\tau = 40$)							
//	**70 ± 00**	**69 ± 00**	//	//	**19 ± 00**	**19 ± 00**	**32 ± 00**
HGR-KB	21 ± 03	20 ± 02	36 ± 03	36 ± 03	10 ± 01	10 ± 01	70 ± 03
HGR-SK	23 ± 02	22 ± 01	36 ± 02	36 ± 02	11 ± 00	11 ± 01	38 ± 00
HGR-NN	19 ± 04	19 ± 04	35 ± 05	35 ± 04	10 ± 01	10 ± 01	88 ± 00
COMMUNITIES ($\tau = 30$)							
//	**100 ± 00**	**52 ± 02**	//	//	**26 ± 01**	**26 ± 03**	**06 ± 00**
HGR-KB	74 ± 04	27 ± 05	28 ± 03	37 ± 07	10 ± 01	08 ± 02	39 ± 04
HGR-SK	74 ± 05	27 ± 06	29 ± 02	36 ± 10	10 ± 02	08 ± 04	12 ± 00
HGR-NN	72 ± 05	28 ± 07	30 ± 03	46 ± 07	10 ± 02	08 ± 04	60 ± 01
ADULT ($\tau = 20$)							
//	**92 ± 00**	**91 ± 00**	//	//	**29 ± 01**	**29 ± 01**	**15 ± 00**
HGR-KB	88 ± 00	88 ± 01	19 ± 01	19 ± 01	23 ± 01	23 ± 01	57 ± 01
HGR-SK	88 ± 00	87 ± 01	20 ± 01	20 ± 01	23 ± 01	23 ± 01	22 ± 00
HGR-NN	88 ± 00	88 ± 00	19 ± 00	20 ± 00	23 ± 00	23 ± 01	73 ± 01

Table 4.2: Results of HGR learning experiments conducted on the three benchmarks.

regularizer result, thus addressing only the task loss.

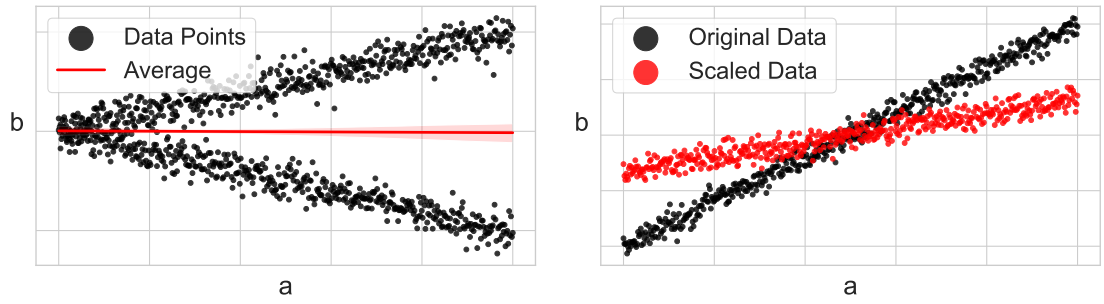
The entire training history is depicted in Figure 4.2. For each combination of model and regularizer, we present the score at each training step alongside the correlation computed on the full split. We use HGR-SK$(z, \hat{y}(\theta); 5)$ as the correlation measure for every combination, since calculating both HGR-KB and HGR-NN from scratch and without using mini-batches for each training step would introduce a strong computational overhead. Furthermore, we report the DIDI score between the predictions $\hat{y}(\theta)$ and the surrogate s – we use the subscripts z and s to underline that the metric is computed with respect to the continuous protected attribute z; or with respect to the binary surrogate s. Moreover, we show the evolution of the multiplier λ as it gets automatically calibrated throughout the training steps. As we can see, its increment works as expected until the constraint is adequately satisfied, thus validating our algorithm choice, especially given that the thresholds are mostly satisfied across all training instances.

In addition to the training history, Table 4.2 shows the final results of our training procedures. Here, under the "Constraint$_z$" column, we report the correlation measure as calculated by the same indicator tested; that is, if the regularizer is based on the semantics of HGR-KB, then the correlation for the entire batch of

predictions is also calculated using HGR-KB. Again, the z subscript highlights the fact that we are computing the correlation with respect to the continuous protected attribute. We also underline that, since constraint satisfaction is assessed using the selected regularizer, a direct comparison between methods is not possible. In fact, to have a such comparison, we would need an oracle that can provide the true correlation between the continuous protected input and the output target and which is not available due to the limitations of HGR.

Nonetheless, we can assess the extent to which each indicator offers informative gradient information by examining the final constraint value and checking if it respects our threshold. We report the respective threshold multiplied by 10^2 for easier comparison alongside each dataset, and we can see that on average the constraints are always satisfied in the training data. The values that are closer to threshold level occur in the *Adult* dataset, which could be due to either the strictest threshold assigned to the task or to potential inherent issues with classification tasks. Indeed, when computing inference-time correlation, we can easily transition from predicted probabilities $\hat{y} \in [0, 1]$ to predicted classes $\hat{y} \in \{0, 1\}$; however, this is not feasible during training, as binarizing continuous values by rounding results in null gradients, rendering the regularization term ineffective. We emphasize that this problem is intrinsic to the task and the learning methodology itself, and will thus be present in any HGR-based indicator. However, a reduction in unfairness is consistently observed in the validation data in all cases, with minimal increases due to generalization issues.

We also report the R^2 or AUC values in the "Score" column, while the "DIDI_s" column contains the absolute value of the disparate impact calculated on the surrogate binary attribute s. For better visualization, values are multiplied by 100, and the first row of each dataset is highlighted for comparison, as it represents the unconstrained model. We can notice that the accuracies are consistent across all tested regularizers, with HGR-NN slightly underperforming, possibly due to its tendency to overestimate the true correlation, whereas HGR-KB shows greater stability, as indicated by the lowest standard deviations in the measured correlations across both training and validation data. Finally, the "Time" column displays the overall training time in seconds, highlighting HGR-SK as the fastest among the



(a) Non-functional Dependencies

(b) Invariance to Scale

Figure 4.3: Two examples of potential limitations of HGR when used as a fairness measure.

regularizers, with minimal computational overhead introduced in comparison to the unconstrained model.

On a conclusive note, we emphasize that the reduction in correlation between the continuous protected attribute z and the predictions $\hat{y}(\theta)$ consistently results in lower unfairness with respect to the binary protected surrogate, as measured by the Disparate Impact Discrimination Index. This happens for all the tested regularizers, thereby supporting the application of HGR as a fairness metric, as demonstrated by its link to a more established indicator.

4.2 Limitation of HGR Semantics

The HGR indicator has several notable properties and, as demonstrated in the preceding section, it can serve as a well-defined measure of fairness. However, there could be cases where its inherent semantics entails some undesired characteristics. Before presenting our solution, we highlight two such instances.

4.2.1 Support for Non-Functional Dependencies

Firstly, the use of unrestricted copula transformations allows HGR to capture very broad forms of dependency between a and b. This includes scenarios like the one shown in Figure 4.3(a), where divergent target values can be intercepted by the great expressivity of HGR. Nonetheless, there are instances where discrimination

arises only when the *expected* value of the target is altered, meaning it relates to the strength of the functional dependency $\mathbb{E}[b \mid a]$. For example, in the case depicted, the divergence might be caused by a third confounding variable correlated with a. In such a case, the confounder might not pose an ethical concern, but the HGR metric would not be able to exclusively measure the functional dependency.

4.2.2 Invariance to Scale

Secondly, the HGR indicator satisfies all the Rényi properties [Rényi, 1959] as it uses the Pearson's correlation coefficient. However, this entails the inability of the method to consider scale effects on fairness. For instance, assume that the target values are linearly correlated with the continuous protected attributes. In some practical scenarios, an affine transformation applied to the targets could reduce discrimination in a similar way as illustrated in Figure 4.3(b). Still, the HGR indicator fails to capture this effect due to the scale-invariance of Pearson's correlation, hence reporting an identical level of unfairness.

4.3 Generalized Disparate Impact

Motivated by the limitations mentioned above, this section introduces a new fairness indicator along with an explanation of its semantics. The objective here is to enhance, rather than replace, the HGR approach by offering additional options for continuous protected attributes. To achieve this, we designed the new indicator, named Generalized Disparate Impact, so that it behaves in a manner that is similar to and, under specific circumstances, identical to the Disparate Impact Discrimination Index (DIDI) [Aghaei et al., 2019].

4.3.1 Definition of GeDI

Let us recall the definition of HGR using polynomial models as for Equation (3.14):

$$\arg\min_{\alpha,\beta,r} \left\| \frac{\mathbf{P}_a^h \cdot \alpha - \mu(\mathbf{P}_a^h \cdot \alpha)}{\sigma(\mathbf{P}_a^h \cdot \alpha)^2} \cdot r \; - \frac{\mathbf{P}_b^k \cdot \beta - \mu(\mathbf{P}_a^k \cdot \beta)}{\sigma(\mathbf{P}_a^k \cdot \beta)^2} \right\|_2^2 \tag{4.3}$$

Its main insight is that the HGR computation can be seen as a process akin to least-square fitting. Building on this concept, we formulate the GeDI indicator with two significant modifications aimed at addressing the limitations discussed in Section 4.2. Firstly, we remove the copula transformation on the b variable – i.e., the g function. This ensures that our indicators can exclusively measure the strength of the functional dependency $\mathbb{E}[b \mid a]$, thus considering situations like the one depicted in Figure 4.3(a) completely fair. Secondly, we eliminate the standardization terms in the denominators. This prevents the indicators from fulfilling certain Rényi properties, which are applicable only to scale-invariant measures, but allows to effectively yield lower values of unfairness when the target data is scaled as in Figure 4.3(b). The optimization problem presented in Equation (4.3) is therefore reformulated as follows:

$$\arg\min_{\alpha,r} \left\| (\mathbf{P}_a^h \cdot \alpha - \mu(\mathbf{P}_a^h \cdot \alpha)) \cdot r - (y - \mu(y)) \right\|_2^2 \quad \text{s.t.} \quad \sigma(\mathbf{P}_a^h \cdot \alpha) = \sigma(a) \tag{4.4}$$

where the constraint on $\sigma(\mathbf{P} \cdot \alpha)$ is necessary to prevent unbounded solutions that could arise due to the scale-independence of GeDI.

Eventually, we define the indicator as the absolute value of the solution r^*, and demonstrate in Appendix C.1 that it can be expressed in the following closed form:

$$\text{GeDI}(a, b; h) = |r^*| = \left| \frac{\text{cov}(\mathbf{P}_a^h \cdot \alpha^*, b)}{\text{var}(\mathbf{P}_a^h \cdot \alpha^*)} \right| \tag{4.5}$$

with α^* being the optimal coefficient obtained from Equation (4.4).

We recall that the GeDI indicator is not intended to replace HGR, but rather to enhance and expand the existing range of options. In doing so, our aim is to correlate its value with the established DIDI metric; for this reason, we demonstrate in Appendix C.2 that, for any binary input vector $a \in \{0, 1\}^n$, the DIDI for regression problems can be calculated as follows:

$$\text{DIDI}(a, b) = \left| \frac{\text{cov}(a, b)}{\text{var}(a)} \right| \tag{4.6}$$

and that a proportional value, multiplied by a factor of 2, is yielded in binary classification scenarios where $b \in \{0, 1\}^n$ as well. In this regard, the constraint on

$\sigma(\mathbf{P} \cdot \alpha)$ is specifically designed to match $\sigma(a)$ in order to ensure that GeDI yields the exact DIDI value when handling binary vectors. In fact, when using a degree $h = 1$, our indicator reduces to:

$$\text{GeDI}(a, b; 1) = \left| \frac{\text{cov}(\alpha^* \cdot a, b)}{\text{var}(\alpha^* \cdot a)} \right| = \left| \frac{\text{cov}(a, b)}{\text{var}(a)} \right| = \text{DIDI}(a, b) \tag{4.7}$$

where $\alpha^* = 1$ is implied by the constraint $\sigma(\alpha^* \cdot a) = \sigma(a)$. Moreover, since the polynomial expansion of a binary value merely equals the value itself, this result can be extended to any degree $h \in \mathbb{N}$, proving that the GeDI indicator can be viewed as a generalization of the (binary) DIDI to continuous protected attributes.

Reworked Formulation

Before detailing the procedure to calculate the GeDI indicator, it is worth highlighting that the definition provided in Equation (4.4) slightly differs from the original one proposed in [Giuliani et al., 2023]. Specifically, in that version we imposed a constraint $\|\alpha\|_1 = 1$ instead of the one on the standard deviation $\sigma(\mathbf{P}_a^h \cdot \alpha)$. Both conditions allow to equate the DIDI value for binary vectors, as both require the optimal value $\tilde{\alpha}^*$ to be 1 in case of unitary degree. Nonetheless, while the constraint on the norm-1 appears somewhat arbitrary and implementation-specific, that on the standard deviation grants three important advantages:

1. It can be seen as a mechanism for adjusting the distribution of data points without altering their sparsity. Although this aspect has no effect in HGR due to the irrelevance of the scale, limiting the sparsity of projected data is crucial in GeDI to prevent solutions from being unbounded. A constraint on the standard deviation therefore offers a clearer and more comprehensible approach to addressing this need.

2. It is applied to the data rather than the model parameters, hence having the potential to be generalized to other learning approaches. For instance, using the constraint on norm-1 it would not be possible to apply the GeDI semantics while using a neural-based copula transformation, while no problems would arise in case of the constraint on the standard deviation.

3. It allows to replace the denominator in Equation (4.12) with the constant term $\text{var}(a)$. Consequently, the indicator would grow solely based on the numerator $\text{cov}(\mathbf{P}_a^h \cdot \alpha^*, b)$, thereby assuring monotonicity with respect to the degree h, akin to HGR. With the norm-1 constraint, this property is compromised, as the denominator may vary and potentially make the optimal solution for $\text{GeDI}(a, b; h')$ lower than that for $\text{GeDI}(a, b; h)$ even with $h' > h$.

For a deeper examination of why the reworked formulation enables to extend the semantics of GeDI to other computational algorithm, and how it allows to correlate the values of HGR and GeDI up to a data-dependent scaling factor, we refer the reader to Section 4.6.2.

4.3.2 Computation of GeDI

Here, we discuss how to derive an unconstrained version of the GeDI formulation using substitutions analogous to those applied in Section 3.3.2. More specifically, starting from the following substitutions:

$$\widetilde{\mathbf{P}}_a^h = \mathbf{P}_a^h - \mu(\mathbf{P}_a^h) \qquad \widetilde{y} = y - \mu(y) \qquad \widetilde{\alpha} = \alpha \cdot r \tag{4.8}$$

we can reduce Equation (4.4) to a classical minimal Least Squares formulation:

$$\arg\min_{\widetilde{\alpha}} \|\widetilde{\mathbf{P}}_a^h \cdot \widetilde{\alpha} - \widetilde{y}\|_2^2 \tag{4.9}$$

The substitutions involve no loss of generality, as we can pre-calculate the zero-centered polynomial kernel $\widetilde{\mathbf{P}}_a^h$ and vector $\widetilde{y}$, and the combination of r and α into a single variable admits a solution for any $\tilde{\alpha} \in \mathbb{R}^h$. Once the optimal vector of coefficients $\widetilde{\alpha}^*$ is obtained as the solution of an unconstrained least-square problem, we can rely on its definition from Equation (4.8) to derive the actual value of GeDI. Particularly, by multiplying both sides with $\mathbf{P}_a^d$ and calculating the standard deviation, we have:

$$\sigma(\mathbf{P}_a^h \cdot \widetilde{\alpha}^*) = \sigma(\mathbf{P}_a^h \cdot \alpha^* \cdot r^*) \tag{4.10}$$

which simplifies to:

$$\sigma(\mathbf{P}_a^h \cdot \widetilde{\alpha}^*) = r^* \cdot \sigma(\mathbf{P}_a^h \cdot \alpha^*) = r^* \cdot \sigma(a) \tag{4.11}$$

given the constraint on the standard deviation outlined in Equation (4.4).

Eventually, since we defined GeDI as $|r^*|$ in Equation (4.5), we can derive a closed-form expression as:

$$\text{GeDI}(a, b; h) = \frac{\sigma(\mathbf{P}_a^h \cdot \widetilde{\alpha}^*)}{\sigma(a)} \tag{4.12}$$

without the need of using the absolute value, as the standard deviation operator is inherently non-negative.

4.3.3 Enforcing GeDI

Beyond assessing fairness, GeDI indicators can be used to enforce constraints on a protected attribute z. In this framework, the purpose would be to adjust the predictions $\hat{y}$ either to steer the training of a machine learning model or to modify the data itself when using pre- or post-processing constraint enforcement algorithms. Similar to our discussion in Section 4.1.2, we strive to address the following problem:

$$\arg\min_{\hat{y}} \mathcal{L}(y, \hat{y}) \qquad \text{s.t.} \qquad \text{GeDI}(z, \hat{y}; h) \leq \tau, \quad \tau \in \mathbb{R}^+ \tag{4.13}$$

where, again, τ is a non-negative threshold that acts as an upper bound for the permitted level of unfairness; $\mathcal{L}$ represent the original task loss; and $\hat{y}$, y, and z denote the predicted outcomes, the vector of ground truths, and the continuous protected attribute, respectively.

Similar to the approach used for HGR, a straightforward solution to the problem involves the use of a Lagrangian term (i.e., a regularizer) expressed as:

$$\lambda \cdot \max\{0, \text{GeDI}(z, \hat{y}; h) - \tau\} \tag{4.14}$$

where $\lambda \in \mathbb{R}^+$ represent the weight of the regularizer. Using the value of GeDI as

specified in Equation (4.12), this expression can be recast as:

$$\lambda \cdot \max\{0, \frac{\sigma(\mathbf{P}_z^h \cdot \widetilde{\alpha}^*)}{\sigma(z)} - \tau\} \tag{4.15}$$

Nevertheless, differently from both HGR-KB and HGR-SK, the definition of the GeDI indicator exclusively contains linear constraints. Specifically, the least-square problem outlined in Equation (4.9) can be expressed using a set of piecewise linear relations as follows:

$$\widetilde{\mathbf{P}}_z^T \cdot \widetilde{\mathbf{P}}_z \cdot \widetilde{\alpha} = \widetilde{\mathbf{P}}_z^T \cdot \widetilde{y} \tag{4.16}$$

where we removed the superscripts h and $*$ from $\widetilde{\mathbf{P}}_z$ and $\widetilde{\alpha}$ and the hat from $\hat{y}$ to enhance readability. Furthermore, the constraint described in Equation (4.13) can be reformulated as:

$$\sigma(\mathbf{P}_z \cdot \widetilde{\alpha}) \leq \sigma(z) \cdot \tau \Rightarrow \text{var}(\mathbf{P}_z \cdot \widetilde{\alpha}) \leq \text{var}(z) \cdot \tau^2 \tag{4.17}$$

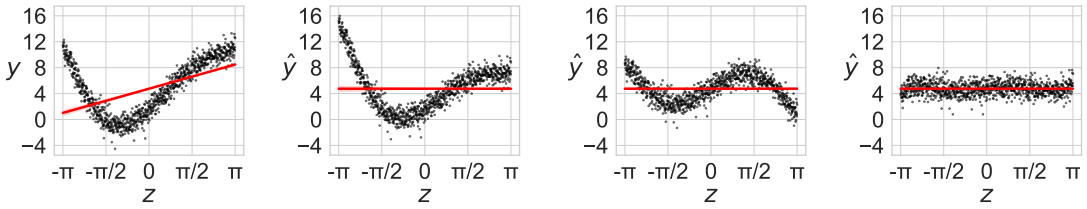
with both sides squared in order to eliminate the need for calculating the square root within the standard deviation operator.

