IMPERIAL

DESIGN ENGINEERING MASTERS PROJECT

IMPERIAL COLLEGE LONDON

DEPARTMENT OF DESIGN ENGINEERING

Practical Aspects of Fairness in AI: Implementing State of The Art Algorithms in the AIF360 Toolkit

Author:
Joseph W. Johnson
CID: 01850831

Supervisor:
Professor Anthony Quinn

June 06, 2024

I would like to acknowledge my supervisor, Professor Anthony Quinn, for his continued guidance and generously given time throughout this academic year. I would also like to thank Abi Langbridge for her endless support, Professor Robert Shorten for providing me with the opportunity to be a part of this work, and Dr Jakub Marecek for his practical advice.

Abstract

Subgroup Fairness, Instantaneous Fairness, and Distributional Repair are state-of-the-art AI fairness remediation algorithms. Work was performed to adapt these from novel published methods and code into general-purpose tools that can be applied as part of the AIF360 fairness toolkit. The need for such tools within the AI fairness agenda and the principles underlying each algorithm are established. A novel evaluation of the algorithms is provided, assessing their scalability across various cases.

Contents

Chapter 1

Introduction

AI fairness is crucial as organisations deploy automated systems to predict human factors like creditworthiness [1], hiring suitability[2], and criminal re-offence risk[3]. Whilst these systems save time and allow all individuals to be assessed under the same process rather than by individuals with their own biases, they can also introduce or amplify unfairness. The field of AI fairness aims to measure and remediate these biases.

Being a relatively modern discipline, technology organisations, including IBM [4], Microsoft [5], and Google [6] are developing toolkits that allow AI practitioners to assess and repair their systems, using state of the art methods. As new methods are developed, work must be done to integrate within toolkits in order to make them accessible to a broad audience.

My work extends IBM's AIF360 toolkit [4] with three new algorithms: Subgroup and Instantaneous Fairness [7], which reweigh model outputs to equalise loss between subgroups; and Distributional Repair [8], which maximises conditional independence of datasets before usage. I adapted research implementations into versatile, accessible tools within AIF360 that can be applied to any dataset.

The project goals were to:

- Explain algorithm functions and concepts for accessibility.
- Adapt research algorithms into generalised implementations.
- Integrate such algorithms within the AIF360 toolkit.
- Produce clear documentation of my work.
- Evaluate algorithms in ways that the hard-coded approaches did not allow.

In Chapter 2, I will introduce the field of AI fairness, explaining the need for a range of fairness tools and the role that AI fairness toolkits play in enabling the real-world application of AI fairness algorithms. Following this, Chapter 3 outlines the three state-of-the-art research algorithms I will implement, along with associated concepts, such as min-max optimisation and conditional independence. Chapter 4 outlines the general software methodology and approach used to convert research algorithms to AIF360-compatible tools, proposing a framework that others can use to implement future algorithms. Chapter 5 and Chapter 6 document the integration of algorithms within AIF360, showing how these were generalised, implemented and evaluated. Finally, Chapter 7 concludes the report, evaluating my work against my goals, reflecting on my work, and providing suggestions for future work.

Direct links to my implementation can be found in Appendix A

Chapter 2

AI Fairness: Background and Current Challenges

This chapter aims to provide an understanding of fairness within the context of AI. The S, X, Y model, which will be extended to the S, U, X, Y model in Section 3.2, is defined and used to demonstrate the breadth of fairness algorithms.

2.1 An Introduction To Fairness

The notion of fairness is a cornerstone of not only responsible AI systems [9] but many modern institutions such as the United Nations [10], stating that all member states "*are accountable to just, fair and equitable laws*". But what is *fair*? On its surface, achieving a fair outcome can seem trivial; however, as I will demonstrate, this is a complex question with no single answer spanning mathematics, statistics and ethics.

To frame this, let us consider the example of University Admissions. Whilst one may argue it is simply a case of admitting those with the most qualifications, in this case, grades or achievements, another may say that it is fairer to consider an individual's circumstance and adjust accordingly. In turn, this could prevent a more qualified individual from achieving admittance due to their advantageous individual circumstance - could this even be argued to be a form of discrimination? In Section 2.2, the notation used throughout the report is formalised, and following this, Section 2.3 explores these notions of fairness under rigorous mathematical formulation.

2.2 S, X, Y Model and Notation

Sensitive attributes, S**,** also known as protected attributes, are characteristics of an individual that are considered illegal or unethical to discriminate against [11]. For example, in the United Kingdom, it is unlawful to discriminate based on age, marital status, disability, race, or sexual orientation[12]. Let $\mathcal{S} = \{s_1, s_2, \ldots, s_i \quad \forall i \in 1, .., n\}$ denote the set of sensitive attribute values present within a dataset of n entries. A binary model is used for S, in which $s \in \{0, 1\}$, representing unprivileged and privileged, respectively.

Features, X**,** are characteristics an AI system uses to make a prediction. These can include raw data such as income, categorised or binned attributes such as pay bands, processed data such as the result of kernel sampling, and calculated metrics such as creditworthiness. Let $\mathcal{X} = \{\{x_1^1, \ldots, x_1^k\} \ldots \{x_i^1, \ldots, x_i^k\} \quad \forall k \in X \quad \forall i \in 1, \ldots, n\}$ denote the set of features present within a dataset of n individuals, where each x_i^k represents the value of X feature k for individual n.

Model outcome, $\hat{Y}$**,** is the output of an AI pipeline, determined by the model $g(\mathcal{X}|\mathcal{S})$. This is typically the output of a classification model, where Y is a statistical output such as classification probability, and $\hat{Y}$

is the classified type. Let $\hat{\mathcal{Y}} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_i \quad \forall i \in 1, .., n\}$ denote the set of outcomes for a dataset of n individuals, where $\hat{y} \in \{0, 1\}$, representing an unfavourable and favourable outcome respectively.

The dataset, $\mathcal{D}$, can be represented as shown in Equation (2.1).

$$\mathcal{D} = \left\{ \mathcal{S} = \begin{Bmatrix} s_1 \\ s_2 \\ \vdots \\ s_i \end{Bmatrix}, \mathcal{X} = \begin{Bmatrix} x_1^1 & \ldots & x_1^k \\ x_2^1 & \ldots & x_2^k \\ \vdots & \ddots & \vdots \\ x_i^1 & \ldots & x_i^k \end{Bmatrix} \forall k \in X, \hat{\mathcal{Y}} = \begin{Bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_i \end{Bmatrix}, \quad \forall i \in 1, \ldots, n \right\} \tag{2.1}$$

2.2.1 Core Fairness Concepts

Regardless of the specific notion used to measure and remediate fairness, group and instantaneous fairness should be considered.
Group Fairness: Group fairness[13] requires that, on the whole, each group is treated equitably. This can be represented as:

$$T(S = 0) = T(S = 1) \tag{2.2}$$

Where $T(S)$ represents the treatment T of a group with sensitive attribute S, treatment is a broad term with many meanings, such as the probability of a favourable outcome or accuracy.

Individual Fairness: Individual fairness[13] requires that individuals with similar features, X, receive similar treatment, regardless of their sensitive attributes, S. In other words, if two individuals have the same features but differ in their sensitive attributes, they should receive equal treatment. This can be expressed as:

$$T(X = (x^1, x^2, x^3), S = 1) = T(X = (x^1, x^2, x^3), S = 0) \tag{2.3}$$

where $T(X, S)$ represents the treatment of an individual, given that the X are the same.

2.3 Notions of Fairness

Many notions of a 'fair' system exist, with many of these definitions being incompatible. As such, practitioners must choose an appropriate notion to measure and repair their system. This includes AI practitioners implementing algorithms and policymakers, ultimately responsible for systems fairness[14] [15].

Below is a selection of simplistic fairness notions, highlighting how approaches can differ, each with advantages and challenges. The selection shows no universal definition of 'fair', and no clear choice exists, even in a given situation.

2.3.1 Group Unaware

Under group unawareness, a fair system is one in which S_i is not considered by the model, g when determining $\hat{Y}$, as shown in Equation (2.4).

$$\hat{\mathcal{Y}} = f(\mathcal{X}) \tag{2.4}$$

This is one of the most simplistic approaches. However, it is naive as if a correlation exists between X and S, for example, postcodes correlating to different ethnic groups [16]. Any underlying bias in this correlation will affect the fairness of the outcome. Despite these flaws, this approach to fairness has been

adopted, such as in Frances's discrimination policies in which it is illegal to record an individual's race [17] except for in extreme circumstances, with policies that instead take a 'colour-blind' approach to fairness, addressing socio-economic unfairness through X attributes such as geographic location [18].

2.3.2 Group Threshold

Group Threshold Fairness applies a different threshold, t, for different S attributes when determining $\hat{Y}$ [19]. For example, in a recruitment system where Y is a hiring worthiness score, and $\hat{Y}$ is whether they are hired or not, t may be lowered for under-represented groups within the organisation, as shown in Equation (2.5).

$$\hat{y}_i = \begin{cases} 1, & \text{if } y_i > t_{s=j} \quad \forall i \in 1, \ldots, n, \quad \forall j \in 0, 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

where t_{sj} represents the threshold for the sensitive attribute j.

Model architects adjust thresholds to achieve desired diversity distributions. However, this poses the challenge of determining fair and justified thresholds for each demographic. This task is often done through statistical methods or manual tuning to achieve desired demographic distributions. This approach is frequently criticised due to its positive discrimination [20], with individuals' outcomes being viewed as a result of their S attribute rather than their merit as an individual, both by others and themselves [21].

2.3.3 Demographic Parity

A system is considered fair in demographic parity if each group has the same probability of a positive outcome [22]. This can be given by:

$$P(\hat{Y} = 1|S = 1) = P(\hat{Y} = 1|S = 0) \tag{2.6}$$

This approach has merit as it leads to equal representation, which can effectively counter historical or societal inequality. In many cases, the law calls for equal representation, and in these situations, demographic parity can be a suitable definition of fairness. However, this approach does not account for situations where some demographics have more qualified individuals, leading to qualified individuals being rejected in favour of less qualified individuals due to their demographic, which could be seen as unfair by some.

2.3.4 Equal Accuracy

In a system with equal accuracy, the model is equally accurate for all demographics [23], as demonstrated below:

$$\text{Accuracy}(S = 1) = \text{Accuracy}(S = 0) \tag{2.7}$$

Let us consider the scenario of a healthcare tool designed to predict if patients are at risk of a disease. Equal accuracy would ensure that it is equally likely to make a correct prediction regardless of S. However, if specific demographics were more challenging to predict, the overall accuracy would have to decrease to achieve equal accuracy, thus putting more lives at risk.

2.4 Toolkits for Fairness in AI

While these notions are simplistic, they highlight the need for a broad range of fairness definitions to fit a range of AI practitioner's use cases. Here, the need for AI fairness toolkits arises, allowing users to apply a broad range of fairness definitions, metrics, and remediation algorithms. These toolkits feature many algorithms designed to perform these tasks, all with standardised usage patterns. For AI practitioners, this results in much less work as they need only install and learn one library from which all algorithms can be trusted to work, a more accessible approach than individually finding algorithms, understanding their specific usage patterns, and trusting that each developer has created a functioning and well-maintained tool as opposed to a larger community often managed by a reputable party.

One toolkit is AIF360 [4], founded by IBM and now part of the Linux Foundation [24]. It is an open-source toolkit designed to provide practitioners with the tools and knowledge necessary to detect and mitigate bias within their systems. AIF360 follows professional open-source development standards and practices to aid its adoption into existing pipelines and encourage contributions from the broader research community. AIF360 enforces standardised methods between its tools, allowing users to select and exchange algorithms in a modular architecture. Fairness remediation algorithms are categorised into pre-processing, in-processing, and post-processing. Pre-processing algorithms operate on the dataset used by a model. In-processing algorithms are used within the training process of an algorithm to alter the trained model. Post-processing algorithms are applied to the output of a model. Existing algorithms and my contributions are listed in Table 2.1.

Pre-Processing	In-Processing	Post-Processing
• Disparate Impact Remover [25] • Learning Fair Representation [26] • Optimised Preprocessing [27] • Reweighing [28] • **Distributional Repair**	• Adversarial Debiasing[29] • ART Classifier[30] • Exponentiated Gradient Reduction [31] • Gerryfair Classifier[32] • Grid Search Reduction [31] • Meta Fair Classifier[33] • Prejudice Remover [34]	• Calibrated Equalised Odds Processing [35][36] • Deterministic Reranking [37] • Equalised Odds Post-Processing [35][36] • Reject Object Classification [38] • **Subgroup Fair Optimiser** • **Instantaneous Fair Optimiser**

Table 2.1: Fairness Remediation Algorithms Within AIF360 *(My contributions highlighted in purple)*

Within AIF360, standard datasets used for assessing fairness algorithms come pre-implemented. This enables contributors to implement and benchmark their code on several datasets while minimising the time spent wrangling data. The pre-implemented datasets include:

- The COMPAS dataset [39] derived from a tool used in the U.S. criminal justice system [40] to predict the likelihood of recidivism, is used to assess bias against age, sex, and race.
- The Adult Census Income dataset [41], based on the 1994 U.S. Census results, uses gender and race as sensitive attributes to examine income disparity.
- The German Credit dataset [42], which classifies individuals as good or bad credit risks, includes sensitive characteristics like sex and age, as well as potentially correlated attributes, such as unemployment periods, which could hurt women who had taken a break from work to raise children.

I will use these pre-implemented datasets throughout my work to test and evaluate my implemented algorithms.

Chapter 3

State-of-the-Art Notions Of AI Fairness and Repair

This chapter aims to explain the implemented algorithms, define their notions of fairness, and explain the core mathematical concepts behind them.

3.1 Subgroup and Instantaneous Fairness

Subgroup and Instantaneous Fairness notions expand on the S, X, Y model by treating S as a set of sensitive attributes rather than just one. This creates a set of subgroups, $\check{s} \in \check{\mathcal{S}}$, where each $\check{s}$ is a combination of S attributes such as black male or white female. A set of trajectories exist for each subgroup $\check{s}$, given by $o \in \mathcal{O}^{\check{s}} \forall \check{s}$, where each o is the forecast over the time period $\mathcal{T}^{(o,\check{s})}$. In many AI systems, a single model g is used for all subgroups, trajectories, and periods to create a single forecast f_t. This approach can lead to under-representation bias if certain subgroups have fewer examples in the training data.

For example, consider a credit scoring system that predicts an individual's likelihood of defaulting on a loan. The population can be divided into subgroups based on sensitive attributes like race or gender. Suppose the training data contains significantly fewer examples of a particular subgroup, such as African American females. The model may not learn to make accurate predictions for this subgroup, leading to potentially unfair credit decisions.

In response to this, Quan [7] proposes two new fairness algorithms, Subgroup and Instantaneous Fairness, as detailed below in Section 3.1.1 and Section 3.1.2 respectively. For both algorithms, a loss function ${}^{(o,\check{s})}\operatorname{loss}(f_t)$ is used, as defined in Equation (3.1).

$$ {}^{(o,\check{s})}\operatorname{loss}\left(f_t\right) := \left\| Y_t^{(o,\check{s})} - f_t \right\| \tag{3.1} $$

Subgroup and Instantaneous Fairness utilise min-max optimisation, also known as minimax or saddle point optimisation. This aims to minimise the potential maximum loss or maximise the potential minimum gain in situations concerning multiple groups or individuals to equalise them according to a measure such as a score or loss function, ${}^{(o,\check{s})}\operatorname{loss}(f_t)$, such that a system disproportionately disadvantages no group. By employing min-max optimisation, algorithms can balance the competing losses for each group, thus increasing the model's fairness.

3.1.1 Subgroup Fairness

Subgroup Fairness aims to minimise the sum of losses for each subgroup, thus minimising the maximum average loss across all subgroups over the entire time frame, as shown in Equation (3.2). Considering the entire timeframe at once, the Subgroup Fairness method best suits datasets with consistent unfairness.

$$\min_{f_t, t\in\mathcal{T}^+} \max_{\check{s}\in\check{\mathcal{S}}} \left\{ \frac{1}{|\mathcal{O}^{\check{s}}|} \sum_{o\in\mathcal{O}^{\check{s}}} \frac{1}{|\mathcal{T}^{(o,\check{s})}|} \sum_{t\in\mathcal{T}^{(o,\check{s})}} {}^{(o,\check{s})}\,\text{loss}\,(f_t) \right\} \tag{3.2}$$

For example, Subgroup Fairness may be applied to a hiring AI pipeline so that, over time, its recommendations promote a diverse workforce. This encourages hiring the most suitable candidates at any given time while maintaining a balanced distribution of subgroups over the long term, thus resulting in an overall diverse workforce within an organisation.