By combining these two equations together, we arrive at:

$$\begin{gathered} \widetilde{\mathbf{P}}_z^T \cdot \widetilde{\mathbf{P}}_z \cdot \widetilde{\alpha} = \widetilde{\mathbf{P}}_z^T \cdot \widetilde{y} \\ \frac{1}{n} \cdot \widetilde{\alpha}^T \cdot \widetilde{\mathbf{P}}_z^T \cdot \widetilde{\mathbf{P}}_z \cdot \widetilde{\alpha} \leq \frac{1}{n} \cdot \widetilde{z}^T \cdot \widetilde{z} \cdot \tau^2 \end{gathered} \tag{4.18}$$

where we replaced $\text{var}(\mathbf{P}_z \cdot \widetilde{\alpha})$ and $\text{var}(z)$ with their respective dot-product expressions using zero-centered matrices/vectors. This problem involves a set of linear equalities featuring a positive semi-definite matrix $\widetilde{\mathbf{P}}_z^T \cdot \widetilde{\mathbf{P}}_z$ on their left-hand side, along with a single lower-than-equal constraint involving squared terms. Consequently, if paired with an objective function that is no more complex than quadratic, this formulation fits in the Quadratically Constrained Quadratic Program (QCQP) framework, allowing it to be solved in a potentially efficient manner.

As a demonstration of this, we implement an optimization problem that satisfies Equation (4.18) and yields the optimal projections $\hat{y}$ by minimizing the Mean Squared Error relative to the original targets y – i.e., we project the targets using



(a) No Constraint (b) GeDI$(z, \hat{y}; 1) = 0$ (c) GeDI$(z, \hat{y}; 2) = 0$ (d) GeDI$(z, \hat{y}; 3) = 0$

Figure 4.4: Example of GeDI enforcement using the declarative formulation with increasing degrees.

the Euclidean distance into a region of the space where these constraints are fulfilled. Figure 4.4 shows how the projections vary as we apply the constraint GeDI$(z, \hat{y}; h) \leq 0$ with increasing degrees, exhibiting how higher values of h respond to distinct dependencies and therefore perform different adjustments in order to remove unfairness. In particular, Figure 4.4(a) displays the original data sampled from $y = f(z) = 4\sin(z) + z^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1.5)$ and $a \sim \mathcal{U}(-\pi, \pi)$. This reveals a linear correlation, as no constraint is imposed, which is promptly removed in Figure 4.4(b) where we apply the constraint using a degree $h = 1$. Nonetheless, GeDI$(z, \hat{y}; 1)$ fails to capture most correlations due to the absence of linear terms, so it merely rotates the data to eliminate linear correlations without significantly altering the structure. In contrast, by employing a degree $h = 2$, the data in Figure 4.4(c) gets successfully deprived of the squared term in $f(z)$, retaining only the sinusoidal component. Finally, Figure 4.4(d) illustrates how the use of degree $h = 3$ effectively removes almost all dependencies, since the sinusoidal function can be approximated as $z - \frac{1}{6}z^3$ according to the Taylor expansion and therefore only $o(z^5)$ plus noise is left. Overall, together with the analysis of the monotonic behavior of the indicator, this serves as an additional pre-processing step to help in selecting a degree that avoids both overfitting and numerical issues.

4.3.4 Fine-Grained Constraint Formulation

Within our polynomial expansion model, each column of the kernel matrix captures different functional dependencies and is associated with a coefficient $\widetilde{\alpha}_i$ indicating the strength of that dependency. By setting a threshold $\tau = 0$ as shown in Figure 4.4,

all coefficients are naturally constrained to zero; nonetheless, using less restrictive thresholds enables the algorithm to redistribute dependencies between columns by altering the values of $\widetilde{\alpha}$, thereby meeting the overall constraint requirements.

Suppose, instead, that we would like to impose limitations on certain terms of the copula transformation. In practice, this involves replacing the aggregate constraint on GeDI with h separate constraints on the absolute values of the $\widetilde{\alpha}_i$ coefficients. Each constraint can be paired with a user-determined threshold $\tau_i \in \mathbb{R}^+$, so that:

$$|\widetilde{\alpha}_i| \leq \tau_i, \quad \forall i \in \{1, \ldots, h\} \tag{4.19}$$

By utilizing vector notation, with $\tau = \begin{bmatrix} \tau_1, & \ldots, & \tau_h \end{bmatrix} \in \mathbb{R}^{+h}$, we can modify Equation (4.18) to incorporate these fine-grained constraints as follows:

$$\begin{aligned} &\widetilde{\mathbf{P}}_z^T \cdot \widetilde{\mathbf{P}}_z \cdot \widetilde{\alpha} = \widetilde{\mathbf{P}}_z^T \cdot \widetilde{y} \\ &-\tau \leq \widetilde{\alpha} \leq \tau \end{aligned} \tag{4.20}$$

where the absolute value has been recast into a pair of linear inequalities.

Given the vector of thresholds τ, it is also possible to determine in advance the upper bound of $\text{GeDI}(z, \hat{y}; h)$. Specifically, we observe that:

$$\frac{\sigma(\mathbf{P}_z^h \cdot \widetilde{\alpha}^*)}{\sigma(z)} = \frac{\sigma(\sum_{i=1}^h \widetilde{\alpha}_i^* \cdot z^i)}{\sigma(z)} = \frac{\sum_{i=1}^h \sum_{j=1}^h \widetilde{\alpha}_i^* \cdot \widetilde{\alpha}_j^* \cdot \text{cov}(z^i, z^j)}{\sigma(z)} \tag{4.21}$$

thus, using Equation (4.12), we get:

$$\text{GeDI}(z, \hat{y}; h) = \frac{\sigma(\mathbf{P}_z^h \cdot \widetilde{\alpha}^*)}{\sigma(z)} \leq \frac{\sum_{i=1}^h \sum_{j=1}^h \tau_i \cdot \tau_j \cdot |\text{cov}(z^i, z^j)|}{\sigma(z)} \tag{4.22}$$

Optimized Fine-Grained Formulation

An interesting use of this feature is to enable a single, easily explainable dependency while explicitly prohibiting others. Specifically, when referring to the "Fine-Grained Formulation" in our experiments, we will restrict any higher-order dependencies, hence limiting all permissible unfairness to the linear form. This enforcement can

be alternatively seen as:

$$\text{GeDI}(z, \hat{y}; h) = \text{GeDI}(z, \hat{y}; 1) \leq \tau \tag{4.23}$$

where $\tau \in \mathbb{R}^+$ is a positive scalar.

Starting from the declarative formulation described in Equation (4.20), the problem can be rewritten as:

$$\begin{gathered} \widetilde{\mathbf{P}}_z^T \cdot \widetilde{\mathbf{P}}_z \cdot \begin{bmatrix} \widetilde{\alpha}_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \widetilde{\mathbf{P}}_z^T \cdot \widetilde{y} \\ -\tau \leq \widetilde{\alpha}_1 \leq \tau \end{gathered} \tag{4.24}$$

where the two known terms in the linear system from Equation (4.24) evaluate to:

$$\widetilde{\mathbf{P}}_z^T \cdot \widetilde{\mathbf{P}}_z = \begin{bmatrix} \text{var}(z) & \text{cov}(z, z^2) & \dots & \text{cov}(z, z^h) \\ \text{cov}(z^2, z) & \text{var}(z^2) & \dots & \text{cov}(z^2, z^h) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(z^h, z) & \text{cov}(z^h, z^2) & \dots & \text{var}(z^h) \end{bmatrix} \tag{4.25}$$

$$\widetilde{\mathbf{P}}_z^T \cdot \widetilde{y} = \begin{bmatrix} \text{cov}(z, \hat{y}) \\ \text{cov}(z^2, \hat{y}) \\ \vdots \\ \text{cov}(z^h, \hat{y}) \end{bmatrix} \tag{4.26}$$

given that both $\widetilde{\mathbf{P}}_z$ and $\widetilde{y}$ are zero-centered.

The outcome is the following system of k equations under a single variable $\widetilde{\alpha}_1$:

$$\text{var}(z) \cdot \widetilde{\alpha}_1 = \text{cov}(z, \hat{y}) \tag{4.27}$$

$$\text{cov}(z^i, z) \cdot \widetilde{\alpha}_1 = \text{cov}(z^i, \hat{y}), \quad \forall i \in \{2, \dots, h\} \tag{4.28}$$

From Equation (4.27), we obtain the solution:

$$\widetilde{\alpha}_1 = \frac{\text{cov}(z, y)}{\text{var}(z)} \tag{4.29}$$

whose result is in line with the value of $\text{GeDI}(z, \hat{y}; 1)$. As regards the remaining $h - 1$ equations outlined in Equation (4.28), they represent constraints that must be satisfied by the projection vector $\hat{y}$. Specifically, by substituting the value of $\widetilde{\alpha}_1$, we can obtain a closed-form expression:

$$\text{cov}(z^i, z) \cdot \frac{\text{cov}(z, \hat{y})}{\text{var}(z)} = \text{cov}(z^i, \hat{y}) \tag{4.30}$$

Eventually, we can move the denominators of both the definition of $\widetilde{\alpha}_1$ and the additional constraints to the right-hand side. This leads us to the following optimized formulation, where we bypass the need to solve any least-square problem:

$$\begin{aligned} &-\tau \cdot \text{var}(z) \leq \text{cov}(z, \hat{y}) \leq \tau \cdot \text{var}(z) \\ &\text{cov}(z^i, z) \cdot \text{cov}(z, \hat{y}) = \text{cov}(z^i, \hat{y}) \cdot \text{var}(z), \quad \forall i \in \{2, \ldots, h\} \end{aligned} \tag{4.31}$$

4.4 Experimental Analysis

After presenting the formulation of the GeDI indicator, we conduct two experimental analyses aimed at comparing the advantages and shortcomings of its different implementations and establishing its applicability in practical scenarios. In the first experiment, we evaluate the differences between the coarse- and the fine-grained formulation, and what is their link with the DIDI metric. Instead, the second experiment involves the use of various machine learning models and constraint enforcement algorithms to determine whether GeDI can serve as a constraint in either its declarative or regularizer form. Both experiments use the same datasets previously discussed in Section 4.1, specifically, the US 2015 Census (*Census*), Communities & Crimes (*Communities*), and Adult Census Income (*Adult*). Further information on datasets and algorithms is available in Appendix A.

4.4.1 Projections Experiments

In this experiment, we employ the declarative formulations of GeDI to directly project the target data into the feasible region, without any training involved. In practice, we tackle a version of Equation (4.13) which is modified as follows:

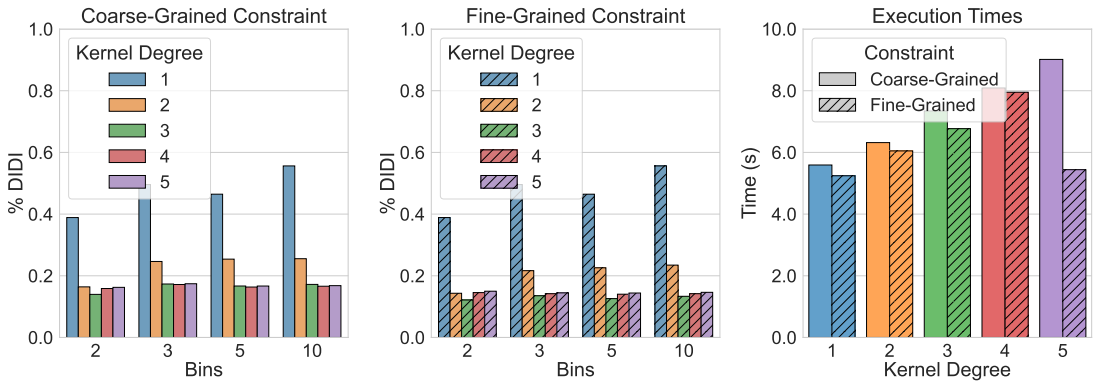
$$\arg\min_{\hat{y}} \mathcal{L}(y, \hat{y}) \quad \text{s.t.} \quad \text{GeDI}(z, \hat{y}; h) \leq \tau_{\%} \cdot \text{GeDI}(z, y; 1), \quad \tau_{\%} \in [0, 1] \tag{4.32}$$

The difference lies in the application of a relative constraint with a threshold of $\tau_{\%}$, indicating our intention to limit the $\text{GeDI}(z, \hat{y}; h)$ value to a percentage of the unfairness calculated on the original dataset. This approach yields results that are more easily comparable across different datasets since, unlike HGR, GeDI lacks a theoretical upper limit. Furthermore, we evaluate the reference unfairness using the indicator with unitary degree $\text{GeDI}(z, y; 1)$; again, this favors more comparable results across different kernel degrees h.

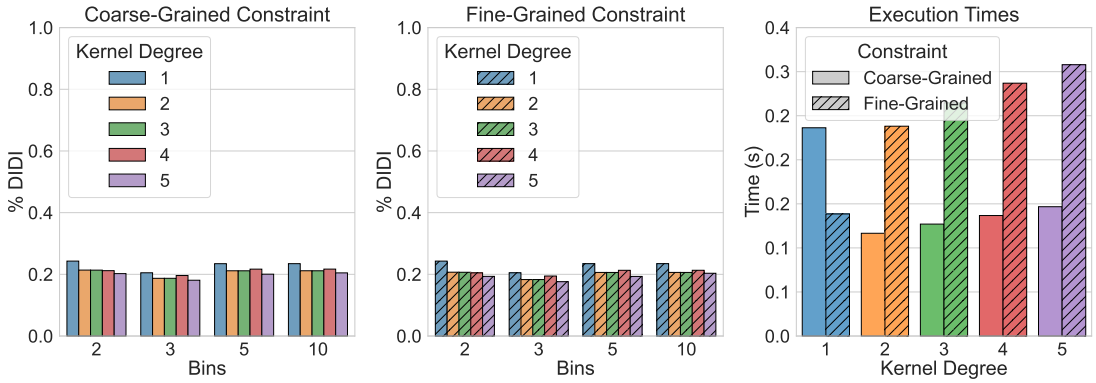
To solve the problem outlined in Equation (4.32), we use the APIs provided by the `gurobipy` package. For the coarse-grained constraint, we implement the formulation from Equation (4.18), whereas for the fine-grained we adopt the optimized strategy detailed in Equation (4.31). In both scenarios, we employ Mean Squared Error – i.e., Euclidean Distance – as the loss function $\mathcal{L}$ for regression tasks (*Census* and *Communities*) and Hamming Distance for classification ones (*Adult*). Upon obtaining the projections $\hat{y}$, we compute a binned version of the DIDI metric by employing a quantile-based discretization into n bins prior to calculating the indicator. Specifically, given the continuous protected attribute z, we initially split it into n equally-sized bins, and subsequently compute the DIDI metric on the resulting vector z_n. Eventually, we report the percentage of unfairness relative to the original targets, i.e.:

$$\%\text{DIDI}_n = \frac{\text{DIDI}(z_n, \hat{y})}{\text{DIDI}(z_n, y)} \tag{4.33}$$

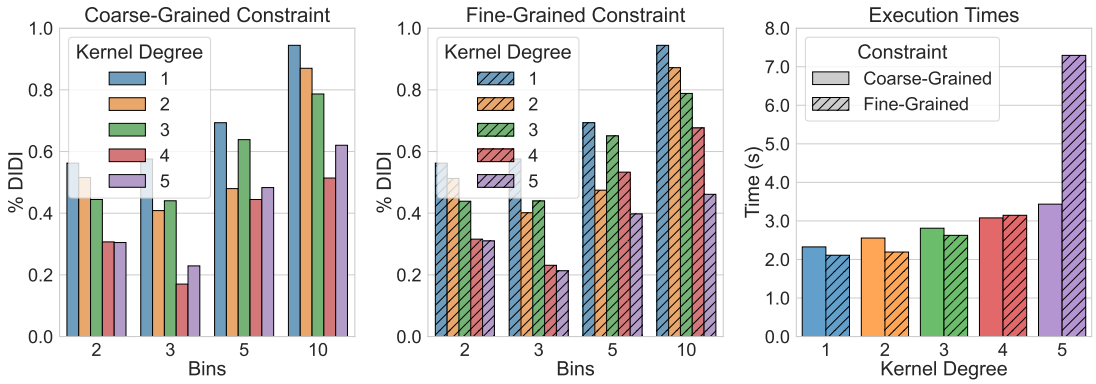
Figure 4.5 presents the results on the three datasets using a relative threshold of $\tau_{\%} = 0.2$. The plots on the left are obtained using the coarse-grained formulation, while the central ones leverage the fine-grained approach. On the x-axis we denote the number of bins n for the $\%\text{DIDI}_n$ metric, whose value is eventually reported on



(a) *Census*



(b) *Communities*



(c) *Adult*

Figure 4.5: Percentage Binned DIDI (%DIDI$_n$) computed on the target projections of the three benchmark datasets using both the Coarse-Grained and Fine-Grained GeDI formulation with varying degrees h and relative threshold $\tau_{\%} = 0.2$.

the y-axis. For every setup, we apply five varying polynomial degrees h, indicated by different colors, to enforce the constraint on $\text{GeDI}(z, \hat{y}; h)$.

Given that this process does not involve training any machine learning model, and that GeDI constraints are inherently satisfied as we use an exact solver, variations in behavior are solely due to changes in the semantics determined by the chosen degree h. Notably, no consistent differences appear between the coarse- and fine-grained approaches, as both methods yield comparable values of disparate impact across all discretizations. Nonetheless, we observe that increasing the polynomial degree effectively diminishes the disparate impact across all the numbers of bins in both formulations and, in certain instances, we also notice that the relative threshold for applying GeDI is maintained when evaluating the Binned DIDI – this is particularly evident in the *Communities* dataset and also noticeable in the *Census* dataset, provided that h is adequately high. These observations reinforce the insight that a higher kernel order enhances the ability of our indicator to assess more complex dependencies, as well as strengthening the connection between the two metrics – despite it being important to emphasize that the equivalence discussed in Section 4.3.2 does not strictly apply to the datasets examined, as the protected attribute is not inherently categorical here. Additionally, in the few cases where $\%\text{DIDI}_3$ and $\%\text{DIDI}_5$ worsen with increasing polynomial degrees, we can hypothesize that this is a result of how the bin boundaries "cut" the shapes introduced by the polynomial kernel, hence indicating discretization noise as a possible cause.