3.1.2 Instantaneous Fairness

Instantaneous Fairness aims to minimise the maximum loss across all subgroups over each time step, equalising the instantaneous loss, as shown in Equation (3.3). While Subgroup Fairness considers the whole timeframe simultaneously, Instantaneous Fairness considers each period individually, aiming for constant Fairness. This makes it better suited to datasets where the unfairness varies over time or if real-time Fairness is more important than long-term Fairness.

$$\min_{f_t, t\in\mathcal{T}^+} \left\{ \max_{t\in\mathcal{T}^{(o,\check{s})}, o\in\mathcal{O}^{\check{s}}, \check{s}\in\check{\mathcal{S}}} \left\{ {}^{(O,\check{s})}\,\text{loss}\,(f_t) \right\} \right\} \tag{3.3}$$

An example of where Instantaneous Fairness may be applicable is within a news feed algorithm. In this way, as news is published and an algorithm determines whether or not to show it to members, no subgroups are disproportionately shown or not shown the news story. This leads to all subgroups being equally informed of current events, regardless of how the news might align with their political stance. This impact can extend far beyond their view of the world, as news feed algorithms are highly influential in cases such as elections.

Figure 3.1 provides a visual example of how these algorithms affect the predictions of a model on a dataset with two subgroups.

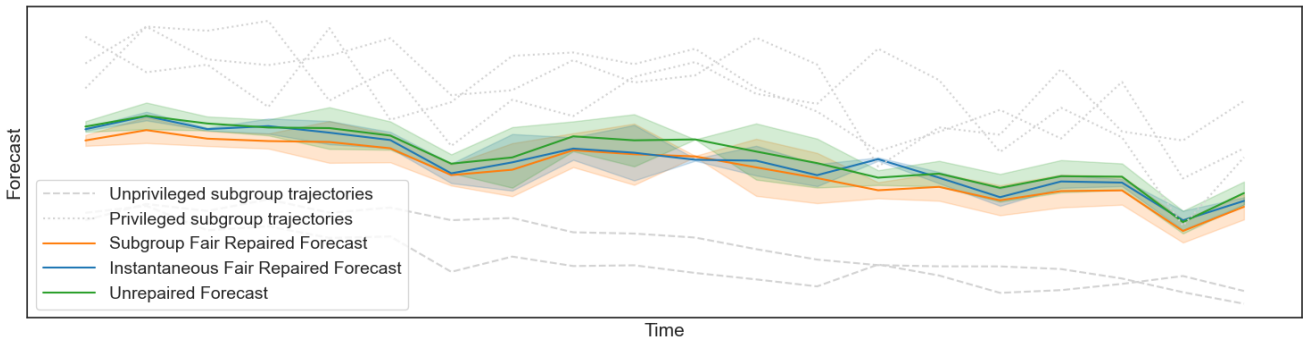


Figure 3.1: Example application of Subgroup and Instantaneous Fairness applied to a forecast

3.2 Conditional Independence

The S, X, Y model is augmented with an additional attribute, U, when considering conditional independence.

Insensitive attributes, U, also known as unprotected attributes, are characteristics of an individual that are considered fair to use when evaluating someone but are not used by the model, g, to determine $\hat{Y}$. Examples include education, social class, political affiliation, or social media presence. Let $\mathcal{U} = \{\{u_1^1, \ldots, u_1^m\} \ldots \{u_i^1, \ldots, u_i^m\} \quad \forall m \in U \quad \forall i \in 1, \ldots, n\}$ denote the set of m insensitive attributes present within a dataset, where each u_i^m represents an individual insensitive attribute.

As a result, when discussing conditional independence, the dataset, $\mathcal{D}$, can be represented as:

$$\mathcal{D} = \left\{ \mathcal{S} = \begin{Bmatrix} s_1 \\ s_2 \\ \vdots \\ s_i \end{Bmatrix}, \mathcal{U} = \begin{Bmatrix} u_1^1 & \dots & u_1^m \\ u_2^1 & \dots & u_2^m \\ \vdots & \ddots & \vdots \\ u_i^1 & \dots & u_i^m \end{Bmatrix}, \mathcal{X} = \begin{Bmatrix} x_1^1 & \dots & x_1^k \\ x_2^1 & \dots & x_2^k \\ \vdots & \ddots & \vdots \\ x_i^1 & \dots & x_i^k \end{Bmatrix}, \hat{\mathcal{Y}} = \begin{Bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_i \end{Bmatrix} \right\} \tag{3.4}$$

Figure 3.2 shows the graphical model of $\mathcal{D}$ where $\mathcal{U}$ is dependent on $\mathcal{S}$, and $\mathcal{X}$ is dependent on $\mathcal{U}$.

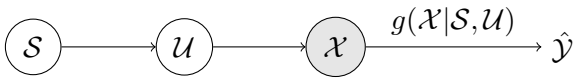


Figure 3.2: Graphical model of a conditional independence fair system

Figure 3.3 shows an unfair system under the S, U, X, Y model, in which there exists a dependence of $\mathcal{X}$ on $\mathcal{S}$. A conditional independence-based repair method, such as Distributional Repair, aims to minimise correlation between $\mathcal{U}$ and $\mathcal{X}$

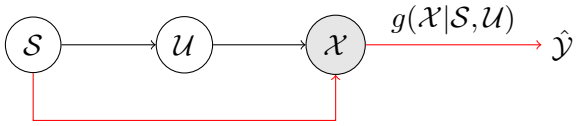


Figure 3.3: Graphical model of a conditional independence unfair system

We construct an $|\mathcal{X}| + 1$ dimensional space when evaluating conditional dependence. Within this, there exist probability functions $P_1 = P(\hat{Y} = 1|S = 1, U, X)$ and $P_0 = P(\hat{Y}|S = 0, U, X$ for the privileged and unprivileged groups respectively. Using the KullbackLeibler Divergence *(KLD)* measure, we can determine the difference between the probability mass functions, in which a lower score indicates greater conditional independence. KLD is calculated in Equation (3.5) and Equation (3.6) for discrete and continuous X attributes, respectively.

$$D_{\mathrm{KL}}(P_0||P_1) = \sum_{x \in \mathcal{X}} P_0(x) \log \frac{P_0(x)}{P_1(x)} \tag{3.5}$$

$$D_{\mathrm{KL}}(P_0||P_1) = \int_{\chi} P_0(x) \log \frac{P_0(x)}{P_1(x)} dx \tag{3.6}$$

Del Barrio et al. [43] propose that to minimise the KLD of the distributions, an optimal transport plan can be constructed to drive both distributions towards a common distribution. The Wasserstein barycentre [44] is used for this, as it is the point between the two distributions that requires minimal work to transform the distributions, thus resulting in minimised damage, or change, to the original data. This results in X values being reweighed so that they have reduced conditionality on U. This process is demonstrated below in Figure 3.4.

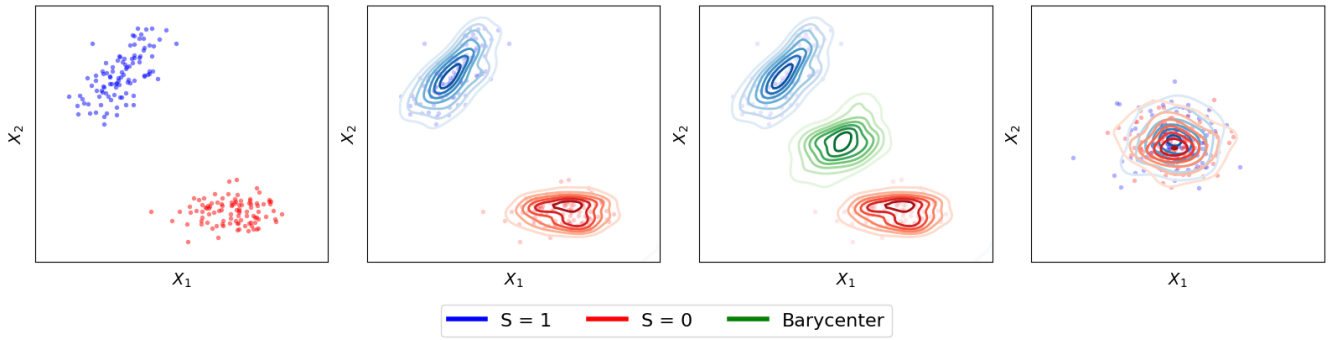


Figure 3.4: The process of using an optimal transport plan to move data towards the Barycentre with 2 X features, from left to right: Mapping instances of $\hat{Y} = 1$ against X; Constructing probability mass functions for $S = 1$ and $S = 0$; determining the Barycentre between the two distributions; applying the optimal transport plan to the data

Optimal transport *(OT)* is a data transformation method that calculates a plan to move data from one point to another while minimising the expected cost, in this case, the difference between the original and the transformed data. OT plans can be calculated from a subset of the data, $\mathcal{D}_{research}$ and then applied to the rest of the data, $\mathcal{D}_{archive}$. The Distributional Repair algorithm [8] uses this method to learn an OT plan and apply it to the broader dataset, allowing AI practitioners to effectively increase the conditional independence of their datasets without the need to reconstruct new OT plans each time the dataset is updated.