Lastly, in the plots on the right, we present the time required to solve the optimization problem in Equation (4.32) by each formulation for each degree. Overall, there is a minimal difference in the time needed by the coarse-grained versus the fine-grained formulation, with the former generally being somewhat more efficient because of fewer constraints. Still, the fine-grained approach might be favored in certain applications in order to have more interpretable outcomes or for excluding specific undesired dependencies.

4.4.2 Learning Experiments

This second experiment mirrors the structure of that in Section 4.1, with the main difference of using several learning models. Indeed, in addition to Neural Networks (NN), we test Linear/Logistic Regression Models (LM), Random Forests (RF), and Gradient Boosting (GB). For each of them, in order to evaluate the effectiveness of the declarative formulation of GeDI, we enforce both the coarse- and fine-grained constraints using Moving Targets; moreover, we also test the regularizer form using the Lagrangian Dual framework when employing Neural Networks.

The experiments are conducted using a 5-fold cross-validation procedure on the entire dataset. In line with previous HGR experiments, we select $h = 5$ for the polynomial degree when calculating and enforcing the GeDI indicator, as it offers a balanced compromise between computational demands and generalization potential. For each dataset, we determine an absolute threshold τ, which is defined as 20% of the GeDI value calculated on the entire data using a unitary polynomial degree. This yields the following thresholds:

$$\tau_{\texttt{Census}} = 0.025 \qquad \tau_{\texttt{Communities}} = 0.017 \qquad \tau_{\texttt{Adult}} = 0.021$$

The protected, target, and surrogate binary attributes are identical to those outlined in Table 4.1, with the neural models configured using the same hyper-parameters described. Similarly to what is shown in Table 4.2, we also report the DIDI metric computed on the surrogate binary attribute to reinforce the relationship between GeDI and DIDI. For a complete list of our hardware and software setup, refer to Appendix A.1; instead, for an expanded explanation of the implementation details related to both the learning models and the constraint enforcement algorithms, see Appendix A.4.2.

Table 4.3 reports the outcomes of these experiments. We provide average values and standard deviations for the train and validation splits concerning the "Score" metric, the "GeDI" computed on the continuous protected attribute z, and the "DIDI" calculated on the surrogate binary attribute s; additionally, the last column lists training times. Unlike in Table 4.2, where the comparisons on the constraints were not entirely fair due to differences in the semantics of the indicators, here all

Algorithm	Score ($\times 10^2$)		GeDI_z($\times 10^3$)		DIDI_s($\times 10^2$)		Time (s)
	train	val	train	val	train	val	
CENSUS ($\tau = 25$)							
LM	**62 ± 00**	**62 ± 00**	**135 ± 01**	**135 ± 00**	**19 ± 00**	**19 ± 00**	**98 ± 000**
+MT Fine	21 ± 00	21 ± 00	25 ± 00	25 ± 00	04 ± 00	04 ± 00	154 ± 008
+MT Coarse	24 ± 00	24 ± 00	25 ± 00	25 ± 00	06 ± 00	06 ± 00	182 ± 008
RF	**96 ± 00**	**70 ± 00**	**153 ± 01**	**153 ± 01**	**18 ± 00**	**18 ± 00**	**281 ± 001**
+MT Fine	31 ± 00	21 ± 00	25 ± 00	25 ± 00	05 ± 00	05 ± 00	1701 ± 040
+MT Coarse	35 ± 00	25 ± 00	25 ± 00	26 ± 01	05 ± 00	05 ± 00	1941 ± 382
GB	**72 ± 00**	**71 ± 00**	**153 ± 01**	**153 ± 01**	**19 ± 00**	**19 ± 00**	**34 ± 000**
+MT Fine	20 ± 00	20 ± 00	25 ± 00	25 ± 00	05 ± 00	05 ± 00	400 ± 003
+MT Coarse	24 ± 00	24 ± 00	25 ± 00	25 ± 00	05 ± 00	05 ± 00	406 ± 007
NN	**70 ± 00**	**69 ± 00**	**152 ± 01**	**152 ± 01**	**19 ± 00**	**19 ± 00**	**24 ± 000**
+MT Fine	24 ± 00	22 ± 00	25 ± 01	25 ± 01	05 ± 00	05 ± 00	117 ± 029
+MT Coarse	28 ± 00	26 ± 00	25 ± 01	25 ± 01	06 ± 00	06 ± 00	172 ± 008
+LD Fine	25 ± 02	24 ± 01	38 ± 01	38 ± 02	06 ± 00	06 ± 00	112 ± 000
+LD Coarse	29 ± 04	28 ± 04	31 ± 06	31 ± 06	06 ± 01	06 ± 01	140 ± 000
COMMUNITIES ($\tau = 17$)							
LM	**68 ± 02**	**57 ± 07**	**86 ± 02**	**86 ± 07**	**27 ± 01**	**25 ± 02**	**00 ± 000**
+MT Fine	38 ± 02	28 ± 08	17 ± 00	20 ± 08	08 ± 01	07 ± 02	02 ± 000
+MT Coarse	38 ± 02	28 ± 09	17 ± 00	20 ± 07	07 ± 01	06 ± 02	02 ± 000
RF	**95 ± 00**	**63 ± 02**	**85 ± 02**	**83 ± 05**	**27 ± 01**	**27 ± 02**	**07 ± 000**
+MT Fine	46 ± 01	34 ± 05	18 ± 00	20 ± 05	08 ± 00	08 ± 02	78 ± 001
+MT Coarse	46 ± 01	33 ± 05	18 ± 00	20 ± 05	08 ± 00	08 ± 02	78 ± 001
GB	**87 ± 01**	**63 ± 02**	**85 ± 02**	**83 ± 05**	**27 ± 01**	**27 ± 02**	**04 ± 000**
+MT Fine	44 ± 01	34 ± 06	17 ± 00	19 ± 06	08 ± 00	08 ± 02	37 ± 000
+MT Coarse	44 ± 01	34 ± 07	17 ± 00	19 ± 06	08 ± 00	08 ± 02	37 ± 000
NN	**100 ± 01**	**52 ± 02**	**85 ± 04**	**84 ± 07**	**26 ± 01**	**25 ± 04**	**04 ± 000**
+MT Fine	70 ± 01	26 ± 05	17 ± 00	20 ± 08	07 ± 01	05 ± 01	32 ± 000
+MT Coarse	70 ± 01	26 ± 04	17 ± 00	19 ± 08	06 ± 01	05 ± 02	33 ± 000
+LD Fine	76 ± 02	38 ± 05	40 ± 04	35 ± 10	18 ± 01	16 ± 04	07 ± 000
+LD Coarse	70 ± 05	25 ± 05	18 ± 05	21 ± 10	07 ± 02	06 ± 01	08 ± 000
ADULT ($\tau = 21$)							
LM	**91 ± 00**	**90 ± 00**	**117 ± 01**	**117 ± 03**	**28 ± 00**	**28 ± 01**	**02 ± 000**
+MT Fine	77 ± 01	77 ± 01	44 ± 05	44 ± 03	10 ± 02	10 ± 01	437 ± 055
+MT Coarse	76 ± 01	76 ± 01	34 ± 04	34 ± 03	09 ± 01	09 ± 01	28 ± 002
RF	**100 ± 00**	**90 ± 00**	**139 ± 00**	**134 ± 02**	**30 ± 00**	**28 ± 01**	**11 ± 000**
+MT Fine	92 ± 00	83 ± 01	21 ± 00	30 ± 02	12 ± 00	09 ± 02	70 ± 010
+MT Coarse	93 ± 00	83 ± 01	21 ± 00	31 ± 02	13 ± 00	09 ± 02	55 ± 001
GB	**92 ± 00**	**92 ± 00**	**125 ± 01**	**125 ± 04**	**26 ± 00**	**26 ± 01**	**19 ± 000**
+MT Fine	80 ± 01	79 ± 00	21 ± 02	24 ± 04	12 ± 01	12 ± 01	293 ± 095
+MT Coarse	77 ± 04	77 ± 03	34 ± 04	37 ± 06	14 ± 02	14 ± 02	73 ± 009
NN	**92 ± 00**	**91 ± 00**	**141 ± 03**	**140 ± 06**	**29 ± 01**	**29 ± 02**	**48 ± 002**
+MT Fine	72 ± 01	71 ± 02	36 ± 06	36 ± 08	19 ± 01	19 ± 01	495 ± 046
+MT Coarse	72 ± 01	71 ± 01	37 ± 06	39 ± 07	19 ± 01	19 ± 02	133 ± 004
+LD Fine	90 ± 00	89 ± 00	81 ± 06	83 ± 05	23 ± 02	23 ± 01	97 ± 000
+LD Coarse	91 ± 00	91 ± 00	101 ± 02	101 ± 03	25 ± 01	25 ± 00	109 ± 000

Table 4.3: Results of GeDI learning experiments conducted on the three benchmarks.

models use the same constraint, thus a fair comparison is ensured. Nonetheless, the objective of this experiment is not to determine which model performs best, but rather to confirm that any combination of constraint enforcement algorithm and indicator implementation yields accurate results, and that the availability of various combinations is a beneficial aspect of the GeDI indicator which is, in contrast, absent in any HGR implementation. For this reason, we simply emphasize the unconstrained models using bold fonts to favor a comparison against them.

Overall, it appears that both accuracy and constraint satisfaction exhibit fairly consistent behaviors, as reflected by the low standard deviations across nearly all approaches. Threshold levels are generally met in most constrained approaches, except for the classification task in the `Adult` dataset where many models fail to meet the desired bound even in the training split. Additionally, constraints enforced through the regularizer form often fall short of the required threshold. This might be attributed to less informative gradients provided by the GeDI indicator, or it may suggest that a greater number of steps is necessary for convergence.

In all cases, a reduction of the unfairness on the continuous protected attribute correlates with a decrease in the DIDI respective to the binary surrogate, thereby restating the alignment between our indicator and the well-established metric. Concerning execution times, instead, Moving Targets consistently demands a notably higher computational effort compared to the Lagrangian Dual framework, particularly when handling fine-grained constraints. Nonetheless, it remains a viable choice as it offers the flexibility to use various learning models, thus enabling the user to find a balance between the necessary computational time, the degree of constraint satisfaction, and the accuracy relative to the task score.

4.5 Long-Term Fairness in Ranking

Before concluding this chapter, we present a practical application of our GeDI indicator that we introduced in [Giuliani et al., 2024]. In this study, we leveraged real-world sociodemographic data collected by the Canarian Agency for Quality Assessment and Accreditation[2] from students over four academic years (2015-2019),

[2]The original dataset can be accessed at: `https://zenodo.org/records/11171863`.

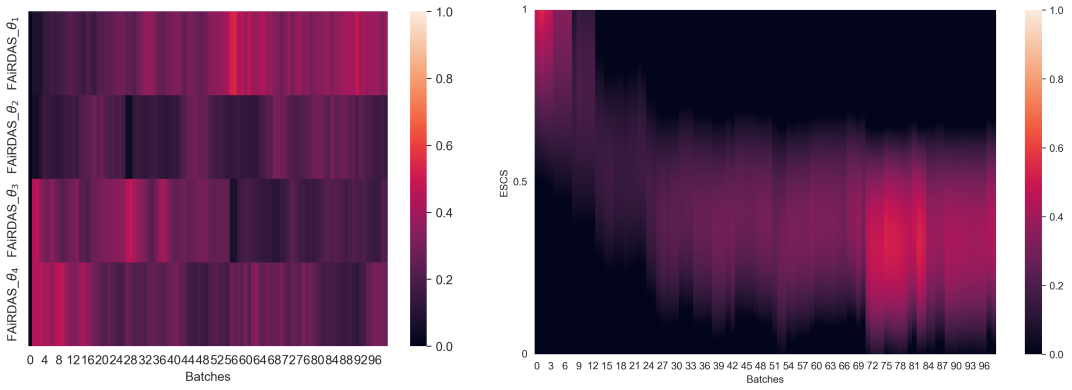
and ultimately used it to rank students based on their predicted academic score in mathematics in order to foresee potential risks of dropout.

The given dataset contains multiple sensitive information; for this reason, we selected the ESCS indicator [Avvisati, 2020] as our protected attribute, given that it aggregates various Economic, Social, and Cultural factors into a single variable. Furthermore, in order to achieve high accuracy and fairness while avoiding detrimental effects on the academic development of students, we opted for the FAiRDAS [Misino et al., 2023] framework to enforce constraints. This choice was based on its ability to model long-term fairness as a dynamic system, as it focuses on keeping fairness and task metrics stable under user-defined limits in ranking tasks over time rather than in individual batches. FAiRDAS falls into the class of post-processing methods for fairness enforcement, and requires the grounding of its three core components in order to be applied to a specific use case, namely:

1. the chosen metrics for fairness and task score, with their respective thresholds;
2. a set of actions aimed at adjusting predictions to optimize the balance between fairness and score;
3. a distance function and an optimization method to solve the dynamic system governing the progression of the algorithm.

In our research, we explored two distinct groundings of FAiRDAS. The first one, inspired by [Misino et al., 2023], implements a set of discrete actions that require the discretization of the protected attribute; conversely, the second instance relies on continuous actions, thus eliminating the need for discretization. Nonetheless, it is important to note that the discretization is solely related to the actions, hence we use the GeDI indicator as a fairness measure in both cases since the protected attribute is still continuous at that stage. Moreover, we underline that the discussed paper was authored before the preparation of this dissertation; therefore, the formulation of GeDI used in the numerical experiments aligns with [Giuliani et al., 2023] rather than incorporating the differences outlined in Section 4.3.1.

Figure 4.6 illustrates the action vectors produced by FAiRDAS for both the discrete and the continuous grounding over 100 consecutive batches using thresholds $\{0.2, 0.2\}$. In the discrete grounding, each part of the action vectors impacts students



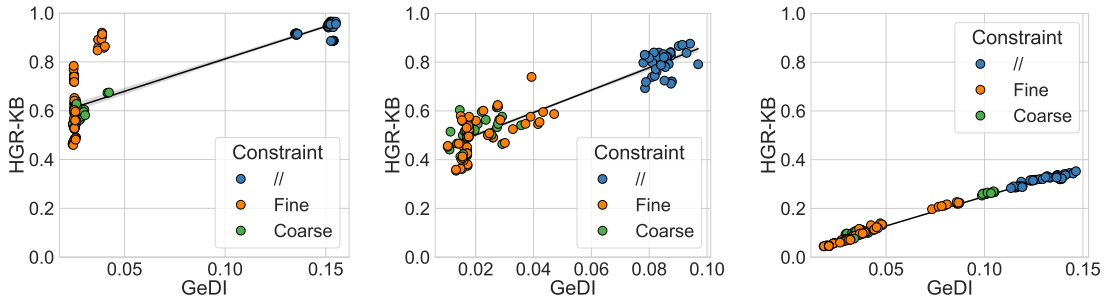
(a) Discrete Grounding (b) Continuous Grounding

Figure 4.6: Actions returned by FAiRDAS using thresholds ($\{0.2, 0.2\}$).
From: *Long-Term Fairness Strategies in Ranking with Continuous Sensitive Attributes* [Giuliani et al., 2024]

from the respective ESCS bin, from lower (θ_4) to upper levels (θ_1). Higher values are represented by lighter colors and signify greater penalization, which might in turn affect the ranking of the student; values closer to zero, instead, indicate that the student's score remains unaffected. We can observe that FAiRDAS displays a moderate and balanced behavior, with the action vectors evolving smoothly during the experiment, as indicated by the gradual color shifts along the x-axis, and similar penalties applied across groups, as indicated by uniform color along the y-axis. Similarly, within the continuous grounding, we use a learnable polynomial function to represent the penalty of each student based on the value of their ESCS across its domain $[0, 1]$. Here, each vertical line in the plot represents the penalty function obtained for a given batch, and FAiRDAS shows again a moderated and balanced approach, with polynomial functions transitioning smoothly across the batches and a more steady penalization across varying ESCS values.

4.6 GeDI and HGR Comparison

When we presented the GeDI indicator, we highlighted that it is designed to complement HGR instead of replacing it. In fact, both indicators share common features, starting from the non-linear copula transformations and going through the lower-level details of their implementation. However, they also have significant



(a) *Census* (b) *Communities* (c) *Adult*

Figure 4.7: Comparison between computed values of GeDI and HGR-KB during the learning experiments on the three benchmark datasets.

differences that partially change their semantics and the yielded results.

Using the results obtained from the learning experiment discussed in the previous section, Figure 4.7 empirically examines the relationship between the GeDI and HGR indicators. For each vector of predictions $\hat{y}(\theta)$ yielded by both constrained and unconstrained models across the three datasets, we compute $\text{GeDI}(z, \hat{y}(\theta); 5)$ and $\text{HGR-KB}(z, \hat{y}(\theta); 5, 5)$ and report their values on the x and y axes, respectively. As expected, unconstrained models typically exhibit a higher disparate impact than constrained models, but more interestingly this aspect is correlated with an increase in the HGR value. The black line in the figure shows the amount of the (linear) correlation between the values returned by the indicators; as we can see, the fitting is not perfect but rather shows some irregularity, particularly in the datasets related to regression tasks. This emphasizes that both GeDI and HGR can serve as valid fairness metrics; they correlate with one another, yet each captures distinct aspects depending on to their ability to capture non-functional dependencies and to address scaling effects, from which the noisy behavior shown in Figure 4.7 stems. Since fairness measures are strongly application-specific, neither indicator is better, but rather their utility is tied to the unique scenario taken into consideration.

4.6.1 A Broader Spectrum of Indicators

In Table 4.4, we report the main similarities and differences between GeDI and (Kernel-Based) HGR. This serves to illustrate that both indicators could be part

	GeDI	**HGR-KB**	**HGR-SK**
Transformations	f	f **and** g	f **or** g
Dependencies			
Functional	✓	✓	✓
Non-Functional	×	✓	✓
Scale-Dependent	✓	✓	✓
Scale-Independent	✓	×	×
Constraints			
Regularizer	✓	✓	✓
Declarative	✓	×	×
Coarse-Grained	✓	✓	✓
Fine-Grained	✓	×	×
Properties			
Configurability	✓	✓	✓
Determinism	✓	✓	✓
Monotonicity	✓	✓	✓
Least-Square	✓	×	✓

Table 4.4: Characteristics of the GeDI indicator when compared to HGR.

of a broader spectrum, with additional metrics filling the gap between these two. For instance, although both declarative and fine-grained models are theoretically feasible even for HGR indicators, their practical implementation is hindered by computational constraints, which make them impractical due to prohibitive computational times. As a solution to that, we could introduce a functional form of HGR, expressed as:

$$\text{HGR-FN}(a, b; h) = \max_{\alpha} \frac{\text{cov}(\mathbf{P}_a^h \cdot \alpha, b)}{\sigma(\mathbf{P}_a^h \cdot \alpha) \cdot \sigma(b)} \tag{4.34}$$

This formulation aligns more closely with the current functional definition of GeDI, enabling the application of similar methods for achieving both declarative and fine-grained constraints. Likewise, despite having defined GeDI to account solely for functional dependencies, we could relax this condition by applying a copula transformation to the target vector and imposing a standard deviation constraint similar to that of the input. This would lead to an intermediate indicator, which

is sensitive to scale fluctuations but can account for non-functional dependencies. Formally, it would be defined as:

$$\begin{aligned} \text{GeDI-KB}(a, b; h, k) &= \max_{\alpha,\beta} \frac{\text{cov}(\mathbf{P}_a^h \cdot \alpha, \mathbf{P}_b^k \cdot \beta)}{\text{var}(\mathbf{P}_a^h \cdot \alpha)} \\ \text{s.t.} \quad \sigma(\mathbf{P}_a^h \cdot \alpha) &= \sigma(a), \quad \sigma(\mathbf{P}_b^k \cdot \beta) = \sigma(b) \end{aligned} \tag{4.35}$$

and can be seen as a generalization of our original GeDI definition for $k > 1$. Moreover, by employing the same approximations used in HGR-SK, we could build a respective GeDI-SK indicator, hence closing both the semantic and the computational distance between the two metrics.