These concepts explain the goals of the implemented algorithms, showing what they strive to achieve when remediating unfairness in a dataset. More specific information on how these algorithms function is found in Chapter 5 and Chapter 6.

Chapter 4

Software Methodology for Implementing Algorithms Within AIF360

This chapter outlines the process of converting the research code into general-purpose tools within the AIF360 library and the associated evaluation of these tools.

4.1 Development

The adaptation process began with developing the new tools, as detailed below.

4.1.1 Algorithm Redesign and Refactoring

I started by writing abstract pseudo-codes to understand the generalised processes within the algorithms, gaining a deeper understanding of how data is manipulated. The existing research code was hard-coded for singular datasets, making it inaccessible to a broader audience. To address this, I refined the pseudo-code to use parameters such as S, U, X, Y column names, allowing them to be used on any dataset. I then wrote the Python code using strong object-oriented principles, abstracting methods into subfunctions to improve readability and reusability. This improved the source code bases, which often lacked these principles, leading to long and unstructured algorithms.

4.1.2 Implementation

Implementing algorithms within AIF360 must adhere to standardised inputs, functions, and returns for their respective tool types. This involves adapting the algorithms to use `dataset` objects instead of their existing inputs, such as Pandas `DataFrame` objects [45]. Algorithms must also use standardised functions like `fit`, `transform`, and `predict`, which behave consistently across AIF360, allowing users to switch between algorithms easily.

Dependencies are updated to match pre-existing ones within the codebase, reducing user installation requirements and increasing readability. Selected tools should be purely Python-based, allowing out-of-the-box usage when installing AIF360 with the Python Package Manager *(PIP)* [46].

4.1.3 Documentation

The research code often lacked user documentation, comments, and formal guidance, making it inaccessible to a broader audience. Therefore, I wrote thorough documentation adhering to AIF360 standards [47]. This included writing sphinx-compatible [48] docstrings for automatic integration with the AIF360 website

[4]. Demonstration files were created in Jupyter [49] format, allowing functioning Python code and detailed Markdown text to be written side by side for visually appealing and readable documentation.

4.1.4 Adherence To Standards

In addition to AIF360's requirements, I ensured my code adhered to professional standards, including additional comments explaining complex steps and following the PEP8 [50] style guide for Python.

4.2 Testing and Evaluation

After implementing the new tools, tests were performed to ensure functionality and assess usability and performance.

4.2.1 Reproducibility Testing

Reproducibility tests ensured the algorithms functioned as expected by running the new AIF360 implementations and original research algorithms with the same parameters and datasets, comparing results using plotted data and statistical measures.

4.2.2 Usability Testing

To test documentation quality, I tasked peers with implementing the tools using the demo file instructions. This validated that the instructions could be used by someone unfamiliar with the algorithms, and feedback was used to update the documentation where needed.

4.2.3 Evaluation of Algorithms

Once implemented, the algorithms were used to evaluate the research algorithms' performance in broader contexts. Due to the hard-coded nature of the research code, the algorithms had only been applied to singular datasets with little exploration of parameter variance impact. I investigated the effects of parameter values on performance and computational runtime and assessed performance metrics on alternate datasets to evaluate the algorithms under different scenarios.

Tests were performed on a Dell Inspiron 16 laptop with an 11th Gen Intel Core i7-11800H CPU at 2.3GHz, NVIDIA GeForce RTX 3060 Laptop GPU, 16GB RAM, and Windows 11 [51].

Chapter 5

LDS Fairness Algorithms

This chapter outlines the development and evaluation of the Subgroup and Instantaneous Fair algorithms within the AIF360 toolkit.

5.1 Development

The development process followed my outlined methodology from Chapter 4, by abstracting, generalising, refactoring and then implementing the algorithm within AIF360.

5.1.1 Algorithm Redesign and Refactoring

First, I generalised the optimisation problem from the original research code[7], using the S, X, $\hat{Y}$ model. While the general structure of both the Subgroup and Instantaneous Fairness are similar, they differ in the optimisation problem that they solve, shown in Equation (3.2) and Equation (3.3), respectively. These optimisation problems can be seen below in Equation (5.1) and Equation (5.2). It is worth noting that the algorithms only act to equalise between pairs of subgroups rather than all subgroups at once. The dataset is filtered by one S value, in this case sex, and then a separate S attribute, such as race, can be applied to equalise loss between subgroup pairs such as white males and non-white males.

Following this, I abstracted and generalised the overall flow of the algorithm, again under the S,X,Y format. As the algorithm operates on the output of a model, it is classified as a post-processing algorithm within AIF360 [4].

When initialised, the class takes S,$\hat{Y}$ as strings, and X as a list of strings. The values of these strings are the names of the columns used for each attribute. These values are then stored for later use, as shown in Algorithm 1.

Once initialised, the fit function, Algorithm 2, is used to generate the decision variables and perform the optimisation. The fit function takes a subset of $\mathcal{D}$, referred to as the training dataset, $\mathcal{D}_{train}$. The base rates at which $\hat{Y} = 1$ occur are calculated for each S value to be used within the transform function. The data is split into datasets where $\hat{Y} = 1$ and $\hat{Y} = 0$. This differs from the original algorithm, which operates using indexing, as splitting the datasets significantly increased the code readability without significantly impacting algorithm performance. Following this, the optimisation problem is formulated and solved, using the optimisations outlined in Equation (5.1) and Equation (5.2) for the Subgroup and Instantaneous Fairness algorithms, respectively. Finally, the optimisation results are stored within the class, and the updated class is returned.

(Subgroup Fairness Optimization Problem)

$$
\begin{aligned}
\text{Decision Variables: } & a^1_{S=0}, a^1_{S=1}, \ldots, a^k_{S=0}, a^k_{S=1} \text{ } (x^1, x^2, \ldots, x^k), \\
& e_0, e_1 \text{ } \textit{(error)}, \\
& z_0, z_1, z_2 \text{ } \textit{(objective function)} \\
\text{Constraints: } & z_0 + \hat{y}_i - \sum_{j=1}^{k} (a^j_{S=0} \cdot x_j) + e_0 \geq 0, \quad \forall \{i \in D \mid s_i = 0\} \\
& z_0 - \hat{y}_i + \sum_{j=1}^{k} (a^j_{S=0} \cdot x_j) + e_0 \geq 0, \quad \forall \{i \in D \mid s_i = 0\} \\
& z_0 + \hat{y}_i - \sum_{j=1}^{k} (a^j_{S=1} \cdot x_j) + e_1 \geq 0, \quad \forall \{i \in D \mid s_i = 1\} \\
& z_0 - \hat{y}_i + \sum_{j=1}^{k} (a^j_{S=1} \cdot x_j) + e_1 \geq 0, \quad \forall \{i \in D \mid s_i = 1\} \\
& z_1 - \frac{1}{|\{i \in D \mid s_i = 0\}|} \sum_{i \in \{i \in D | S_i = 0\}} (\hat{y}_i - \sum_{j=1}^{k} (a^j_{S=0} \cdot x_j) + e_0)^2 \geq 0 \quad \forall \{i \in D \mid s_i = 0\} \\
& z_2 - \frac{1}{|\{i \in D \mid s_i = 1\}|} \sum_{i \in \{i \in D | S_i = 1\}} (\hat{y}_i - \sum_{j=1}^{k} (a^j_{S=1} \cdot x_j) + e_1)^2 \geq 0 \quad \forall \{i \in D \mid s_i = 1\} \\
\text{Objective: } & \min z_0 + z_1 + z_2 + 0.5(z_2 - z_1)
\end{aligned}
\tag{5.1}
$$