4.6.2 Unified Formulation for Non-Linear Indicators

The theoretical contribution of narrowing the semantic gap between GeDI and HGR should not be limited to our kernel-based methodology only. As discussed in Section 4.3.1, the reworked formulation of GeDI could allow its estimation even with alternative algorithmic approaches that do not rely on mixing coefficients. Let us take Equation (4.35) into account; by replacing our kernel-based transformations with more general functions f and g, we get:

$$\text{GeDI}(a, b) = \max_{f,g} \frac{\text{cov}(f_a, g_b)}{\text{var}(a)} \qquad \text{s.t.} \quad \sigma(f_a) = \sigma(a), \quad \sigma(g_b) = \sigma(b) \tag{4.36}$$

where f_a and g_b are aliases of $f(a)$ and $g(b)$, and $\text{var}(f_a)$ has been replaced by $\text{var}(a)$ in line with the respective constraint on the standard deviation.

Recalling the original definition of Kernel-Based HGR, and given the scale invariance of the metric, we can impose constraints on the standard deviation of the projected vectors without affecting the optimal solution. Specifically, rather than constraining unitary standard deviations as in Equation (3.2), we reformulate it in its sample variation using the same constraints of Equation (4.36), i.e.:

$$\text{HGR}(a, b) = \max_{f,g} \frac{\text{cov}(f_a, g_b)}{\sigma(a) \cdot \sigma(b)} \qquad \text{s.t.} \quad \sigma(f_a) = \sigma(a), \quad \sigma(g_b) = \sigma(b) \tag{4.37}$$

accordingly substituting $\sigma(f_a)$ with $\sigma(a)$ and $\sigma(g_b)$ with $\sigma(b)$ in the denominator.

Equations (4.36) and (4.37) illustrate how discovering a unified framework between HGR and GeDI is straightforward and does not rely on the choice of the estimation algorithm. Notably, with constant denominators in both equations, and since the numerators are the same, we deduce that the optimal copula transformations for both scenarios are identical and that the two measures are equivalent except for a scaling factor, or rather:

$$\mathrm{GeDI}(a, b) = \mathrm{HGR}(a, b) \cdot \frac{\sigma(a)}{\sigma(b)} \tag{4.38}$$

Within this dissertation, we decided not to evaluate these alternative indicators, as they merely represent variations of the two primary ones, lacking any additional insights. Nonetheless, for practical use, some indicators may be more appropriate than others. To address this, we developed a Python package named `maxcorr`, which implements both GeDI and HGR semantics, allowing for the use of two different copula transformations to address non-functional dependencies or a single transformation to focus solely on functional dependencies. Additionally, we included a third semantics, called Non-Linear Covariance (NLC), which simply extends the covariance operator into the non-linear domain as follows:

$$\begin{aligned} \mathrm{NLC}(a, b) &= \max_{f,g} \mathrm{cov}(f_a, g_b) \qquad \text{s.t.} \quad \sigma(f_a) = \sigma(a), \quad \sigma(g_b) = \sigma(b) \\ &= \mathrm{HGR}(a, b) \cdot \sigma(a) \cdot \sigma(b) \end{aligned} \tag{4.39}$$

We remark that such formulations offer the advantage of being algorithm-independent. For this reason, the package includes six distinct computational algorithms: (*i*) our Kernel-Based formulation; (*ii*) our Single-Kernel approximation; (*iii*) the Neural-Based algorithm from [Grari et al., 2020] and (*iv*) a variation of it using Lattice Models instead of Neural Networks; (*v*) the Kernel-Density approach by [Mary et al., 2019]; and (*vi*) the Randomized Dependence Coefficient described in [Lopez-Paz et al., 2013]. The source code of `maxcorr` is publicly available under MIT License and can be accessed at `https://github.com/giuluck/maxcorr`, or it may be installed via any Python package manager.

4.7 Final Discussion

We started this chapter by showing that our HGR formulations can be applied to the task of fair learning, proving that their computation is differentiable and yields informative gradients that can guide the learning procedure towards fair regions of the space. Following this, we explored certain drawbacks of the HGR coefficient when used as a fairness metric: we highlighted that its semantics could pose issues due to its measurement of non-functional dependencies and its invariance to scale and, in order to address this, we introduced a new indicator, termed Generalized Disparate Impact (GeDI), which slightly alters the semantics of HGR to align more closely with the legal notion of disparate impact even in the presence of continuous protected attributes. We further elaborated on the definition of GeDI and the method to compute it, as well as discussing several strategies to constraint its value below a certain threshold by means of different interpretations and solution approaches. Subsequently, we conducted an empirical evaluation that confirmed the connection between GeDI and disparate impact, as measured by the more conventional DIDI indicator which is only applicable to categorical data, and demonstrated the versatility of GeDI in the context of fairness enforcement. In conclusion, we examined another study where we applied GeDI to a real-world scenario about fair ranking of student performances, and finally discussed the similarities and differences between HGR and GeDI. Specifically, we noted that a spectrum of other indicators could potentially bridge the gap between the two, and thereby introduced a unified formulation for all the non-linear correlation indicators, showing how they are all equivalent except for a data-dependent scaling factor.

Chapter 5

Future Directions

Before proceeding with our concluding remarks, we will briefly outline some research paths that we are interested in exploring or have begun examining without definite conclusions yet. Specifically, Section 5.1 will discuss the use of different kernels to compute HGR in a more efficient and effective manner. Section 5.2 will investigate potential methods to extend the definition of HGR to encompass multivariate correlations, along with scenarios where this can be of interest. Lastly, in Section 5.3, we will present a proof of concept on the use of HGR to remove biases from datasets based on a causal, rather than statistical, definition of fairness.

5.1 Different Kernel Expansions

For both our HGR and GeDI indicators, we used polynomial expansions to compute correlations. This is just one approach among many, as various other types of kernel can be employed. Let us consider two functions $F : \mathbb{R} \mapsto \mathbb{R}^h$ and $G : \mathrm{R} \mapsto \mathrm{R}^k$, we could formulate the HGR indicator as:

$$\text{HGR-KB}(a, b; F, G) = \max_{\alpha,\beta} \rho(F(a) \cdot \alpha, G(b) \cdot \beta) \tag{5.1}$$

where $F(a) = \begin{bmatrix} F_1(a) & \dots & F_h(a) \end{bmatrix}$ and $G(b) = \begin{bmatrix} G_1(b) & \dots & G_k(b) \end{bmatrix}$, and each F_i and G_i represents an independent and univariate mapping function.

With respect to Equation (5.1), the indicator discussed in Chapter 3 is just one

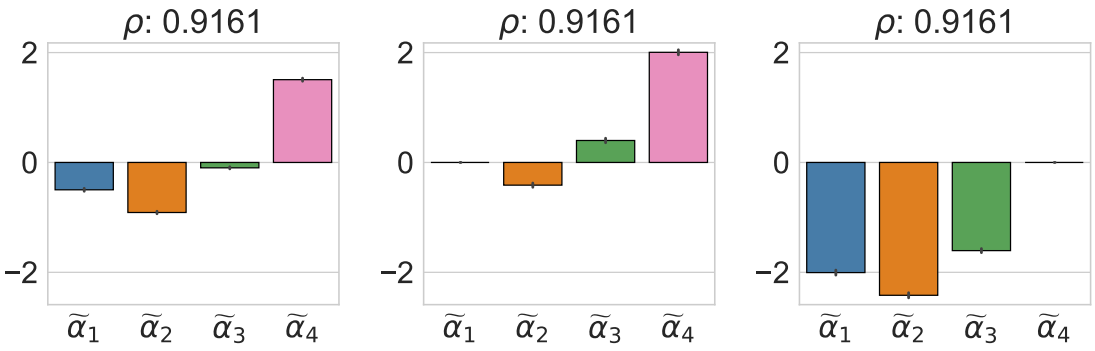
of its possible instances where we set $F(a) = \mathbf{P}_a^h$ and $G(b) = \mathbf{P}_b^k$. Nonetheless, if certain background knowledge or system requirement can be expressed through a particular relationship, this could easily be incorporated into the kernel as one of the mapping functions. Conversely, if we lack prior information and we wish to preserve the flexibility of polynomials, alternative universal approximations can be employed. Of these, sinusoidal functions and one-hot encoding are noteworthy options for modeling continuous and categorical variables, respectively.

Sinusoidal Kernels

One significant drawback of polynomials is that their computations rely on the Vandermonde matrix $\begin{bmatrix} x & x^2 & \dots & x^h \end{bmatrix}$, which is known for its numerical instability. Although our experiments confirmed that low kernel degrees are often sufficient to identify the majority of patterns in the data, having a more stable alternative could be advantageous in certain scenarios. Sinusoidal functions may provide such an option, as they offer greater stability and are less prone to numerical errors. Furthermore, while polynomial kernels can approximate any dependency through their association with the Taylor series, sinusoidal functions are closely linked with Fourier expansions, thereby offering the same capability. Selecting which sinusoidal component to incorporate into the kernel is a challenge, nevertheless. The Randomized Dependence Coefficient employs randomly selected scaling coefficients ω, but we have already demonstrated that this approach is problematic due to the intrinsic lack of determinism, therefore a more reliable solution would be preferable. Additionally, it is worth noting that polynomials are generally more interpretable for human evaluators compared to sinusoidal functions, as the latter require reasoning within the frequency domain to understand the outcomes.

One-Hot Kernels

An additional challenge with polynomial kernels is that they are ineffective or ill-defined when applied to categorical data. Indeed, binary variables are invariant to polynomial expansions, while multi-class ones can benefit from projections into higher dimensions, although in a scarcely interpretable way. A potential remedy could be to use one-hot encoding for categorical variables. Given a variable



(a) All Columns (b) Column 1 Removed (c) Column 4 Removed

Figure 5.1: Coefficients of the one-hot kernel on a categorical variable, with three distinct ways to handle overspecification.

$a \in \{1, \dots, c\}^n$, the generated kernel takes the form:

$$F(a) = \begin{bmatrix} \mathbb{I}(a_1 = 1) & \mathbb{I}(a_1 = 2) & \dots & \mathbb{I}(a_1 = c) \\ \mathbb{I}(a_2 = 1) & \mathbb{I}(a_2 = 2) & \dots & \mathbb{I}(a_2 = c) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{I}(a_n = 1) & \mathbb{I}(a_n = 2) & \dots & \mathbb{I}(a_n = c) \end{bmatrix} \tag{5.2}$$

An issue with this type of kernel is the intrinsic lack of linear independence among columns, as any column F_i can be represented as $1 - \sum_{j \neq i} F_j$. Figure 5.1(a) illustrates this problem on a synthetic dataset by reporting the coefficients of the one-hot kernel applied to a variable $a \in \{1, 2, 3, 4\}^n$. To build the dataset, we uniformly sampled 1000 values for a and constructed a target variable b as follows:

$$b_i = \begin{cases} 0 & \text{if } a_i = 1 \\ -1 & \text{if } a_i = 2 \\ 1 & \text{if } a_i = 3 \\ 5 & \text{if } a_i = 4 \end{cases} + \quad \epsilon \tag{5.3}$$

with $\epsilon \sim \mathcal{N}(0, 1)$ representing additive Gaussian noise. For the b variable, we apply a linear kernel, hence the coefficients $\widetilde{\alpha}$ related to the kernel $F(a)$ can be obtained by solving a least-square problem. We can notice that the coefficients illustrated in

Figure 5.1(a) do not match the weights defined in Equation (5.3); this is because the least-square method finds one of the many possible equivalently valid solutions. In contrast, Figure 5.1(b) does not suffer from this issue because we removed the first column from the kernel, thus ensuring linearly independent columns. Here, the returned coefficients are proportional to the original weights $\begin{bmatrix}0 & -1 & 1 & 5\end{bmatrix}$, but we underline that this is just a coincidence, as the choice of which column to omit is arbitrary; indeed, excluding the fourth column instead of the first results in entirely different coefficients, as depicted in Figure 5.1(c). We also highlight that the correlation ρ reported is identical in all three scenarios, indicating that this is strictly an interpretability problem and not an evaluation one.

5.2 Multi-Variate & Conditional Correlation

The indicators discussed in Chapters 3 and 4 are limited to computing correlations on bivariate datasets only. However, the possibility to take multiple variables into account is not only advantageous, but also essential for a broad array of tasks that require assessing correlations. Among all, we mention two interesting scenarios: in the first, the objective is to evaluate the correlation between multiple protected input attributes and the output target in order to evaluate or enforce intersectional fairness; in the second, conditional correlation between two variables has to be measured in order to extract causal knowledge from data.

Intersectional Fairness

The concept of "intersectionality" was first introduced in [Crenshaw, 2013] to identify unfair policies that result in discrimination against individuals who are part of multiple disadvantage groups. Crenshaw referred to the *DeGraffenreid v. General Motors*[1] American court case as an example, where African-American women claimed to be systematically excluded from employment due to discrimination. However, the court did not find any discriminatory practice in this case, as the company met its quotas for black and female workers by exclusively hiring black men for warehouse roles and white women for secretarial positions.

[1] `https://supreme.justia.com/cases/federal/us/401/424/`

Let us consider a scenario involving m protected attributes $Z_1, \ldots, Z_m$. From a statistical perspective, addressing intersectionality requires not only to guarantee independence $Z_i \perp\!\!\!\perp \hat{Y}, \forall i \in \{1, \ldots, m\}$, but also to ensure the independence of the combined distribution of all protected attributes: $Z_1, \ldots, Z_m \perp\!\!\!\perp \hat{Y}$. As noted in [Gohar and Cheng, 2023], most approaches for achieving intersectional fairness in machine learning simply create a joint protected attribute $Z = Z_1 \otimes \ldots \otimes Z_m$ including every possible combination of values from each Z_i. This method, however, is highly impractical as the combinatorial explosion could potentially result in subgroups too small to yield meaningful results. Furthermore, if continuous attributes need to be involved as well, the complexity would increase dramatically, making it even more challenging to effectively bin the data.

A potential solution to this could be to broaden the definition of HGR (and similarly, GeDI) in order to incorporate multiple input variables. Practically, this involves applying a copula transformation $f : \mathbb{R}^m \to \mathbb{R}$, resulting in:

$$\mathrm{HGR}(z_1, \ldots, z_m, \hat{y}) = \max_{f,g} \rho(f(z_1, \ldots, z_m), g(\hat{y})) \tag{5.4}$$

We underline that our polynomial expansion method may be impractical here, as the number of columns would significantly increase to account for all multiplicative effects among input variables; similarly, using one-hot kernels for categorical variables would lead to the same combinatorial complexity described above. Instead, neural models may offer a better capability to synthesize information in this scenario, or even lattice models could be considered as an alternative, given that they are less susceptible to overfitting due to their higher smoothness.

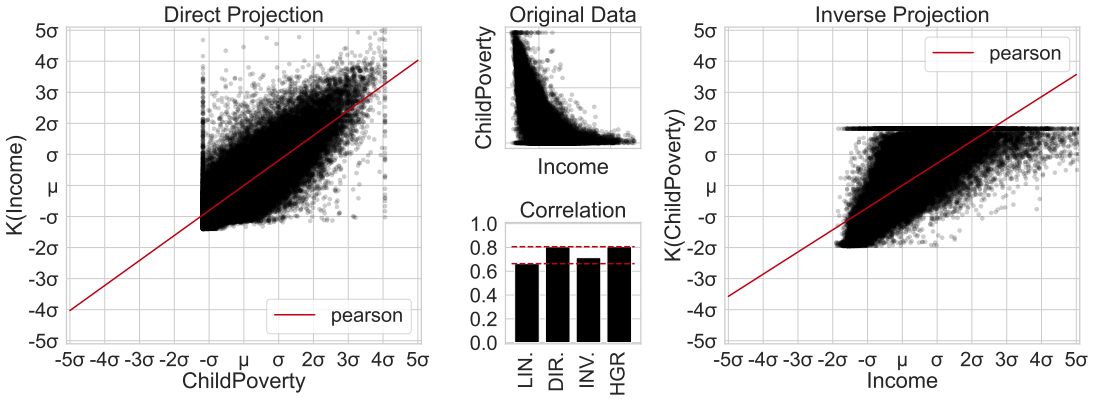
Causal Discovery

The task of discovering causal relationships among variables is gaining prominence in the literature, as evident from the increasing number of surveys on the subject [Nogueira et al., 2022]. The connection between statistical dependence and causality can be summarized by Reichenbach's Principle of the Common Cause, stating that if two variables X and Y are statistically dependent, then there must be a variable Z that accounts for the entire dependency by causally affecting

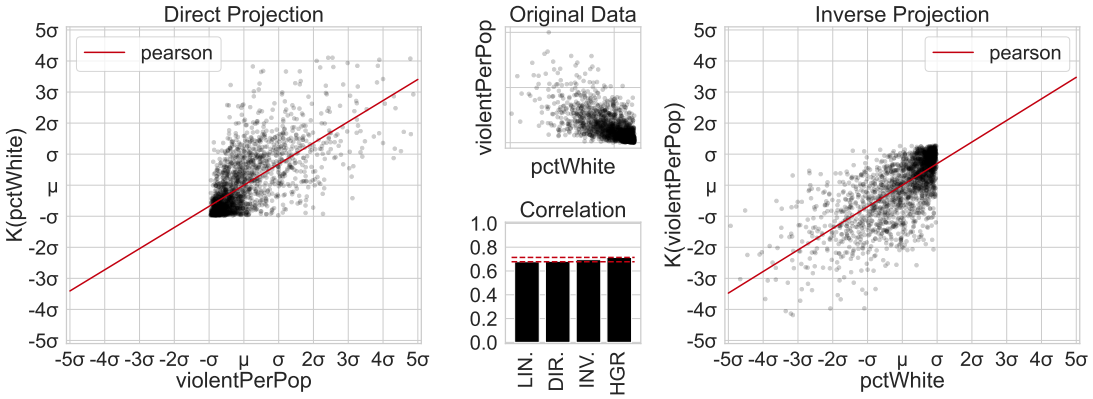
both. In particular, Z might be either identical to X or Y – thus indicating a direct causal link between the two – or it can be a third variable, referred to as "confounder", which leads to the causal structure $X \leftarrow Z \rightarrow Y$ and implies conditional independence between X and Y given Z [Schölkopf et al., 2021].