(Instantaneous Fairness Optimization Problem)

$$
\begin{aligned}
\text{Decision Variables: } & a^1_{S=0}, a^1_{S=1}, \ldots, a^k_{S=0}, a^k_{S=1} \text{ } (x^1, x^2, \ldots, x^k), \\
& e_0, e_1 \text{ } \textit{(error)}, \\
& z_0, z_1 \text{ } \textit{(objective function)} \\
\text{Constraints: } & \frac{z_0 + \hat{y}i - \sum_{j=1}^{k} (a^j_{S=0} \cdot x_j) + e_0}{|i \in D \mid s_i = 0|} \geq 0, \quad \forall \{i \in D \mid s_i = 0\} \\
& \frac{z_0 - \hat{y}i + \sum_{j=1}^{k} (a^j_{S=0} \cdot x_j) + e_0}{|i \in D \mid s_i = 0|} \geq 0, \quad \forall \{i \in D \mid s_i = 0\} \\
& \frac{z_0 + \hat{y}i - \sum_{j=1}^{k} (a^j_{S=1} \cdot x_j) + e_1}{|i \in D \mid s_i = 1|} \geq 0, \quad \forall \{i \in D \mid s_i = 1\} \\
& \frac{z_0 - \hat{y}i + \sum_{j=1}^{k} (a^j_{S=1} \cdot x_j) + e_1}{|i \in D \mid s_i = 1|} \geq 0, \quad \forall \{i \in D \mid s_i = 1\} \\
& \frac{z_1 + \hat{y}i - \sum_{j=1}^{k} (a^j_{S=0} \cdot x_j) + e_0}{|i \in D \mid s_i = 0|} \geq 0, \quad \forall \{i \in D \mid s_i = 0\} \\
& \frac{z_1 - \hat{y}i + \sum_{j=1}^{k} (a^j_{S=1} \cdot x_j) - e_1}{|i \in D \mid s_i = 1|} \geq 0, \quad \forall \{i \in D \mid S_i = 1\} \\
\text{Objective: } & \min z_0 + z_1
\end{aligned}
\tag{5.2}
$$

Algorithm 1 Initialisation

procedure INITIALISE($S, X, \hat{Y}$)
  store $S, X, \hat{Y}$
end procedure

Algorithm 2 Fit

procedure FIT($\mathcal{D}_{train}$)
  load $S, X, \hat{Y}$
  $baseRates \leftarrow$ CALCULATEBASERATE($\mathcal{D}_{train}, S, \hat{Y}$)
  $\mathcal{D}_{train}^{privileged}, \mathcal{D}_{train}^{unprivileged} \leftarrow$ SPLITDATA($\mathcal{D}_{train}, S$)
  $constraints, objD, xVars, e, z \leftarrow$ CREATEOPTIMIZATION($\mathcal{D}_{train}^{privileged}, D_{train}^{unprivileged}, X, \hat{Y}$)
  $xVars_{solved}, e_{solved}, z_{solved} \leftarrow$ SOLVEOPTIMIZATION($constraints, objD, xVars, e, z$)
  store $xVars_{solved}, e_{solved}, z_{solved}, baseRates$
  return $self$
end procedure

Once the fit function has been performed, the transform function, Algorithm 3, is applied. Here, a subset of $\mathcal{D}$, referred to as the test dataset, $\mathcal{D}_{test}$ is used. $\mathcal{D}_{test}$ is reweighted using the solved decision variables, generating reweighted Y scores which undergo a normalisation process that binds them between 0 and 1. Finally, repaired $\hat{Y}$ values within $\mathcal{D}$ are determined by evaluating the reweighted Y values against the previously calculated base rates, and the new dataset, $\tilde{\mathcal{D}}_{test}$ is returned.

Algorithm 3 Transform

procedure TRANSFORM($\mathcal{D}_{test}$)
  load $S, X, \hat{Y}, xVars_{solved}, e_{solved}, z_{solved}, baseRates$
  $\mathcal{D}_{test}\_reweighted \leftarrow$ REWEIGHDATA($\mathcal{D}_{test}, X, \hat{Y}, xVars_{solved}, e_{solved}$)
  $\mathcal{D}_{test}normalised \leftarrow$ NORMALIZE($\mathcal{D}_{test}reweighted, \hat{Y}$)
  $\tilde{\mathcal{D}}_{test} \leftarrow$ APPLYTHRESHOLD($\mathcal{D}_{test}normalised, baseRates, S, \hat{Y}$)
  return $\tilde{\mathcal{D}}_{test}$
end procedure

Due to the high similarity between both methods, I refactored them into two separate classes, SubgroupFair and InstantaneousFair, which both inherit from a BaseLDS class. This significantly improved the codebase's maintainability, as debugging changes reflected in both algorithms. The BaseLDS class is an abstraction of the AIF460 Transformer class, the base class for processes that act on datasets within the AIF360 toolkit. A Unified Modelling Language (UML) diagram of the class structure can be seen below in Figure 5.1.

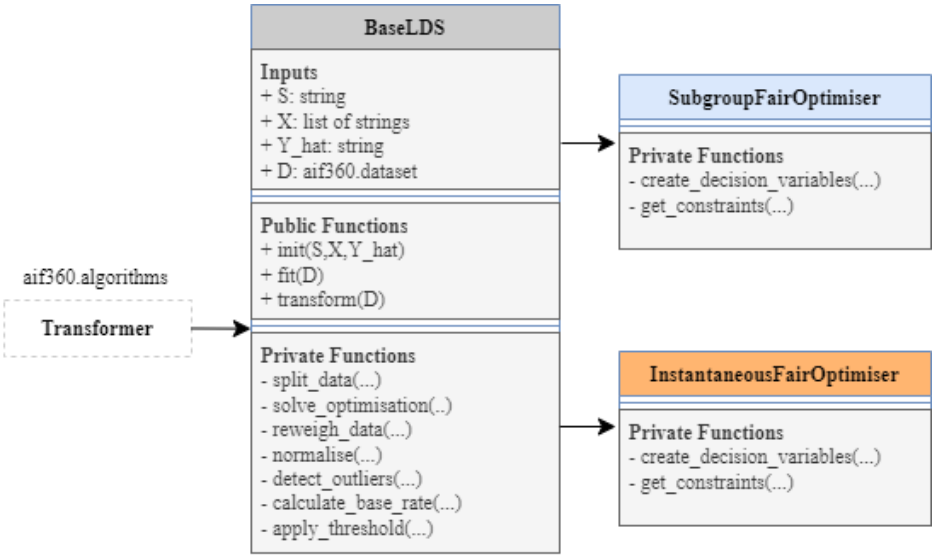


Figure 5.1: UML Diagram for SubgroupFair, InstantaneousFair and BaseLDS classes

5.1.2 Implementation

The algorithm was implemented within the AIF360 structure, taking dataset objects as inputs as is standard for the toolkit. The original method required the user to install an additional executable to use the SDPA solver [52], which reduced the ease of installation compared to alternate methods within AIF360. As a result, I tested the use of the alternate CVXPY solver [53], as it could be installed automatically alongside existing Python dependencies. The runtime performance, shown in Figure 5.3, shows that the computational cost of this solver was significantly greater and proved to increase exponentially as dataset size increased. As such, users are given a choice between using the CVXPY solver, or if a path to the SDPA executable is provided, it is used instead.

5.1.3 Documentation

In line with AIF360's contribution requirements, I wrote sphinx-compatible docstrings for the initialisation, fit, and transform functions, detailing each parameter to ensure users can select the appropriate S, X and $\hat{Y}$ attributes. The use of the optional SPDA solver was also detailed. Comments were written to explain complex operations like optimisation and reweighting processes. A demonstrative Jupyter notebook was also created. Plotting code in line with plots from Section 5.2.4 were included to demonstrate how the results can be visualised.

5.2 Testing and Evaluation

Once implemented, thorough testing was performed to validate my work and further assess the algorithms beyond their original scope.

5.2.1 Usability Testing

To assess documentation quality, I asked ten peers to apply the algorithms using template Jupyter files and the provided documentation and demonstration file. Users selected varying, but appropriate attributes for S and X and correctly selected the $\hat{Y}$ attribute of income. Although some users relied solely on the demonstration file, all successfully implemented the algorithms in under 30 minutes, validating the documentation quality. Some users found the optional usage of the SPDA solver confusing, instead opting to

use the slower CVXPY solver. While this validated the need for an alternate method to increase accessibility, documentation was updated to detail the installation and usage of the tool.

5.2.2 Evaluation Metrics

As in [54], algorithm performance was measured using the following metrics:

Independence (*IND*) , shown in Equation (5.3), measures the difference in the probability of a favourable outcome, $\hat{Y}_{repaired} = 1$, between the two sensitive attribute groups $s = 0$ and $s = 1$ after applying the fairness algorithm. A lower IND value indicates that the algorithm has reduced the disparity in favourable outcomes between the groups, making the predictions more independent of the sensitive attribute. Ideally, IND should be close to zero for a fair algorithm.

$$\text{IND} = |P(\hat{Y}_{repaired} = 1 \mid s = 0) - P(\hat{Y}_{repaired} = 1 \mid s = 1)| \tag{5.3}$$

Separation (*SP*) , shown in Equation (5.4), measures the difference in the probability of the repaired outcome differing from the original outcome, conditioned on the sensitive attribute and the original predicted outcome. It calculates this difference for both cases whereby the original outcome was unfavourable and favourable and for both sensitive attribute groups $s = 0$ and $s = 1$. A lower SP value indicates that the algorithm has maintained consistency in its repairs across the sensitive groups and original predicted outcomes.

$$\begin{aligned}\text{SP} = &|P(\hat{Y}_{repaired} = 0 \mid \hat{Y}_{unrepaired} = 1, s = 0) - P(\hat{Y}_{repaired} = 0 \mid \hat{Y}_{unrepaired} = 1, s = 1)| \\ &+ |P(\hat{Y}_{repaired} = 1 \mid \hat{Y}_{unrepaired} = 0, s = 0) - P(\hat{Y}_{repaired} = 1 \mid \hat{Y}_{unrepaired} = 0, s = 1)|\end{aligned} \tag{5.4}$$

Sufficiency (*SF*) , shown in Equation (5.5) measures the difference in the probability of the original outcome being favourable, conditioned on the repaired outcome and sensitive attribute. It calculates this difference for both cases whereby the repaired outcome is favourable and unfavourable and for both sensitive attribute groups $s = 0$ and $s = 1$. A lower SF value indicates that the repaired outcomes are sufficient to determine the original outcomes, regardless of the sensitive attribute.

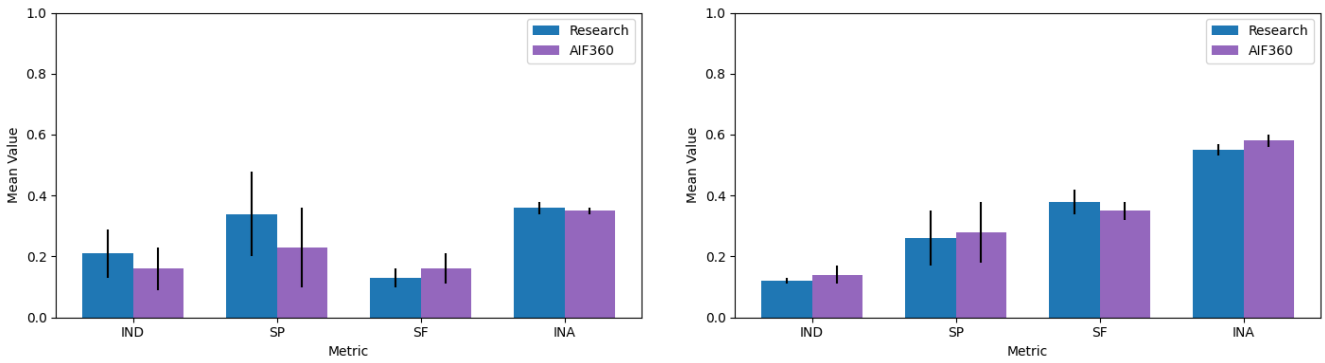
$$\begin{aligned}\text{SF} = &|P(\hat{Y}_{unrepaired} = 1 \mid \hat{Y}_{repaired} = 1, s = 0) - P(\hat{Y}_{unrepaired} = 1 \mid \hat{Y}_{repaired} = 1, s = 1)| \\ &+ |P(\hat{Y}_{repaired} = 0 \mid \hat{Y}_{unrepaired} = 0.s = 0) - P(\hat{Y}_{repaired} = 0 \mid \hat{Y}_{unrepaired} = 0, s = 1)|\end{aligned} \tag{5.5}$$

Inaccuracy (*INA*) , shown in Equation (5.6), measures the overall probability of the repaired outcome differing from the original outcome. It quantifies how much the fairness algorithm has altered the original predictions. A lower INA value indicates that the algorithm has made fewer changes to the original predictions while still achieving fairness. Ideally, the algorithm should minimise INA while satisfying the other fairness metrics.

$$\text{INA} = P(\hat{Y}_{unrepaired} \neq \hat{Y}_{repaired}), \tag{5.6}$$

5.2.3 Reproducibility Testing

Tests run under the same parameters as the source algorithm validated my implementation, achieving results that were within acceptable margins given the standard deviations shown in (Figure 5.2).



(a) Subgroup Fairness **(b)** Instantaneous Fairness

Figure 5.2: Comparison of results between the source algorithm and my implementation

5.2.4 Evaluation of Subgroup Fair and Instantaneous Fair Algorithm

Runtime Comparison Between SDPA and CVXPY solver

The CVXPY solver had significantly higher computational cost than the original SDPA solver, as expected from evaluations in [7]. Further experimentation also showed that the CVXPY solver had a time complexity of $O(n^2)$ with regard to the size of the dataset, compared to $O(n)$ using the SDPA solver. The difference in runtime for a $\mathcal{D}_{train}$ size of 20% can be seen in Figure 5.3.

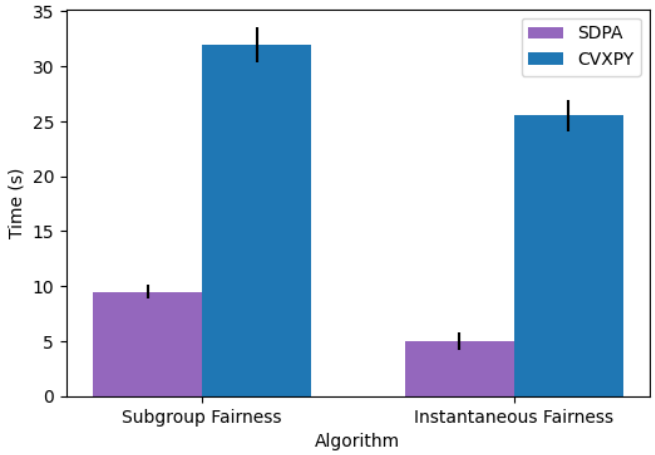
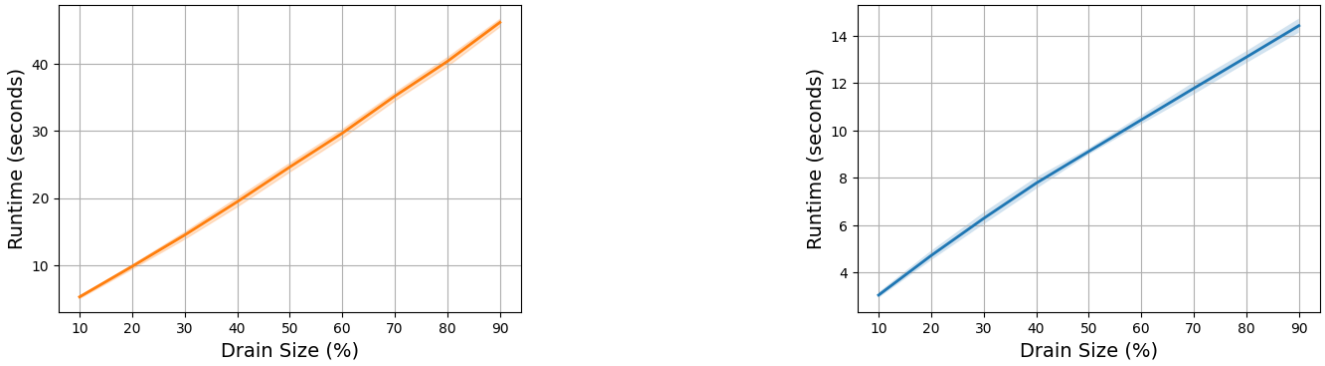


Figure 5.3: Runtime performance of Subgroup and Instantaneous Fairness using the SDPA and CVXPY optimisation solver libraries