Several algorithms for causal discovery have been developed in the past decades. Among them, the most famous is the PC Algorithm [Spirtes and Glymour, 1991]. Starting from a dense graph containing connections between all variables, the algorithm employs conditional independence tests to determine if two variables X and Y are independent when conditioned on a set $S = \{Z_1, \ldots, Z_m\}$. As noted in Section 3.1, a significant advantage of HGR over other correlation metrics is its sufficiency to guarantee independence, which makes it a good candidate for independence tests. However, its semantics do not inherently account for the conditioning set, thus requiring ad hoc modifications. The HGR-KDE technique from [Mary et al., 2019] is already equipped for this, as it estimates probability distributions; specifically, when experimenting with HGR as a proxy for Equalized Odds, the authors use the formulation HGR-KDE$(Z \mid Y = y, \hat{Y} \mid Y = y)$, which requires conditional independence with respect to the ground truths Y. Nevertheless, extending this concept to functional approaches such as ours or the neural-based one presents additional challenges.

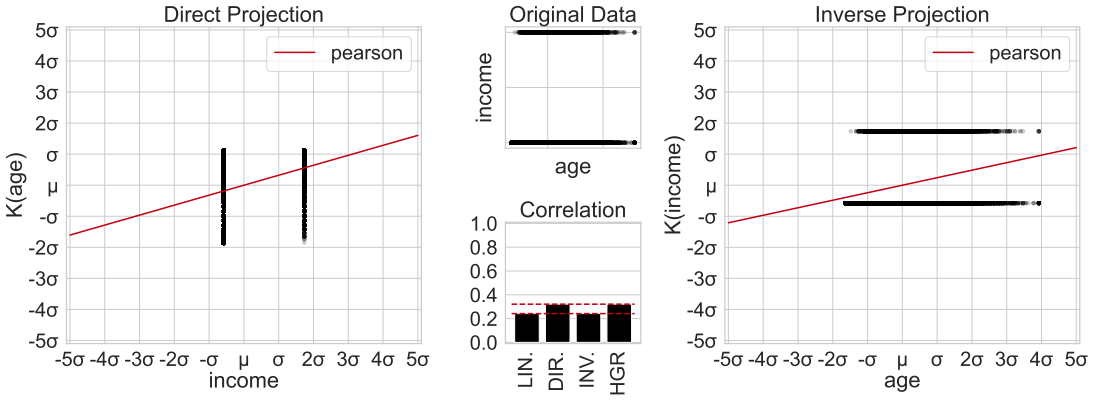
We also claim that HGR offers another notable advantage in the realm of causal discovery. Besides graph representations, causal structures can be also expressed in terms of functions: this implies that if there exists a causal link $X \mapsto Y$, then there must be a function f such that $Y = f(X, \ldots)$, where the ellipsis suggests the presence of potential additional factors. With its dual copula transformations, HGR facilitates the estimation of these functional effects while testing for causal dependencies. Specifically, for a given kernel degree d, we can evaluate the difference between $\mathrm{HGR}(X, Y; d, 1)$ and $\mathrm{HGR}(X, Y; 1, d)$ and assess which variable more effectively predicts the other through a non-linear transformation. Furthermore, we can compare these metrics to both linear (Pearson's) and non-linear (HGR) correlations to determine how much the discrepancy between the functional dependencies can be attributed to the variables themselves or to external confounders. Figure 5.2 provides an example of this approach, where we analyze the



(a) *Census*



(b) *Communities*



(c) *Adult*

Figure 5.2: Analysis of causal relationships in bivariate datasets using HGR.

differences between linear and non-linear relationships on the continuous protected attribute z and the output target y across the three benchmark datasets used in our study. For each dataset, we calculated: the linear correlation $\mathrm{HGR}(z, y; 1, 1)$, the "direct" correlation $\mathrm{HGR}(z, y; 5, 1)$, the "inverse" correlation $\mathrm{HGR}(z, y; 1, 5)$, and the full non-linear correlation $\mathrm{HGR}(z, y; 5, 5)$. For instance, in the *Census* and *Adult* datasets, we can notice that nearly all non-linear correlations are captured by the "direct" dependency, while for *Communities*, the minor differences in computed correlations indicate either a strong co-dependency between the variables or an underlying confounding factor. It is important to note that a functional relationship is not sufficient to establish or determine causal direction, but it can still provide a substantial hint in ambiguous situations.

5.3 Redefining Fairness

In Chapter 4, we explored how to use correlation measures to enforce fairness constraints in machine learning models. However, this approach is purely statistical and solely depends on the available data, making it susceptible to noise and complicating the selection of the threshold τ. In our earlier research [Maggio et al., 2023], we demonstrated that task accuracy might not be a reliable indicator for fairness-related applications, as historical biases in datasets cause distribution shifts that could make ground truths particularly unreliable in certain instances. Our future proposal involves the construction of alternative versions of all inputs so that they are equally informative but independent of the protected attribute. This approach can also be considered as a way to fully anonymize the sensitive information.

Latent Independent Variables

Let us consider a scenario involving two variables, x and y. Suppose that our goal is to anonymize x while maintaining predictive capabilities with respect to y. To accomplish this, we might construct a latent variable w, obtained as:

$$\arg\min_{w}\{H(y, w) \mid H(x, w) = \max_{w'} H(x, w')\} \tag{5.5}$$

In this context, H represents the entropy, and our objective is to construct w in such a way that it carries no information about x while simultaneously minimizing uncertainty with respect to y. We start by observing that the entropy reaches its peak when $x \perp\!\!\!\perp w$, and achieves its lowest value when the correlation between y and w is at its maximum. Using HGR as both a measure of correlation and a test of independence up to a threshold ε, we can reformulate Equation (5.5) as:

$$\arg\max_{w}\{\text{HGR}(y, w) \mid \text{HGR}(x, w) \leq \varepsilon\} \tag{5.6}$$

Fairness Through Decorrelation

Equation (5.6) is particularly challenging to resolve. However, assuming that we can achieve a solution within a feasible time, we could apply it to the dataset $\mathcal{D} = \{X, z, y\}$ to fully remove the correlation between the inputs $x_i \in X$ and the protected attribute z. Practically, our goal is to build the decorrelated variables:

$$x'_i \in \arg\max_{w}\{\text{HGR}(x_i, w) \mid \text{HGR}(z, w) \leq \varepsilon\} \tag{5.7}$$

and eventually train our model on a transformed dataset $\mathcal{D}' = \{X', y\}$, where X' is the matrix with all the x'_i columns. We also mention that computing these new variables alone is not sufficient, as it would prevent from executing similar operations on the test data. To address this issue, instead of retrieving x'_i through projections, we would rather calculate them using a learning model $\mathcal{M}(\cdot; \theta_i)$. Consequently, we reformulate Equation (5.7) as:

$$\begin{aligned} &x'_i = \mathcal{M}(x_i, \theta^*_i) \quad \text{s.t.} \\ &\theta^*_i \in \arg\max_{\theta_i}\{\text{HGR}(x_i, \mathcal{M}(x_i; \theta_i)) \mid \text{HGR}(z, \mathcal{M}(x_i; \theta_i)) \leq \varepsilon\} \end{aligned} \tag{5.8}$$

Although this concept may offer a new definition of fairness, more rooted into a causal framework rather than merely based on statistical independence from the attribute, we recognize the significant challenge it presents and have yet to conduct experiments to determine its practical applicability.

Chapter 6

Conclusions

In this dissertation, we introduced a novel computational approach for identifying non-linear correlations in bivariate datasets. Our technique is based on polynomial kernel functions parametrized by a vector of mixing coefficients. Since polynomials are universal approximators, our method can theoretically capture any form of dependency, and it also distinguishes itself from existing methodologies proposed in the literature thanks to its stronger robustness, interpretability, and configurability.

We applied our methodology to the task of estimating the Hirschfeld–Gebelein–Rényi (HGR) coefficient, a theoretical extension of Pearson's correlation capable of detecting non-linear relationships as well. Our experiments validated the effectiveness of the approach, as well as showing that an approximation can be used to gain two orders of magnitude in computational time at the cost of lower expressivity.

Next, we tested the behavior of our Kernel-Based HGR indicator when used as a fairness measure. We imposed constraints on three benchmark datasets for fair machine learning, dealing with continuous rather than categorical protected attributes. Although the computation of our method is not entirely differentiable, it still demonstrated the ability to guide the learning process towards fair solutions thanks to the information embedded in its sub-gradient.

In discussing fairness with continuous protected attributes, we also noted some limitations in the semantics of HGR, which led to the development of the Generalized Disparate Impact (GeDI), a novel indicator extending the legal notion of disparate impact to the continuous domain. We proposed an implementation of

this indicator using the same kernel-based strategy adopted for HGR, and eventually performed further empirical evaluations to determine if its enforcement could yield the expected results. In particular, we used it to inject fairness constraints both via loss regularizers and in a declarative fashion using a projection-based algorithm for constrained machine learning. Finally, we also reported the results of another empirical study that we conducted on a real-world dataset about child education.

To conclude, we discussed that HGR and GeDI represent the two ends of a wider spectrum of fairness indicators, which can be derived with minor variations in their semantics. Moreover, as our objective is to continue exploring this computational method and the indicators it can generate, we highlighted some open research areas that have been partially explored but have not yet produced definitive results. Among these, we mentioned the application of different kernels, such as using one-hot encodings for categorical data or sinusoidal expansions for continuous data, as well as the application of lattice models to achieve a better balance between computational cost and expressiveness when addressing multivariate correlations in tasks like intersectional fairness or causal discovery; finally, we introduced a proof-of-concept for a new approach to eliminating undesirable biases in data by removing unwanted correlations. We hope that our work will inspire further research in this field, leading to more reliable and trustworthy data-driven applications.

Bibliography

[Aghaei et al., 2019] Aghaei, S., Azizi, M. J., and Vayanos, P. (2019). 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*, volume 33, pages 1418–1426. AAAI Press.

[Angwin et al., 2022] Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2022). Machine bias. In *Ethics of data and analytics*, pages 254–264. Auerbach Publications.

[Avvisati, 2020] Avvisati, F. (2020). The measure of socio-economic status in pisa: a review and some suggested improvements. *Large-scale Assessments in Education*, 8(1).

[Bach and Jordan, 2003] Bach, F. and Jordan, M. (2003). Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48.

[Badreddine et al., 2022] Badreddine, S., d'Avila Garcez, A., Serafini, L., and Spranger, M. (2022). Logic tensor networks. *Artificial Intelligence*, 303:103649.

[Baharlouei et al., 2019] Baharlouei, S., Nouiehed, M., Beirami, A., and Razaviyayn, M. (2019). Rényi fair inference. *arXiv preprint arXiv:1906.12005*.

[Biega et al., 2018] Biega, A. J., Gummadi, K. P., and Weikum, G. (2018). Equity of attention: Amortizing individual fairness in rankings. In *The 41st international*

acm sigir conference on research & development in information retrieval, pages 405–414.

[Breiman and Friedman, 1985] Breiman, L. and Friedman, J. H. (1985). Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80(391):580–598.

[Calders and Verwer, 2010] Calders, T. and Verwer, S. (2010). Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292.

[Calmon et al., 2017] Calmon, F., Wei, D., Vinzamuri, B., Natesan Ramamurthy, K., and Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Caton and Haas, 2020] Caton, S. and Haas, C. (2020). Fairness in machine learning: A survey. *ACM Computing Surveys*.

[Celis et al., 2020] Celis, L. E., Keswani, V., and Vishnoi, N. (2020). Data preprocessing to mitigate bias: A maximum entropy based approach. In *International conference on machine learning*, pages 1349–1359. PMLR.

[Conn et al., 2000] Conn, A. R., Gould, N. I. M., and Toint, P. L. (2000). *Trust Region Methods*. MOS-SIAM Series on Optimization. SIAM.

[Cotter et al., 2019] Cotter, A., Jiang, H., Gupta, M., Wang, S., Narayan, T., You, S., and Sridharan, K. (2019). Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, 20(172):1–59.

[Crenshaw, 2013] Crenshaw, K. (2013). Demarginalizing the intersection of race and sex: A black feminist critique of antidiscrimination doctrine, feminist theory and antiracist politics. In *Feminist legal theories*, pages 23–51. Routledge.

[D'Alessandro et al., 2017] D'Alessandro, B., O'Neil, C., and LaGatta, T. (2017). Conscientious classification: A data scientist's guide to discrimination-aware classification. *Big Data*, 5(2):120–134.

[Detassis et al., 2021] Detassis, F., Lombardi, M., and Milano, M. (2021). Teaching the old dog new tricks: Supervised learning with constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):3742–3749.

[Diligenti et al., 2017] Diligenti, M., Gori, M., and Saccà, C. (2017). Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165.

[Donini et al., 2018] Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J., and Pontil, M. (2018). Empirical risk minimization under fairness constraints. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2796–2806. Curran Associates Inc.

[Dwork et al., 2012] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. S. (2012). Fairness through awareness. In Goldwasser, S., editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 214–226. ACM.

[Fabris et al., 2023] Fabris, A., Baranowska, N., Dennis, M. J., Graus, D., Hacker, P., Saldivar, J., Zuiderveen Borgesius, F., and Biega, A. J. (2023). Fairness and bias in algorithmic hiring: a multidisciplinary survey. *ACM Transactions on Intelligent Systems and Technology*.

[Feldman et al., 2015] Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In Cao, L., Zhang, C., Joachims, T., Webb, G. I., Margineantu, D. D., and Williams, G., editors, *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 259–268. ACM.

[Fioretto et al., 2021] Fioretto, F., Hentenryck, P. V., Mak, T. W. K., Tran, C., Baldo, F., and Lombardi, M. (2021). Lagrangian duality for constrained deep

learning. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track*, pages 118–135. Springer International Publishing.

[Fish et al., 2016] Fish, B., Kun, J., and Lelkes, Á. D. (2016). A confidence-based approach for balancing fairness and accuracy. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics.

[Garcia and Gupta, 2009] Garcia, E. and Gupta, M. (2009). Lattice regression. *Advances in Neural Information Processing Systems*, 22:594–602.

[Ge et al., 2021] Ge, Y., Liu, S., Gao, R., Xian, Y., Li, Y., Zhao, X., Pei, C., Sun, F., Ge, J., Ou, W., et al. (2021). Towards long-term fairness in recommendation. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 445–453.

[Giuliani et al., 2024] Giuliani, L., Misino, E., Calegari, R., and Lombardi, M. (2024). Long-term fairness strategies in ranking with continuous sensitive attributes. In *Proceedings of the 2nd Workshop on Fairness and Bias in AI, AEQUITAS 2024 co-located with 27th European Conference on Artificial Intelligence (ECAI 2024)*. CEUR-WS.

[Giuliani et al., 2023] Giuliani, L., Misino, E., and Lombardi, M. (2023). Generalized disparate impact for configurable fairness solutions in ML. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11443–11458. PMLR.

[Goh et al., 2016] Goh, G., Cotter, A., Gupta, M., and Friedlander, M. P. (2016). Satisfying real-world goals with dataset constraints. *Advances in neural information processing systems*, 29.

[Gohar and Cheng, 2023] Gohar, U. and Cheng, L. (2023). A survey on intersectional fairness in machine learning: Notions, mitigation, and challenges. *arXiv preprint arXiv:2305.06969*.

[Grari et al., 2020] Grari, V., Lamprier, S., and Detyniecki, M. (2020). Fairness-aware neural rényi minimization for continuous features. In Bessiere, C., editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2262–2268. International Joint Conferences on Artificial Intelligence Organization.

[Greco et al., 2023] Greco, G., Alberici, F., Palmonari, M., Cosentini, A., et al. (2023). Declarative encoding of fairness in logic tensor networks. *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 372:908–915.

[Gretton et al., 2005] Gretton, A., Herbrich, R., Smola, A., Bousquet, O., and Schölkopf, B. (2005). Kernel methods for measuring independence. *J. Mach. Learn. Res.*, 6:2075–2129.

[Hardoon and Shawe-Taylor, 2008] Hardoon, D. R. and Shawe-Taylor, J. (2008). Convergence analysis of kernel canonical correlation analysis: theory and practice. *Machine Learning*, 74(1):23–38.

[Hardt et al., 2016] Hardt, M., Price, E., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29 of *NIPS'16*, page 3323–3331. Curran Associates, Inc.

[High-Level Expert Group on AI, 2019] High-Level Expert Group on AI (2019). Ethics guidelines for trustworthy AI. Technical report, European Commission.

[Hutchinson and Mitchell, 2019] Hutchinson, B. and Mitchell, M. (2019). 50 years of test (un) fairness: Lessons for machine learning. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 49–58.

[James et al., 2023] James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An introduction to statistical learning: With applications in python*. Springer Nature.

[Jiang et al., 2022] Jiang, Z., Han, X., Fan, C., Yang, F., Mostafavi, A., and Hu, X. (2022). Generalized demographic parity for group fairness. In *International Conference on Learning Representations*.

[Kamiran and Calders, 2011] Kamiran, F. and Calders, T. (2011). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33.

[Kamiran et al., 2010] Kamiran, F., Calders, T., and Pechenizkiy, M. (2010). Discrimination aware decision tree learning. In *2010 IEEE International Conference on Data Mining*. IEEE.

[Kendall, 1948] Kendall, M. G. (1948). *Rank correlation methods.* Griffin.

[Kizilcec and Lee, 2022] Kizilcec, R. F. and Lee, H. (2022). Algorithmic fairness in education. In *The ethics of artificial intelligence in education*, pages 174–202. Routledge.

[Komiyama et al., 2018] Komiyama, J., Takeda, A., Honda, J., and Shimao, H. (2018). Nonconvex optimization for regression with fairness constraints. In *International conference on machine learning*, pages 2737–2746. PMLR.

[Locke et al., 2021] Locke, D. H., Hall, B., Grove, J. M., Pickett, S. T., Ogden, L. A., Aoki, C., Boone, C. G., and O'Neil-Dunne, J. P. (2021). Residential housing segregation and urban tree canopy in 37 us cities. *NPJ Urban Sustainability*, 1(1):15.

[Lombardi et al., 2021] Lombardi, M., Baldo, F., Borghesi, A., and Milano, M. (2021). An analysis of regularized approaches for constrained machine learning. In *Trustworthy AI-Integrating Learning, Optimization and Reasoning: First International Workshop, TAILOR 2020, Virtual Event, September 4–5, 2020, Revised Selected Papers 1*, pages 112–119.

[Lopez-Paz et al., 2013] Lopez-Paz, D., Hennig, P., and Schölkopf, B. (2013). The randomized dependence coefficient. *Advances in neural information processing systems*, 26.

[Luong et al., 2011] Luong, B. T., Ruggieri, S., and Turini, F. (2011). k-NN as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11.* ACM Press.

[Madiega, 2021] Madiega, T. (2021). Artificial intelligence act. *European Parliament: European Parliamentary Research Service*.

[Maggio et al., 2023] Maggio, A., Giuliani, L., Calegari, R., Lombardi, M., and Milano, M. (2023). A geometric framework for fairness. In *Proceedings of the 1st Workshop on Fairness and Bias in AI, AEQUITAS 2023 co-located with 26th European Conference on Artificial Intelligence (ECAI 2023)*, volume 3523, pages 1–17. CEUR-WS.