Effect of Dataset Sizes on Subgroup and Instantaneous Fairness Algorithms

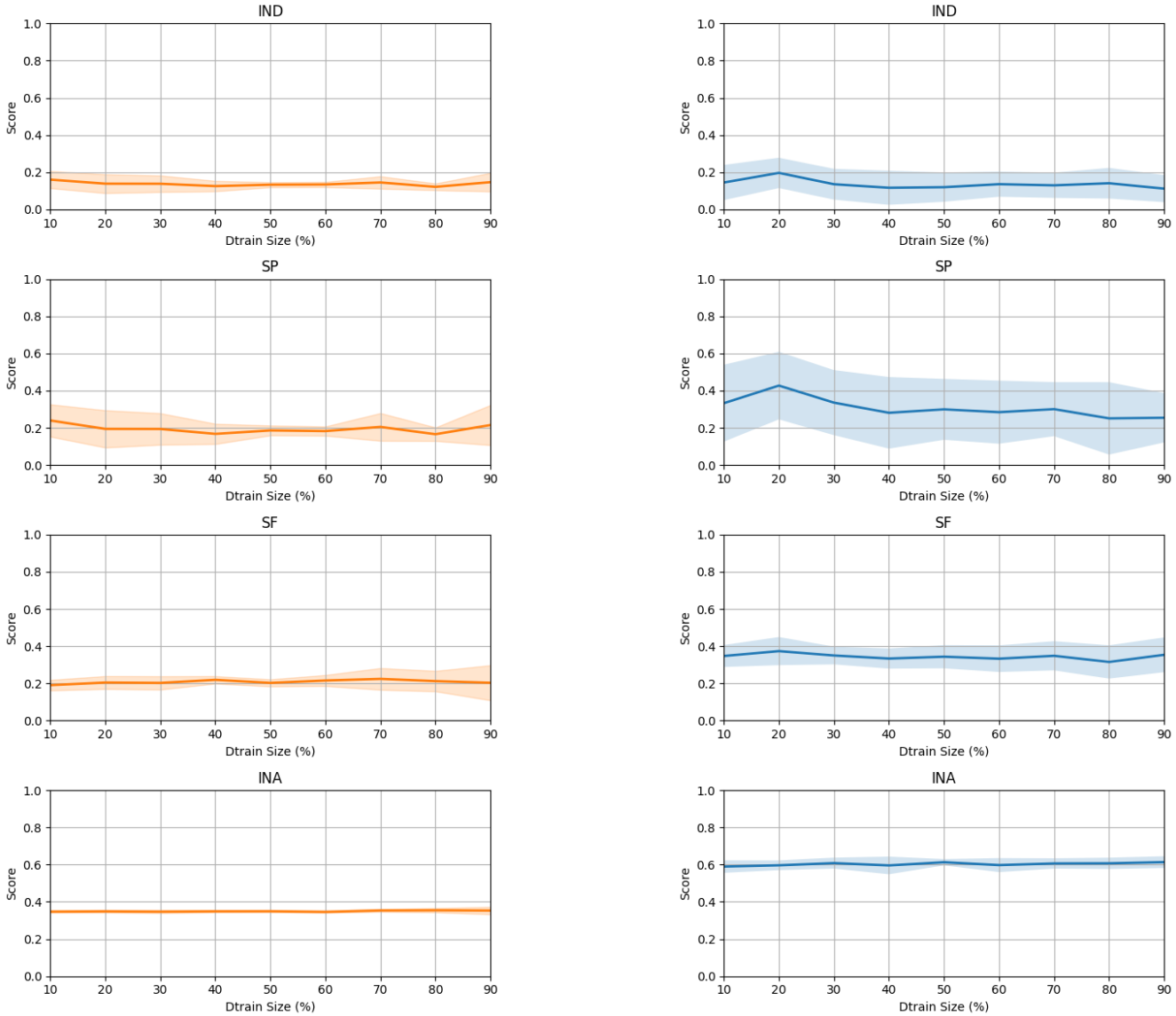
Algorithm runtimes were evaluated against $\mathcal{D}train$ and $\mathcal{D}test$ sizes, with varied split percentages, shown in Figure 5.4. Results show that for both algorithms, the runtime had a complexity of $O(n)$ with regard to the size of the test dataset

Figure 5.5 shows the effect of $\mathcal{D}train$ size on the algorithms' respective metrics. The algorithms performed comparably for all sizes, although the standard deviation was often marginally greater for lower $\mathcal{D}train$. Still, this is caused by the increased variation in $\mathcal{D}train$ due to its smaller size. The consistency of the results across varied training sizes shows the algorithm's resilience to small training sets, showing that effective remediation can be achieved even when compute cost is constrained.



(a) Subgroup Fairness **(b)** Instantaneous Fairness

Figure 5.4: Runtimes of fit and transform functions within the Subgroup and Instantaneous Fairness algorithms over varies $\mathcal{D}_{test}$ sizes



(a) Subgroup Fairness **(b)** Instantaneous Fairness

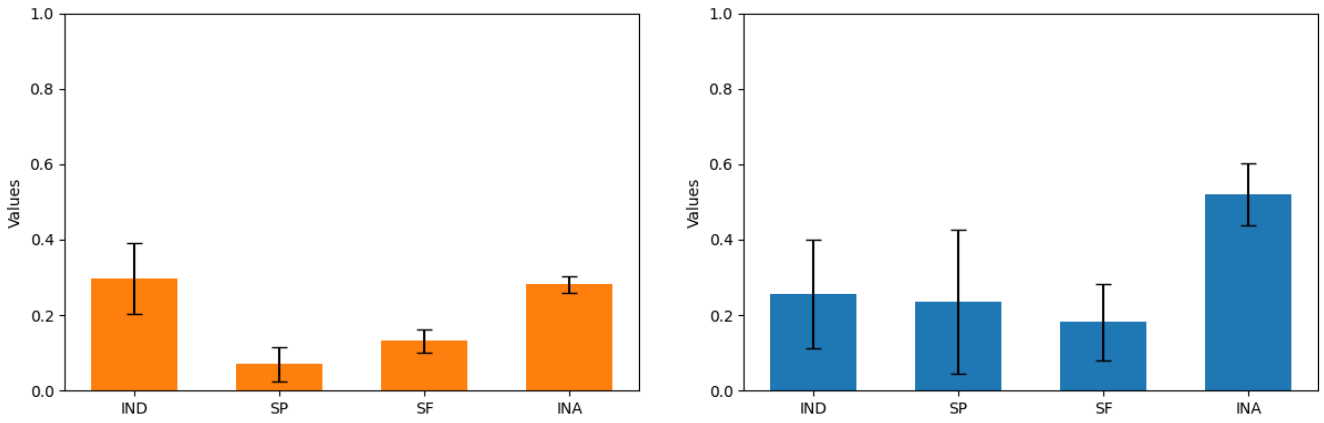
Figure 5.5: Metric performance of Subgroup and Instantaneous Fairness algorithms over varied D_{test} sizes

Evaluation of Applying the Algorithm to the Adult Dataset

The subgroup and instantaneous fair algorithms were applied to the adult to test whether the approach was suitable for other applications rather than COMPAS. The following parameters were used:

- 10% of data used to train the model
- S = sex
- X = age, hours of work per week, education number
- Y = annual income over $50,000$ *(binary)*

The results (Figure 5.6) show that comparable scores for the assessed metrics were achieved for both the COMPAS and Adult datasets, showing that the method applies to other datasets.



(a) Subgroup Fairness

(b) Instantaneous Fairness

Figure 5.6: Metric performance of Subgroup and Instantaneous Fairness algorithms applied to the Adult Dataset

5.3 Discussion

Through my work, I successfully implemented the Subgroup and Distributional Fairness algorithms within the AIF360 toolkit. The alternate CVXPY solver's additional capability allows users to choose between runtime and ease of installation. Implementing the algorithm strengthened my understanding and ability to solve complex multivariate optimisation problems within Python.

By generalising the algorithms, evaluation beyond the scope of the original work[7] shows that the algorithm is robust to varied dataset sizes and alternate datasets with different properties and relationships between attributes.

Both algorithms were limited by their scope of considering only two subgroups at once. Future work should aim to expand on the algorithms, improving the optimisation processes to account for a scalable number of subgroups, although this will incur additional computational costs.

Chapter 6

Distributional Repair Algorithm

The development and evaluation of the Distributional Repair algorithm [8] is detailed in this chapter, documenting the implementation and evaluation process.

6.1 Development

Like the Subgroup and Instantaneous Fair algorithms in Chapter 5, I followed the development process outlined in Chapter 4.

6.1.1 Algorithm Redesign and Refactoring

The code [8] was decomposed into separate processes. For each process, hard-coded strings were replaced with appropriately named variables. In keeping with the AIF360 standards for preprocessing algorithms[55], I made sure to separate the algorithm into a fit and transform function, along with an initialisation function as is standard for a class within Python. Additional private subclasses were used to abstract the algorithm, improving readability and maintainability. Algorithms of subfunctions can be found in Appendix B.1.