[Manhaeve et al., 2018] Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. (2018). Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31:3749–3759.

[Mary et al., 2019] Mary, J., Calauzènes, C., and Karoui, N. E. (2019). Fairness-aware learning for continuous attributes and treatments. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4382–4391. PMLR.

[Mehrabi et al., 2021] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM computing surveys*, 54(6):1–35.

[Misino et al., 2023] Misino, E., Calegari, R., Lombardi, M., and Milano, M. (2023). Fairdas: Fairness-aware ranking as dynamic abstract system. In Calegari, R., Tubella, A. A., González-Castañé, G., Dignum, V., and Milano, M., editors, *Proceedings of the 1st Workshop on Fairness and Bias in AI co-located with 26th European Conference on Artificial Intelligence (ECAI 2023), Kraków, Poland, October 1st, 2023*, volume 3523 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Morley et al., 2021] Morley, J., Kinsey, L., Elhalal, A., Garcia, F., Ziosi, M., and Floridi, L. (2021). Operationalising ai ethics: barriers, enablers and next steps. *AI Soc.*, 38(1):411–423.

[Nogueira et al., 2022] Nogueira, A. R., Pugnana, A., Ruggieri, S., Pedreschi, D., and Gama, J. (2022). Methods and tools for causal discovery and causal

inference. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 12(2):e1449.

[Padala and Gujar, 2020] Padala, M. and Gujar, S. (2020). Fnnc: Achieving fairness through neural networks. In Bessiere, C., editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2277–2283. International Joint Conferences on Artificial Intelligence Organization.

[Póczos et al., 2012] Póczos, B., Ghahramani, Z., and Schneider, J. (2012). Copula-based kernel dependency measures. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1635–1642. Omnipress.

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

[Schölkopf et al., 2021] Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634.

[Singh and Joachims, 2018] Singh, A. and Joachims, T. (2018). Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2219–2228.

[Spirtes and Glymour, 1991] Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72.

[Srivastava et al., 2019] Srivastava, M., Heidari, H., and Krause, A. (2019). Mathematical notions vs. human perception of fairness: A descriptive approach to fairness for machine learning. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2459–2468.

[Székely and Rizzo, 2009] Székely, G. J. and Rizzo, M. L. (2009). Brownian distance covariance. *The Annals of Applied Statistics*, 3(4).

[Ueda et al., 2024] Ueda, D., Kakinuma, T., Fujita, S., Kamagata, K., Fushimi, Y., Ito, R., Matsui, Y., Nozaki, T., Nakaura, T., Fujima, N., et al. (2024). Fairness of artificial intelligence in healthcare: review and recommendations. *Japanese Journal of Radiology*, 42(1):3–15.

[Von Rueden et al., 2021] Von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Walczak, M., Pfrommer, J., Pick, A., Ramamurthy, R., Garcke, J., Bauckhage, C., and Schuecker, J. (2021). Informed machine learning - a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1.

[Wang and Gupta, 2020] Wang, S. and Gupta, M. (2020). Deontological ethics by monotonicity shape constraints. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2043–2054. PMLR, PMLR.

[Witsenhausen, 1975] Witsenhausen, H. S. (1975). On sequences of pairs of dependent random variables. *SIAM Journal on Applied Mathematics*, 28(1):100–113.

[Woodworth et al., 2017] Woodworth, B. E., Gunasekar, S., Ohannessian, M. I., and Srebro, N. (2017). Learning non-discriminatory predictors. In Kale, S. and Shamir, O., editors, *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, volume 65 of *Proceedings of Machine Learning Research*, pages 1920–1953. PMLR.

[Xian et al., 2023] Xian, R., Yin, L., and Zhao, H. (2023). Fair and optimal classification via post-processing. In *International Conference on Machine Learning*, pages 37977–38012. PMLR.

[Yin et al., 2024] Yin, T., Raab, R., Liu, M., and Liu, Y. (2024). Long-term fairness with unknown dynamics. *Advances in Neural Information Processing Systems*, 36.

[You et al., 2017] You, S., Ding, D., Canini, K., Pfeifer, J., and Gupta, M. (2017). Deep lattice networks and partial monotonic functions. *arXiv preprint arXiv:1709.06680*.

[Zafar et al., 2017] Zafar, M. B., Valera, I., Rodriguez, M. G., and Gummadi, K. P. (2017). Fairness beyond disparate treatment & disparate impact. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee.

[Zehlike et al., 2017] Zehlike, M., Bonchi, F., Castillo, C., Hajian, S., Megahed, M., and Baeza-Yates, R. (2017). Fa*ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1569–1578.

[Zehlike et al., 2022a] Zehlike, M., Yang, K., and Stoyanovich, J. (2022a). Fairness in ranking, part i: Score-based ranking. *ACM Comput. Surv.*, 55(6).

[Zehlike et al., 2022b] Zehlike, M., Yang, K., and Stoyanovich, J. (2022b). Fairness in ranking, part ii: Learning-to-rank and recommender systems. *ACM Comput. Surv.*, 55(6).

[Zemel et al., 2013] Zemel, R., Wu, Y., Swersky, K., Pitassi, T., and Dwork, C. (2013). Learning fair representations. In *International conference on machine learning*, pages 325–333. PMLR.

Appendix A

Shared Experimental Details

In this appendix, we provide a detailed examination of the common elements of all our experiments. The codebase used to run these experiments is publicly available under the MIT license and can be accessed at `https://github.com/giuluck/non-linear-correlations`. Unless noted otherwise, we used the official code for the baseline approaches without changing the default parameters, and we incorporated them into our project to ensure the reproducibility of the results.

A.1 Hardware & Software Setup

We implemented our code using Python 3.12 and leveraged the following packages:

```
gurobipy==11.0.3
lightning==2.3.3
matplotlib==3.9.0
moving_targets==0.4.0
numpy==2.0.1
pandas==2.2.2
scikit-learn==1.5.1
scipy==1.14.0
seaborn==0.13.2
torch==2.4.0
tqdm==4.66.5
```

We ran all our experiments on a MacBook Pro equipped with an Apple M3 Pro processor and 36GB RAM. We did not use Graphics Processing Units (GPUs) in our experiments. Prior to each execution, our code automatically initializes specific seeds for all random number operations using the `seed_everything` method from Pytorch Lightning and use the `deterministic=True` training option to ensure the complete reproducibility of our results and enforce deterministic outcomes even in neural network training and inference.

Unless an external source is specified, every figure in this document is generated by a respective script available in our code repository. To facilitate the execution process, we also provided a custom Docker image along with a configuration file (`docker.yaml`) that leverages Docker Compose to set up a service for each script. Additional details on how to run the code are provided in the `README.md` file.

A.2 Benchmark Datasets

Throughout our experiments with real-world datasets, we leverage data from three well-established benchmarks in fair machine learning:

Census The *US 2015 Census* dataset[1] contains records from the 2015 American Community Survey 5-year Estimates. Among the two available files, we only used `acs2015_census_tract_data.csv`, which comprises data for each census tract in the United States, namely areas designated by the bureau to have a more uniform size – typically around 5000 residents. The predictive variables include aggregated socioeconomic information about residents of the census track, and the primary objective is to forecast the `ChildPoverty` rate within each of them, which is represented by a continuous variable.

Communities The *Communities & Crime* dataset[2] integrates socio-economic and survey law enforcement data with FBI crime statistics from the 1990s. As before, the predictive variables range from demographic segmentation in social groups to other economic indicators aggregated over the population of

[1] `https://www.kaggle.com/datasets/muonneutrino/us-census-demographic-data`
[2] `https://archive.ics.uci.edu/dataset/183/communities+and+crime`

Dataset	*Census*	*Communities*	*Adult*
Output Target	`ChildPoverty`	`violentPerPop`	`income`
Protected Attribute	`Income`	`pctWhite`	`age`
Surrogate Attribute	`Unemployement`	`race`	`Never-married`

Table A.1: Salient input and output features of the three benchmarks.

the whole neighborhood. The aim is to determine the incidence rate of violent crimes per capita, hence the target variable `ViolentPerPop` is continuous.

Adult The *Adult Census Income* dataset[3] derives from the 1994 US Census database. Unlike the previous ones, each entry here represents a single individual, with their status defined by economic indicators and association to certain social categories. Since the goal is to predict whether an individual earns more than 50K annually, the target variable `Income` is binary.

For each dataset, we preprocess the data using standard data cleaning procedures. In particular, we eliminate duplicate features – e.g., selecting either `Men` or `Women` in the `Census` dataset – and we remove highly correlated demographic features that should not be present during testing – e.g., `Poverty`, which correlates with the target variable `ChildPoverty` in `Census`. Subsequently, we:

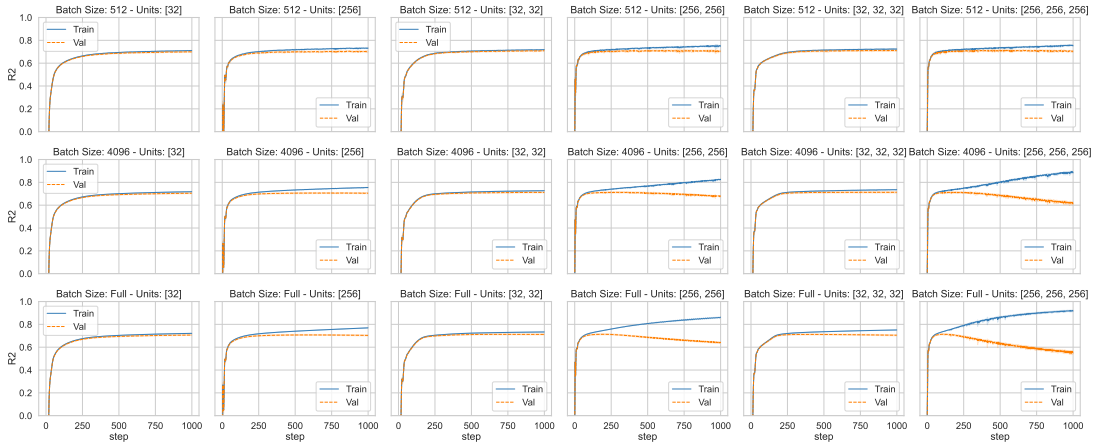
1. Normalize the target variable within the interval $[0, 1]$.
2. One-hot encode all the multi-class categorical inputs.
3. Standardize all the continuous input variables.

Finally, both the continuous protected attributes z and their binary surrogate s are selected according to the procedure described in Section 4.1.1. An overview of them, which will be used during training experiments, is presented in Table A.1.

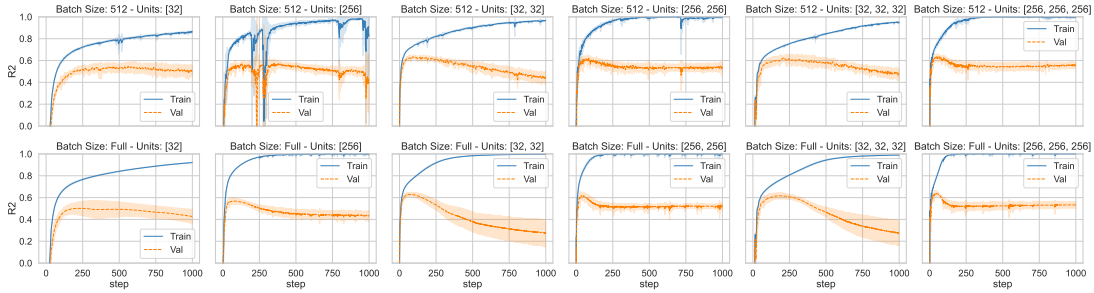
A.3 Unconstrained Neural Network Calibration

Given that all our experiments use neural networks as learning models, we also perform a calibration phase to identify the optimal hyperparameters configuration.

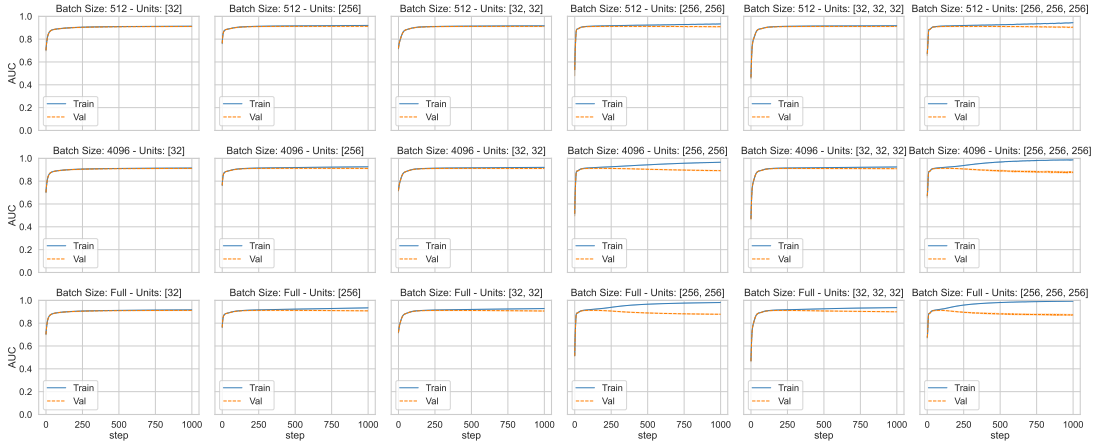
[3] `https://archive.ics.uci.edu/dataset/2/adult`



(a) *Census*



(b) *Communities*



(c) *Adult*

Figure A.1: Neural networks scores across the three benchmarks.

Dataset	*Census*	*Communities*	*Adult*
Loss Function	MSE	MSE	BCE
Score Function	R^2	R^2	AUC
Hidden Units	[32]	[256, 256]	[32, 32]
Batch Size	2048	Full-Batch	2048
Steps	500	500	500

Table A.2: Hyperparameter configuration of the baseline neural networks.

The results obtained in this phase are meant to validate the selection of architectures for the benchmark experiments discussed in Chapter 4. We train several neural networks using a default task loss – Mean Squared Error (MSE) for regression tasks, Binary Crossentropy (BCE) for classification ones. For each dataset, we use a 5-fold cross-validation procedure and test the following set of parameters using a plain grid search methodology:

Batch Size 512, 4096, or Full-Batch.

Number of Layers 1, 2, or 3.

Units per Layer 32 or 256.

For each configuration, we use the Adam optimizer with a fixed learning rate $\text{lr} = 10^{-3}$, and set ReLU activations in all hidden layers while the output layer uses either linear or sigmoid activation depending on the task loss. To ensure comparable results and consistent training times across different batch sizes, we use training steps instead of training epochs.

Training histories are reported in Figure A.1. We report elapsed time, loss function, and score function after every training step on both the training and validation splits. More precisely, score functions are chosen as R^2 score and Area Under Curve (AUC), respectively for regression and classification tasks. Each network undergoes 1000 training steps and, ultimately, we select the lightest configuration among those achieving the best validation accuracies. Finally, we determine the optimal value at the point where the validation accuracy converges to a consistent result. We summarize our choices in Table A.2 and pair them with the respective loss and score functions of the datasets.

A.4 Implementation Details

This section provides an in-depth explanation of the implementation details for the metrics we developed, the baselines we tested, and the techniques we employed to inject constraints into machine learning models.

A.4.1 Correlation Metrics

In this section, we detail the fundamental configuration of all the methods discussed in Chapter 3 to estimate the Hirschfeld–Gebelein–Rényi Coefficient, as well as those for calculating the Generalized Disparate Impact introduced in Section 4.3. Where needed, we also provide additional information about tailor-made choices aimed at enhancing efficiency in the context of constrained machine learning.

HGR-KB

The implementation of HGR-KB leverages the `trust-constr` optimization method available in the `scipy.optimize.minimize` package. We provide a symbolic representation of the objective function and the non-linear constraint, together with their gradients and Hessian matrices. The initial values are assigned to $\alpha_0 = \mathbf{1}_h \cdot \sigma(\mathbf{P}_a^h \cdot \mathbf{1}_h)^{-1}$ and $\beta_0 = \mathbf{1}_k \cdot \sigma(\mathbf{P}_b^k \cdot \mathbf{1}_k)^{-1}$ so that the constraint is satisfied by default, and we use 10^{-2} as the general tolerance for the algorithm. All other hyperparameters remain unchanged, thus letting the optimization method run for up to 1000 iterations.

Before running the optimization routine, we try to remove linearly dependent columns in the concatenated matrix $\left[\mathbf{P}_a^h \mid \mathbf{P}_b^k\right]$ using the QR matrix factorization algorithm. Once the factorization values r_{ii} belonging to the diagonal of the matrix R are obtained, we eliminate all the columns associated to a factorization value smaller than the threshold 10^{-2}. This step aims to limit the overspecification of copula transformations, as our method theoretically requires linear independence between the kernel columns.

Finally, when using HGR-KB as a regularizer in the context of constrained machine learning, we "warm start" the optimization routine at iteration $i+1$ by passing the optimal coefficients α_i^* and β_i^* obtained from the previous step i as initial

guesses. Given that distribution should not deviate much across different epochs or mini-batches, this strategy facilitates quicker and more accurate convergence.

HGR-SK

Regarding HGR-SK, optimal results are computed using the least-square problem implementation of `numpy` when there are no gradients required. Instead, when the metric is used as a loss regularized, the `torch.linalg.lstsq` implementation is employed as it offers both gradient information and optimal outcomes. For the latter, we also specify the `gelsd` drivers to ensure more precise and reproducible results, albeit with a higher computational overhead. Unlike HGR-KB, here no initial guesses or supplementary hyperparameters are required.

HGR-NN

In order to implement the baseline adversarial approach, we rely on the source code[4] of [Grari et al., 2020]. We preserve the original code without altering its internal functions and incorporate it into our own codebase for ease of execution. Practically, when computing HGR-NN, we make use of two distinct networks featuring linear layers of units $[15, 15, 15, 1]$ – some with ReLU and some others with tanh activations –, and a final layer of batch normalization. The overall system is trained full-batch for 1000 epochs.

When this method is used as a loss regularizer, we follow the speed-up strategy described in the original paper. Specifically, during the first learning step we train the adversarial networks for 1000 epochs, while in the following phases we simply fine-tune them for 50 epochs using the pre-trained models from the previous iteration. Similarly to HGR-KB, this approach accelerates convergence at no performance cost, assuming that there are no significant distributional changes between epochs or mini-batches.

[4] `https://github.com/fairml-research/HGR_NN/`

HGR-KDE

For HGR-KDE, we lean on the code[5] provided in the original paper, keeping the default hyperparameters and employing the `torch` library for its implementation, as did the original authors. We integrated all components into our codebase while minimizing alterations to their original code; nevertheless, we recognize that the results presented in Figure 3.7 differ from those reported in their paper [Mary et al., 2019]. Given that we did not conduct experiments with HGR-KDE in the realm of constrained machine learning, there are no further strategies to report.

RDC

We could not find the original code from [Lopez-Paz et al., 2013]. However, since there was an implementation available in the codebase of [Grari et al., 2020], we used it as a substitute. Similarly to the other baselines, we integrated the functions into our codebase without modifying the source code.

GeDI

Finally, to implement the GeDI indicator, we adopt a methodology similar to that of HGR-SK. Specifically, we obtain the optimal coefficients $\widetilde{\alpha}^*$ by solving the least-square problem using `numpy` and subsequently compute the indicator value following Equation (4.12). Instead, when using the indicator to impose constraints, we can leverage either the declarative approach or the loss regularizer, and either the coarse-grained or the fine-grained formulation:

Loss Regularizer We rely on `torch.linalg.lstsq` to calculate the $\widetilde{\alpha}^*$ coefficients in a differentiable manner, adopting the `gelsd` drivers to achieve more accurate and consistent results as in HGR-SK. For the coarse-grained formulation, we enforce the constraint on the overall GeDI value computed according to Equation (4.12). Instead, for the fine-grained formulation we return the regularization vector $\begin{bmatrix} |\widetilde{\alpha}_1| - \tau & |\widetilde{\alpha}_2| & \dots & |\widetilde{\alpha}_h| \end{bmatrix}$, which is subsequently paired with a vector of Lagrangian multipliers $\lambda \in \mathbb{R}^{+k}$.

[5] `https://github.com/criteo-research/continuous-fairness/`

Declarative Method We use `gurobipy` to implement our optimization problems. For the coarse-grained formulation, we generate a vector of variables for the targets v and one for the coefficients $\widetilde{\alpha}$; then we impose the linear constraints representing the least-square problem and the constraint on the GeDI value as detailed in Equation (4.18). For the fine-grained formulation, instead, we simply build the vector of variables v and take advantage of the optimized formulation described in Section 4.3.4 to avoid the explicit introduction of variables for $\widetilde{\alpha}$.

The loss regularizer method is applied when training a neural model, similar to other HGR-based regularizers; while for the declarative method, we use Moving Targets as the constraint enforcement algorithm, using various machine learning models as learners.

A.4.2 Constrained Machine Learning Methods

In this section, we will briefly outline the implementation details of two methodologies we employ to enforce our constraint across various ML models. Both techniques are intended to solve the following constrained optimization problem:

$$\arg\min_{\theta} \mathcal{L}(y, \hat{y}(\theta)) \qquad \text{s.t.} \qquad \hat{y}(\theta) \in \mathcal{C} \tag{A.1}$$

where $\hat{y}(\theta) = \mathcal{M}(x; \theta)$ denotes the predictions yielded by the machine learning model $\mathcal{M}$ using the learned parameters θ. Here, $\mathcal{C}$ represents the feasible region and $\mathcal{L}$ is a loss function specific to the task. For both approaches, the original papers are referenced for further details.

Lagrangian Dual Framework

The Lagrangian Dual framework [Fioretto et al., 2021] improves the use of loss regularization techniques in gradient-based learning by enabling the automatic tuning of Lagrangian multipliers $\lambda \in \mathbb{R}^{+k}$. Consider a regularization vector $\mathcal{R}(y, \hat{y}(\theta)) \in \mathbb{R}^{+k}$ that encapsulates violations for k distinct constraints, namely the "distance" from $\hat{y}(\theta) \in \mathcal{C}$ for each constraint in $\mathcal{C}$. We can incorporate these violations into the loss

function $\mathcal{L}(y, \hat{y}(\theta)) \in \mathbb{R}^+$ of our neural network by scaling each violation with its corresponding multiplier λ_i, obtaining the following regularized loss:

$$\mathcal{L}(y, \hat{y}(\theta)) + \lambda^T \cdot \mathcal{R}(y, \hat{y}(\theta)) \tag{A.2}$$

Traditionally, a major drawback of this method is the need to fine-tune the multiplier vector λ according to the specific task. The proposed framework addresses this issue by solving a bilevel optimization problem where, first, the loss function is minimized using gradient descent with constant multipliers, and second, the loss function is maximized using gradient ascent with a fixed network configuration. Practically speaking, this process involves the following sequential steps:

$$\theta_i \in \arg\min_{\theta} \left\{ \mathcal{L}(y, \hat{y}(\theta)) + \lambda_{i-1}^T \cdot \mathcal{R}(y, \hat{y}(\theta)) \right\} \tag{A.3}$$

$$\lambda_i \in \arg\min_{\lambda} \left\{ \mathcal{L}(y, \hat{y}(\theta_i)) + \lambda^T \cdot \mathcal{R}(y, \hat{y}(\theta_i) \right\} \tag{A.4}$$

where the subscript i denotes the i^{th} training step, and λ_0 is the null vector $\mathbf{0}_k$.

Regarding our experiments, the regularizer vector $\mathcal{R}(y, \hat{y}(\theta))$ is a scalar value that quantifies the correlation using the specified indicator I, be it any HGR method or the coarse-grained formulation of GeDI, discounted by the threshold value and clipped to a zero lower bound. In practice, we have:

$$\mathcal{R}(y; \hat{y}(\theta)) = \max\{0, I(y, \hat{y}(\theta)) - \tau\} \tag{A.5}$$

The only exception arises for the fine-grained GeDI formulation, where the regularizer consists of a vector with h different components, one per coefficient $\widetilde{\alpha}_i^*$; all these components quantify a violation proportional to their absolute values, apart from the first term which is discounted by the threshold, and eventually clipped to zero. This results in:

$$\mathcal{R}(y; \hat{y}(\theta)) = \max\{\mathbf{0}_k, \left[|\widetilde{\alpha}_1^*(\theta)| - \tau, \quad |\widetilde{\alpha}_2^*(\theta)|, \quad \ldots, \quad |\widetilde{\alpha}_h^*(\theta)| \right]\} \tag{A.6}$$

In both cases, we include an additional Adam optimizer to control the progression of the multiplier vector λ, consistently setting its learning rate as $\mathrm{lr} = 10^{-3}$.

On a final note, we must highlight a minor yet significant limitation of loss regularization methods in classification tasks. The operators "arg max" and "round" are incompatible with gradient-based learning algorithms, as their gradients are null at every point. This requires relying on predicted probabilities rather than predicted class targets when computing our constraints for classification tasks. However, we use class targets when measuring correlations in the predicted data at the end of the training process, thus leading to potential differences between the expected threshold value and the actual reported value, even within the training data. The impact of this limitation can be observed in Table 4.2 as well as in Table 4.3 – although only with respect to the Lagrangian Dual entries, as the Moving Targets method does not encounter the same issue.

Moving Targets

Moving Targets [Detassis et al., 2021] is a framework aimed at solving constrained machine learning tasks using bilevel decomposition. It operates by iteratively alternating between a "learner step", responsible for training the learning model, and a "master step", which adjusts the predicted targets to fit into the feasible space while minimizing the discrepancy between both the predictions themselves and the original targets. Practically, it addresses the problem outlined in Equation (A.1) by switching between these two sub-problems:

$$v_i \in \arg\min_{v} \nabla\mathcal{L}(v, \hat{y}(\theta_{i-1})) + \gamma_i \cdot \|v - y\|_2^2 \qquad \text{s.t.} \qquad v \in \mathcal{C} \tag{A.7}$$

$$\theta_i \in \arg\min_{\theta} \mathcal{L}(v_i, \hat{y}(\theta)) \tag{A.8}$$

where the subscript i refers to the i^{th} iteration, γ_i is a coefficient used to balance the distance between the original targets y and the predictions $\hat{y}$ in the master step, and the initial parameters θ_0 are acquired by pre-training the learning model.

This algorithm is ideal for our needs for three main reasons. First, it is model-independent, enabling us to examine how our constraints operate across various learning models, each characterized by its own unique biases and limitations. Second, it is designed to handle declarative group-indicators such as GeDI, allowing to use mini-batch training in the learning if necessary, while considering the constraint as

a whole during the projection phase. Third, it can seamlessly manage classification tasks without requiring the transformation of class targets into class probabilities, which was instead a problem for the Lagrangian Dual approach.

In our experiments, we apply the formulation presented in Equations (A.7) and (A.8). This has been refined over the years, hence it differs from the original one proposed in [Detassis et al., 2021]. Nevertheless, this is the official approach implemented in the `moving-targets` package, which we are currently developing and maintaining. More specifically, to address Equation (A.8), we employ four unconstrained machine learning models. For the Linear/Logistic Regression (LM), Random Forest (RF), and Gradient Boosting (GB) models, we utilize the implementations provided by `scikit-learn`, keeping the default hyperparameters except for Logistic Regression, where we adjust `max_iter` to 10000. For the Neural Network (NN), we leverage `torch` and apply the identical hyperparameters determined in the calibration experiment outlined in Table A.2.

With respect to Equation (A.7), this is instead formulated as an optimization problem where our adjusted projections are represented by a vector of variables v. For the coarse-grained GeDI approach, we introduce an extra vector of h variables for the coefficients $\tilde{\alpha}$, whose value is obtained by enforcing both the least-square definition and the constraint on the indicator, as shown in Equation (4.31); regarding the fine-grained constraint, instead, we utilize the optimized formulation where all higher-order terms are nullified, as detailed in Equation (4.20). In both scenarios, the i^{th} value of the balancing term γ_i is assigned to the corresponding element of the harmonic series – i.e., $\gamma_i = 1/i$ – and the loss function $\mathcal{L}$ is either set to MSE or Hamming Distance, depending on whether the task is regression or classification. The solution of the optimization problem is delegated to the `Gurobi` solver using the Python APIs provided by the `gurobipy` package, with the `WorkLimit` parameter configured to 60 to restrict hardware usage rather than time to ensure reproducible results.

Appendix B

Proofs of Chapter 3

This appendix contains the proofs of theorems and properties presented in Chapter 3. These demonstrations have been taken and partially revised from [Giuliani et al., 2023] and from another research paper that is currently being reviewed at a prestigious Artificial Intelligence conference.

B.1 Pearson's Correlation as Least Squares

Let us start from the following least-square problem over standardized variables:

$$\arg\min_{r} \frac{1}{n} \left\| \frac{a-\mu_a}{\sigma_a} \cdot r \ - \frac{b-\mu_b}{\sigma_b} \right\|_2^2 \tag{B.1}$$

with μ_a, μ_b and σ_a, σ_b being the mean and standard deviations of vectors a and b, respectively. We know that the problem is convex, as it simply features a vector product between the inputs and the variable r. This means that the optimal solution can be achieved by setting the gradient with respect to r to zero, i.e.:

$$\frac{2}{n} \left(\frac{a-\mu_a}{\sigma_a} \cdot r - \frac{b-\mu_b}{\sigma_b} \right)^T \frac{a-\mu_a}{\sigma_a} = 0 \tag{B.2}$$

which, after algebraic manipulation, can be rewritten as:

$$\frac{1}{n} \frac{(a-\mu_a)^T (a-\mu_a)}{\sigma_a^2} \cdot r = \frac{1}{n} \frac{(a-\mu_a)^T (b-\mu_b)}{\sigma_a \cdot \sigma_b} \tag{B.3}$$

With further observations, we can notice that the term $\frac{1}{n}(a-\mu_a)^T(a-\mu_a)$ denotes the variance σ_a^2. We can therefore simplify the left side, thus arriving at:

$$r = \frac{1}{n}\frac{(a-\mu_a)^T(b-\mu_b)}{\sigma_a \cdot \sigma_b} \tag{B.4}$$

which corresponds in fact to the sample Pearson correlation coefficient.

B.2 Simplification to a Single-Level Problem

Consider the definition of HGR given in Equation (3.12). We can fix zero mean and unitary standard deviation without loss of generality as explained in Section 3.1, hence obtaining:

$$\max_{\substack{f,g \\ \mathbb{E}[f_a]=\mathbb{E}[g_b]=0 \\ \mathbb{E}\left[f_a^2\right]=\mathbb{E}\left[g_b^2\right]=1}} \arg\min_r \frac{1}{n}\left\|r \cdot f_a - g_b\right\|_2^2 \tag{B.5}$$

where $f_a = f(a)$ and $g_b = g(b)$ are used as aliases to improve clarity.

We introduce two additional copula transformations $p_a = p(a)$ and $q_b = q(b)$, along with their related correlation coefficient w. Assume, without loss of generality, that one transformation pair results in a lower Mean Squared Error, i.e.:

$$\frac{1}{n}\left\|r \cdot f_a - g_b\right\|_2^2 < \frac{1}{n}\left\|w \cdot p_a - q_b\right\|_2^2 \tag{B.6}$$

we can further expand these terms as follows:

$$\frac{f_a^T f_a}{n}r^2 - 2\frac{f_a^T g_b}{n}r + \frac{g_b^T g_b}{n} < \frac{p_a^T p_a}{n}w^2 - 2\frac{p_a^T q_b}{n}w + \frac{q_b^T q_b}{n} \tag{B.7}$$

Given our zero-mean assumption, all quadratic terms such as $f_a^T f_a/n$ represent sample variances $\mathbb{E}\left[f_a^2\right]$. Consequently, since under the same assumptions variances are unitary, we can simplify this inequality as:

$$r^2 - 2\frac{f_a^T g_b}{n}r + 1 < w^2 - 2\frac{p_a^T q_b}{n}w + 1 \tag{B.8}$$

We can further reduce this inequality by noting that $f_a^T g_b/n$ and $w = p_a^T q_b/n$

represent the correlation coefficients r and w, respectively. We obtain:

$$r^2 - 2r^2 + 1 < w^2 - 2w^2 + 1 \tag{B.9}$$

which leads to:

$$r^2 > w^2 \tag{B.10}$$

Given that all transformations are invertible, we can conclude that:

$$r^2 > w^2 \quad \Leftrightarrow \quad \frac{1}{n} \left\| r \cdot f_a - g_b \right\|_2^2 < \frac{1}{n} \left\| w \cdot p_a - q_b \right\|_2^2 \tag{B.11}$$

and, following the same reasoning, the statement is valid as well for both the equality and the opposite inequality. In essence, maximizing the square of the sample HGR equates to minimizing the Mean Squared Error. To maximize r^2, one needs to either maximize r or minimize $-r$. Given the flexible nature of copula transformations, the sign of r can be altered by changing the sign of either f or g. Thus, maximizing r is ultimately equivalent to maximizing r^2 in this context.

B.3 Convexity of Kernel-Based HGR

Let us take the formulation of HGR using polynomial kernels as in Equation (3.17). We can include the constraint in the objective function as a penalty term, obtaining:

$$\arg\min_{\widetilde{\alpha},\beta} \left\| \widetilde{\mathbf{P}}_a^h \cdot \widetilde{\alpha} - \widetilde{\mathbf{P}}_b^k \cdot \beta \right\|_2^2 + \lambda \cdot \left| \sigma(\widetilde{\mathbf{P}}_b^k \cdot \beta)^2 - 1 \right| \quad \text{s.t.} \quad \lambda > M \in \mathbb{R} \tag{B.12}$$

To prove that the problem is convex, we need to demonstrate that the regularizer $\left|\sigma(\widetilde{\mathbf{P}}_b^k \cdot \beta)^2 - 1\right|$ approaches zero as λ becomes sufficiently large. This requires proving that the gradient of the regularization term can exceed that of the objective function. We recall that $\mu(\widetilde{\mathbf{P}}_b^k \cdot \beta) = 0$, which allows us to calculate variances and covariances using straightforward vector products. Consequently, let us start by

dividing Equation (B.12) into two distinct functions:

$$f(\widetilde{\alpha},\beta) = \left\| \widetilde{\mathbf{P}}_a^h \cdot \widetilde{\alpha} - \widetilde{\mathbf{P}}_b^k \cdot \beta \right\|_2^2 \tag{B.13}$$

$$h(\widetilde{\alpha},\beta) = \left| \sigma(\widetilde{\mathbf{P}}_b^k \cdot \beta)^2 - 1 \right| = \left| \frac{(\widetilde{\mathbf{P}}_b^k \cdot \beta)^T \cdot (\widetilde{\mathbf{P}}_b^k \cdot \beta)}{n} - 1 \right| \tag{B.14}$$

with n being the number of samples in the data.

Our aim is to demonstrate that:

$$\exists \lambda \in \mathbb{R} \quad \text{s.t.} \quad \begin{bmatrix} \Delta\widetilde{\alpha}^T & \Delta\beta^T \end{bmatrix} \cdot \nabla h(\widetilde{\alpha},\beta) \cdot \lambda \geq \begin{bmatrix} \Delta\widetilde{\alpha}^T & \Delta\beta^T \end{bmatrix} \cdot \nabla f(\widetilde{\alpha},\beta) \tag{B.15}$$

where $\Delta\widetilde{\alpha} = \widetilde{\alpha}' - \widetilde{\alpha}$ and $\Delta\beta = \beta' - \beta$ represent infinitesimal differences. Without loss of generality, we can restrict ourselves to the equality case, resulting in:

$$\begin{bmatrix} \Delta\widetilde{\alpha}^T & \Delta\beta^T \end{bmatrix} \cdot \nabla h(\widetilde{\alpha},\beta) \cdot \lambda = \begin{bmatrix} \Delta\widetilde{\alpha}^T & \Delta\beta^T \end{bmatrix} \cdot \nabla f(\widetilde{\alpha},\beta) \tag{B.16}$$

from which we can compute λ as:

$$\lambda = \frac{\begin{bmatrix} \Delta\widetilde{\alpha}^T & \Delta\beta^T \end{bmatrix} \cdot \nabla f(\widetilde{\alpha},\beta)}{\begin{bmatrix} \Delta\widetilde{\alpha}^T & \Delta\beta^T \end{bmatrix} \cdot \nabla h(\widetilde{\alpha},\beta)} \tag{B.17}$$

Next, let us calculate the gradients of Equations (B.13) and (B.14) as follows:

$$\nabla f(\widetilde{\alpha},\beta) = 2 \cdot \begin{bmatrix} \widetilde{\mathbf{P}}_a & -\widetilde{\mathbf{P}}_b \end{bmatrix}^T \cdot \left(\widetilde{\mathbf{P}}_a \cdot \widetilde{\alpha} - \widetilde{\mathbf{P}}_b \cdot \beta \right) \tag{B.18}$$

$$\nabla h(\widetilde{\alpha},\beta) = 2 \cdot \begin{bmatrix} \mathbf{0}_h \\ \frac{\operatorname{sign}\left(\sigma(\mathbf{P}_b \cdot \beta)^2 - 1\right)}{n} \cdot \widetilde{\mathbf{P}}_b^T \cdot \widetilde{\mathbf{P}}_b \cdot \beta \end{bmatrix} \tag{B.19}$$

where the superscripts h and k have been omitted from the polynomial expansions

for improved readability. We can incorporate them into Equation (B.17) to get:

$$\begin{aligned}\lambda &= \frac{2 \cdot \begin{bmatrix}\Delta\widetilde{\alpha}^T & \Delta\beta^T\end{bmatrix} \cdot \begin{bmatrix}\widetilde{\mathbf{P}}_a & -\widetilde{\mathbf{P}}_b\end{bmatrix}^T \cdot \left(\widetilde{\mathbf{P}}_a \cdot \widetilde{\alpha} - \widetilde{\mathbf{P}}_b \cdot \beta\right)}{2 \cdot \begin{bmatrix}\Delta\widetilde{\alpha}^T & \Delta\beta^T\end{bmatrix} \cdot \begin{bmatrix}\mathbf{0}_h & \frac{\text{sign}\left(\sigma(\mathbf{P}_b \cdot \beta)^2 - 1\right)}{n} \cdot \widetilde{\mathbf{P}}_b^T \cdot \widetilde{\mathbf{P}}_b \cdot \beta\end{bmatrix}^T} \\ &= \frac{n}{\text{sign}\left(\sigma(\widetilde{\mathbf{P}}_b \cdot \beta)^2 - 1\right)} \cdot \frac{\left(\Delta\widetilde{\alpha}^T \cdot \widetilde{\mathbf{P}}_a^T - \Delta\beta^T \cdot \widetilde{\mathbf{P}}_b^T\right) \cdot \left(\widetilde{\mathbf{P}}_a \cdot \widetilde{\alpha} - \widetilde{\mathbf{P}}_b \cdot \beta\right)}{\Delta\beta^T \cdot \widetilde{\mathbf{P}}_b^T \cdot \widetilde{\mathbf{P}}_b \cdot \beta} \\ &\approx \frac{\Delta\widetilde{\alpha}^T \cdot \widetilde{\mathbf{P}}_a^T \cdot \widetilde{\mathbf{P}}_a \cdot \widetilde{\alpha}}{\Delta\beta^T \cdot \widetilde{\mathbf{P}}_b^T \cdot \widetilde{\mathbf{P}}_b \cdot \beta} - \frac{\Delta\widetilde{\alpha}^T \cdot \widetilde{\mathbf{P}}_a^T \cdot \widetilde{\mathbf{P}}_b \cdot \widetilde{\beta}}{\Delta\beta^T \cdot \widetilde{\mathbf{P}}_b^T \cdot \widetilde{\mathbf{P}}_b \cdot \beta} - \frac{\Delta\beta^T \cdot \widetilde{\mathbf{P}}_b^T \cdot \widetilde{\mathbf{P}}_a \cdot \widetilde{\alpha}}{\Delta\beta^T \cdot \widetilde{\mathbf{P}}_b^T \cdot \widetilde{\mathbf{P}}_b \cdot \beta} + 1\end{aligned}$$

where the sign operator has been omitted assuming that $\sigma(\widetilde{\mathbf{P}}_b \cdot \beta)^2 \neq 1$.

As we approach $\Delta\widetilde{\alpha}, \Delta\beta \xrightarrow{\lim} 0$, the equation converges to a real number, as both the numerators and the denominators have the same degree. This indicates that there exists a real value λ that can guide the solution process towards satisfying the constraint, since an undefined value only appears when the constraint is met.

B.4 Relationship between $\widetilde{\alpha}^*$ and r^*

Let us recall the definition of $\widetilde{\alpha}$ from Equation (3.16):

$$\widetilde{\alpha} = \frac{\alpha}{\sigma(\mathbf{P}_a^h \cdot \alpha)} \cdot r \tag{B.20}$$

By multiplying both sides for $\widetilde{\mathbf{P}}_a^h$ and applying standard deviation, we obtain:

$$\sigma(\mathbf{P}_a^h \cdot \widetilde{\alpha}) = \sigma(\mathbf{P}_a^h \cdot \frac{\alpha}{\sigma(\mathbf{P}_a^h \cdot \alpha)} \cdot r) \tag{B.21}$$

Since r and $\sigma(\mathbf{P}_a^h \cdot \alpha)$ are constants, they can be extracted from the standard deviation operator, leading to:

$$\sigma(\mathbf{P}_a^h \cdot \widetilde{\alpha}) = \frac{\sigma(\mathbf{P}_a^h \cdot \alpha)}{\sigma(\mathbf{P}_a^h \cdot \alpha)} \cdot r = r \tag{B.22}$$

B.5 Monotonicity of Kernel-Based HGR

Let α^* and β^* be the optimal coefficients obtained from the calculation of HGR-KB$(a, b; h, k)$. We want to demonstrate that, for any pair of integers $p \geq h$ and $q \geq k$, the correlation computed using degrees (p, q) is always greater than or equal to that computed using (h, k).

Since HGR-KB is defined as the maximal Pearson correlation obtained using two optimal coefficients ϕ^* and ψ^*, this boils down to demonstrating that:

$$\exists \phi, \psi \qquad \text{s.t.} \qquad \rho(\mathbf{P}_a^p \cdot \phi, \mathbf{P}_b^q \cdot \psi) \geq \rho(\mathbf{P}_a^h \cdot \alpha^*, \mathbf{P}_b^k \cdot \beta^*), \quad \forall a, b \in \mathbb{R}^n \tag{B.23}$$

or rather, that it is always possible to find a pair of coefficients $\phi \in \mathbb{R}^p$ and $\psi \in \mathbb{R}^q$ that, despite potentially being suboptimal, return a greater or equal Pearson's correlation calculated on the polynomial expansion matrices $\mathbf{P}_a^p$ and $\mathbf{P}_b^q$.

This can be easily demonstrated by choosing the vectors $\phi = \begin{bmatrix} \alpha^* & \mathbf{0}_{p-h} \end{bmatrix}$ and $\psi = \begin{bmatrix} \beta^* & \mathbf{0}_{q-k} \end{bmatrix}$, which are formed by appending zeros to the end of the optimal coefficients α^* and β^*. Using these coefficients, we neutralize the effect of higher orders, thus obtaining:

$$\mathbf{P}_a^p \cdot \phi = \sum_{i=1}^{p} a^i \cdot \phi_i = \sum_{i=1}^{h} a^i \cdot \alpha_i^* + \sum_{i=h+1}^{p} a^i \cdot 0 = \mathbf{P}_a^h \cdot \alpha^* \tag{B.24}$$

$$\mathbf{P}_a^q \cdot \psi = \sum_{i=1}^{q} b^i \cdot \psi_i = \sum_{i=1}^{k} a^i \cdot \beta_i^* + \sum_{i=k+1}^{q} b^i \cdot 0 = \mathbf{P}_b^k \cdot \beta^* \tag{B.25}$$

which results in $\rho(\mathbf{P}_a^p \cdot \phi, \mathbf{P}_b^q \cdot \psi) = \rho(\mathbf{P}_a^h \cdot \alpha^*, \mathbf{P}_b^k \cdot \beta^*)$. Considering that ϕ and ψ are potentially suboptimal, we determine that:

$$\rho(\mathbf{P}_a^p \cdot \phi^*, \mathbf{P}_b^q \cdot \psi^*) \geq \rho(\mathbf{P}_a^p \cdot \phi, \mathbf{P}_b^q \cdot \psi) = \rho(\mathbf{P}_a^h \cdot \alpha^*, \mathbf{P}_b^k \cdot \beta^*) \tag{B.26}$$

Appendix C

Proofs of Chapter 4

This appendix presents the demonstrations of theorems and properties discussed in Chapter 4. The same content, with minor edits, is available in the technical appendix of [Giuliani et al., 2023].

C.1 Closed-form Computation of GeDI

We start by considering the definition of GeDI as outlined in Equation (4.4):

$$\arg\min_{\alpha,r} \left\|(\mathbf{P}_a^h \cdot \alpha - \mu(\mathbf{P}_a^h \cdot \alpha)) \cdot r - (b - \mu(b))\right\|_2^2 \quad \text{s.t.} \quad \sigma(\mathbf{P}_a^h \cdot \alpha) = \sigma(a) \tag{C.1}$$

As usual, we can substitute the polynomial kernel and the target vector with their zero-centered versions $\widetilde{\mathbf{P}}_a^h$ and $\widetilde{b}$. Then, we employ a Langrangian multiplier λ to incorporate the constraint into the objective function $C(r, \alpha, \lambda)$, yielding:

$$\arg\min_{r,\alpha,\lambda} C(r, \alpha, \lambda) := \left\|\widetilde{\mathbf{P}}_a^h \cdot \alpha \cdot r - \widetilde{b}\right\|_2^2 + \lambda \left[\sigma(\mathbf{P}_a^h \cdot \alpha) - \sigma(a)\right] \tag{C.2}$$

To determine the optimal solution of the objective function, we need its gradient to be zero. This implies that having a null derivative with respect to r is a necessary condition for optimality, hence we can determine the value of r^* by calculating the

derivative of $C(r, \alpha, \lambda)$ in terms of r:

$$\frac{\partial C(r, \alpha, \lambda)}{\partial r} = 2 \cdot (\widetilde{\mathbf{P}}_a^h \cdot \alpha)^T \cdot (\widetilde{\mathbf{P}}_a^h \cdot \alpha \cdot r - \widetilde{b}) \tag{C.3}$$

and posing it to zero in order to obtain the following equivalence:

$$2 \cdot (\widetilde{\mathbf{P}}_a^h \cdot \alpha)^T \cdot (\widetilde{\mathbf{P}}_a^h \cdot \alpha) \cdot r = 2 \cdot (\widetilde{\mathbf{P}}_a^h \cdot \alpha)^T \cdot \widetilde{b} \tag{C.4}$$

The scalar values $(\widetilde{\mathbf{P}}_a^h \cdot \alpha)^T \cdot (\widetilde{\mathbf{P}}_a^h \cdot \alpha)$ and $(\widetilde{\mathbf{P}}_a^h \cdot \alpha)^T \cdot \widetilde{b}$ represent, respectively, the variance of the vector $\mathbf{P}_a^h \cdot \alpha^*$ and the covariance between $\mathbf{P}_a^h \cdot \alpha^*$ and b, both multiplied by a constant n. Therefore, Equation (C.4) can be rewritten as:

$$2n \cdot \mathrm{var}(\mathbf{P}_a^h \cdot \alpha) \cdot r = 2n \cdot \mathrm{cov}(\mathbf{P}_a^h \cdot \alpha, b) \tag{C.5}$$

from which we can derive the explicit expression for r^* as:

$$r^* = \frac{\mathrm{cov}(\mathbf{P}_a^h \cdot \alpha^*, b)}{\mathrm{var}(\mathbf{P}_a^h \cdot \alpha^*)} \tag{C.6}$$

As a final step, we can leverage the constraint on the standard deviation outlined in Equation (4.4) to replace $\mathrm{var}(\mathbf{P}_a^h \cdot \alpha^*)$ with $\mathrm{var}(a)$, given that variance is defined as the square of the standard deviation. Ultimately, since the GeDI indicator was defined in Equation (4.5) as the absolute value of r^*, we arrive at:

$$\mathrm{GeDI}(a, b; h) = \left| \frac{\mathrm{cov}(\mathbf{P}_a^h \cdot \alpha^*, b)}{\mathrm{var}(a)} \right| \tag{C.7}$$

C.2 Equivalence Between GeDI and DIDI

Let us consider a categorical attribute $a \in \mathcal{A}$ and a continuous attribute $b \in \mathcal{B}$. The definition of DIDI in regression tasks as outlined by [Aghaei et al., 2019] is:

$$\mathrm{DIDI}(a, b) = \sum_{v \in \mathcal{A}} \left| \frac{\sum_{i=1}^n b_i \cdot \mathbb{I}(a_i = v)}{\sum_{i=1}^n \mathbb{I}(a_i = v)} - \frac{1}{n} \sum_{i=1}^n b_i \right| \tag{C.8}$$

where $\mathbb{I}(\pi)$ represents the indicator function that returns 1 if the proposition π holds true and 0 otherwise.

We restrict our proof to scenarios where a takes binary values, meaning that $\mathcal{A} = \{0, 1\}$. We assume that there is at least one element of each group, i.e., $\exists i, j \in \{1, \ldots, n\} \mid a_i = 0 \wedge a_j = 1$; then, based on the value of v, we can replace the indicator function $\mathbb{I}(a_i = v)$ with $1 - a_i$ or a_i, resulting in:

$$\text{DIDI}(a, b) = \left| \frac{\sum_{i=1}^{n} b_i \cdot (1 - a_i)}{\sum_{i=1}^{n} (1 - a_i)} - \frac{1}{n} \sum_{i=1}^{n} b_i \right| + \left| \frac{\sum_{i=1}^{n} b_i \cdot a_i}{\sum_{i=1}^{n} a_i} - \frac{1}{n} \sum_{i=1}^{n} b_i \right| \tag{C.9}$$

By factorizing the multiplications and dividing numerator and denominator of each fractional term by a constant factor n, we can reformulate Equation (C.9) as:

$$\text{DIDI}(a, b) = \left| \frac{\frac{1}{n} \sum_{i=1}^{n} b_i \cdot \frac{1}{n} \sum_{i=1}^{n} a_i b_i}{\frac{1}{n} \sum_{i=1}^{n} 1 - \frac{1}{n} \sum_{i=1}^{n} a_i} - \frac{1}{n} \sum_{i=1}^{n} b_i \right| + \left| \frac{\frac{1}{n} \sum_{i=1}^{n} a_i b_i}{\frac{1}{n} \sum_{i=1}^{n} a_i} - \frac{1}{n} \sum_{i=1}^{n} b_i \right| \tag{C.10}$$

At this stage, we carry out the following substitutions:

$$\frac{1}{n} \sum_{i=1}^{n} 1 = 1 \qquad \frac{1}{n} \sum_{i=1}^{n} a_i = \mu_a \qquad \frac{1}{n} \sum_{i=1}^{n} b_i = \mu_b \qquad \frac{1}{n} \sum_{i=1}^{n} a_i b_i = \mu_{ab} \tag{C.11}$$

where μ_a, μ_b, and μ_{ab} denote the mean values of vectors a, b, and $a \odot b$, respectively. Using these replacements in Equation (C.10), we get:

$$\text{DIDI}(a, b) = \left| \frac{\mu_b - \mu_{ab}}{1 - \mu_a} - \mu_b \right| + \left| \frac{\mu_{ab}}{\mu_a} - \mu_b \right| \tag{C.12}$$

$$= \left| \frac{\mu_b - \mu_{ab} - \mu_b \cdot (1 - \mu_a)}{1 - \mu_a} \right| + \left| \frac{\mu_{ab} - \mu_b \cdot \mu_a}{\mu_a} \right| \tag{C.13}$$

$$= \left| \frac{\mu_a \mu_b - \mu_{ab}}{1 - \mu_a} \right| + \left| \frac{\mu_{ab} - \mu_a \mu_b}{\mu_a} \right| \tag{C.14}$$

In this context, we can observe that $\mu_{ab} - \mu_a \mu_b$ denotes the covariance between a and b. Moreover, by multiplying and dividing both terms by the other's denominator,

we adjust them to the same value and obtain:

$$\text{DIDI}(a,b) = \left|\frac{\mu_a}{\mu_a} \cdot \frac{\text{cov}(a,b)}{1-\mu_a}\right| + \left|\frac{1-\mu_a}{1-\mu_a} \cdot \frac{\text{cov}(a,b)}{\mu_a}\right| \tag{C.15}$$

$$= \left|\frac{\mu_a \cdot \text{cov}(a,b)}{\mu_a \cdot (1-\mu_a)}\right| + \left|\frac{(1-\mu_a) \cdot \text{cov}(a,b)}{(1-\mu_a) \cdot \mu_a}\right| \tag{C.16}$$

$$= \left|\frac{\mu_a \cdot \text{cov}(a,b)}{\mu_a - \mu_a^2}\right| + \left|\frac{\text{cov}(a,b) - \mu_a \cdot \text{cov}(a,b)}{\mu_a - \mu_a^2}\right| \tag{C.17}$$

Given that a is a binary vector, it follows that $0 \leq \mu_a \leq 1$, implying that the numerators have opposite signs as they entirely depend on $\text{cov}(a,b)$. Consequently, the absolute values can be combined into a single entity, resulting in:

$$\text{DIDI}(a,b) = \left|\frac{\mu_a \cdot \text{cov}(a,b) + \text{cov}(a,b) - \mu_a \cdot \text{cov}(a,b)}{\mu_a - \mu_a^2}\right| = \left|\frac{\text{cov}(a,b)}{\mu_a - \mu_a^2}\right| \tag{C.18}$$

Finally, we can observe that the vector a is invariant to the power operator due to its binary nature. Consequently, $\mu_a = \mu_{a^2}$ and, as a result, the denominator in Equation (C.18) simplifies to the variance of a. Therefore:

$$\text{DIDI}(a,b) = \left|\frac{\text{cov}(a,b)}{\text{var(a)}}\right| \tag{C.19}$$

This is precisely the value of $\text{GeDI}(a,b;1)$ as outlined in Section 4.3 and demonstrated in Appendix C.1. Therefore, such a proof shows that our indicator enhances the expressiveness of the DIDI without altering its semantics when dealing with binary input vectors, thus strengthening the link with this established metric.

The same logic applies when both a and b are binary. In this situation, [Aghaei et al., 2019] employs a slightly modified version of DIDI for classification tasks, denoted as DIDI_c, which is:

$$\text{DIDI}_\text{c}(a,b) = \sum_{v \in \mathcal{A}} \sum_{w \in \mathcal{B}} \left| \frac{\sum_{i=1}^{n} \mathbb{I}(a_i = v \wedge b_i = w)}{\sum_{i=1}^{n} \mathbb{I}(a_i = v)} - \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(b_i = w) \right| \tag{C.20}$$

Once more, the indicator function $\mathbb{I}(a_i = v)$ can be substituted with either

$1 - a_i$ or a_i, based on the value of v. Likewise, we substitute $\mathbb{I}(b_i = w)$ with $1 - w_i$ and w_i, and replace $\mathbb{I}(a_i = v \wedge b_i = w)$ with the corresponding product between the previous terms. Ultimately, this yields:

$$\begin{aligned}\mathrm{DIDI}_c(a,b) = &\left|\frac{\sum_{i=1}^n (1-a_i)(1-b_i)}{\sum_{i=1}^n (1-a_i)} - \frac{1}{n}\sum_{i=1}^n (1-b_i)\right| + \\ &\left|\frac{\sum_{i=1}^n a_i(1-b_i)}{\sum_{i=1}^n a_i} - \frac{1}{n}\sum_{i=1}^n (1-b_i)\right| + \\ &\left|\frac{\sum_{i=1}^n (1-a_i)b_i}{\sum_{i=1}^n (1-a_i)} - \frac{1}{n}\sum_{i=1}^n b_i\right| + \\ &\left|\frac{\sum_{i=1}^n a_i b_i}{\sum_{i=1}^n a_i} - \frac{1}{n}\sum_{i=1}^n b_i\right|\end{aligned} \tag{C.21}$$

By using the same notation presented in Equation (4.8) and performing similar mathematical operations as those previously described, we arrive at:

$$\mathrm{DIDI}_c(a,b) = \left|\frac{\mu_a\mu_b - \mu_{ab}}{1-\mu_a}\right| + \left|\frac{\mu_{ab} - \mu_a\mu_b}{\mu_a}\right| + \left|\frac{\mu_a\mu_b - \mu_{ab}}{1-\mu_a}\right| + \left|\frac{\mu_{ab} - \mu_a\mu_b}{\mu_a}\right| \tag{C.22}$$

This value is twice what is presented in Equation (C.14), which means that DIDI_c is twice our indicator $\mathrm{GeDI}(a, b; 1)$. Hence, the two values are distinguished solely by a constant scaling factor, which is frequently cancelled out on its own, since we usually constrain the DIDI indicator up to a relative threshold that depends on the original level of discrimination.