When initialised, the class takes S,U,$\hat{Y}$ as strings, and X as a list of strings. These values are then stored for later use, as shown in Algorithm 4.

Algorithm 4 Initialisation

procedure INITIALISE($S, U, X, \hat{Y}$)
  store $S, U, X, \hat{Y}$
end procedure

The process begins by splitting the dataset, $\mathcal{D}$ into a research dataset, $\mathcal{D}_{research}$ used to learn the OT plan, and an archive dataset, $\mathcal{D}_{archive}$ to which remediation is applied.

The fit function, Algorithm 5, takes $\mathcal{D}_{research}$ and the number of distributional supports, n_q as its input. The algorithm splits the dataset according to its column names under the $S, U, X, \hat{Y}$ values, then iterates through X attributes and unique u values within the dataset. For continuous features of X, supports of resolution n_q are constructed over the distribution, followed by a probability density function *(PDF)* for both the privileged and unprivileged groups. The barycentre between the distributions is determined, and an OT plan is derived to drive data from its original distribution towards the barycentre. Probability mass functions *(PMF)* are calculated for discrete features, from which an OT plan can then be derived. Finally, the data needed to implement the OT plans are stored within the object for later use.

Algorithm 5 Fit

procedure FIT($\mathcal{D}_{research}, n_q$)
  load $S, U, X, \hat{Y}$
  $\mathcal{S}_{research}, \mathcal{U}_{reseach}, \mathcal{X}_{research}, \hat{\mathcal{Y}}_{research} \leftarrow$ SPLITDATASET($\mathcal{D}_{research}, S, U, X, \hat{Y}$)
  for $feat \in X$
    for $u_{val} \in$ UNIQUE($\mathcal{U}_{reseach}$)
      if $feat$ is continuous **then**
        $support \leftarrow$ GETSUPPORT($feat, u_{val}, n_q$)
        $pdf_0, pdf_1 \leftarrow$ GETCONTINUOUSPROBABILITYDENSITYFUNCTIONS($feat, u_{val}$)
        $barycenter \leftarrow$ GETBARYCENTER($pdf_0, pdf_1, feat, u_{val}$)
        $T_0, T_1 \leftarrow$ GETCONTINUOUSTRANSPORTPLANS($pdf_0, pdf_1, barycenter, feat, u_{val}$)
        store $support, pdf_0, pdf_1, T_0, T_1$
      else
        if ISVALIDDATA(u_{val}) **then**
          $pmf_0, pmf_1 \leftarrow$ GETDISCRETEPROBABILTYMASSFUNCTIONS($feat, u_{val}$)
          $T \leftarrow$ GETDISCRETETRANSPORTPLAN(pmf_0, pmf_1)
          store pmf_0, pmf_1, T
        else
          store pmf_0, pmf_1 as **None**
        end if
      end if
    end for
  end for
  return $self$
end procedure

Once the OT plans are formulated, the transform function implements the repair on either $\mathcal{D}_{research}$ or $\mathcal{D}_{archive}$. As shown in Algorithm 6, a repaired $\mathcal{X}$, $\tilde{\mathcal{X}}$ is created. The OT plan is retrieved and applied for each feature within X and each unique value within U. Finally, $\tilde{\mathcal{X}}$ replaces the $\mathcal{X}$ columns to produce the transformed dataset $\tilde{\mathcal{D}}$, which has now been repaired under the notion of conditional independence.

Algorithm 6 Transform

procedure TRANSFORM($\mathcal{D}$)
  load $S, U, X, \hat{Y}$
  $\mathcal{S}, \mathcal{U}, \mathcal{X}, \hat{\mathcal{Y}} \leftarrow$ SPLITDATASET($\mathcal{D}, S, U, X, \hat{Y}$)
  $\tilde{\mathcal{X}} \leftarrow \mathcal{X}$
  for $feat \in X$
    for $u_{val} \in$ UNIQUE($\mathcal{U}$)
      if $feat$ is continuous **then**
        $support \leftarrow$ GETSTOREDSUPPORT($feat, u_{val}$)
        $T_0 \leftarrow$ GETSTOREDTRANSPORTPLAN($feat, u_{val}, 0$)
        $T_1 \leftarrow$ GETSTOREDTRANSPORTPLAN($feat, u_{val}, 1$)
        $\tilde{\mathcal{X}} \leftarrow$ CONTINUOUSTRANSFORM($\mathcal{S}, \mathcal{U}, \mathcal{X}, feat, \tilde{\mathcal{X}}, u_{val}, support, T_0, T_1$)
      else
        $\tilde{\mathcal{X}} \leftarrow$ DISCRETETRANSFORM($\mathcal{S}, \mathcal{U}, \mathcal{X}, feat, \tilde{\mathcal{X}}, u_{val}$)
      end if
    end for
  end for
  $\tilde{\mathcal{D}}_{archive} \leftarrow$ JOINCOLUMNS($\mathcal{S}_{archive}, \mathcal{U}_{archive}, \tilde{\mathcal{X}}_{archive}, \hat{\mathcal{Y}}_{archive}$)
  return $\tilde{\mathcal{D}}_{archive}$
end procedure

Implementing the Distributional Repair algorithm was less complex than the Subgroup and Instantaneous Fairness algorithms. The original implementation used libraries that could be automatically installed alongside AIF360 through the dependency list, and as such, no new libraries were needed to be implemented. The algorithms were updated to take and return dataset objects from the AIF360 library instead of data frames from the panda's library. The class was updated to inherit from the abstract Transformer class from within AIF360, as is the standard for algorithms within AIF360 that handle dataset objects. The final implementation structure can be seen in Figure 6.1, represented as a UML model.

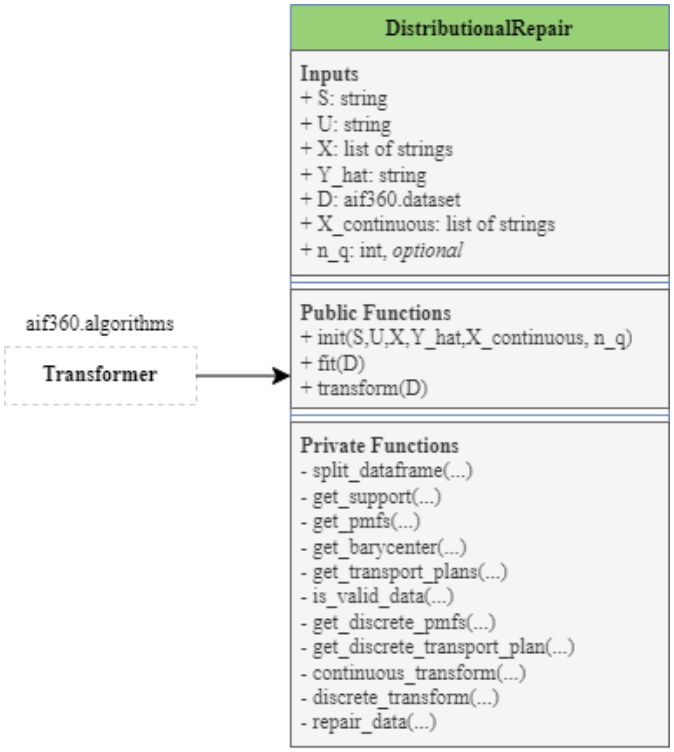


Figure 6.1: UML Diagram for the DistributionalRepair class

6.1.2 Documentation

In line with the requirements for contributing to AIF360, sphinx-compatible docstrings were written for the initialisation, fit, and transform functions. These detailed each parameter, ensuring the user can select the appropriate S, U, X and $\hat{Y}$ attributes. Additional comments were added to the code to explain complex operations, such as creating the OT plan and using the barycentre.

A demonstration file was also written as a Jupyter notebook consisting of Markdown and Python code, explaining the algorithm's core concepts and implementation. Plots akin to those in Section 6.2.4 were also included, demonstrating to users how to measure and visualise the repair performance for their use case.

6.2 Testing and Evaluation

Following the implementation of the algorithm, evaluation and validation of my work was performed following the methodology in Chapter 4.

6.2.1 Usability Testing

The documentation quality was tested by challenging ten peers to apply the algorithms. Template Jupyter files were written to load the necessary imports, along with the functions for plotting the results and

loading the data, as knowledge of this is available to users of the AIF360 toolkit. Users were tasked with writing the code needed to assign S, U, X and $\hat{Y}$ parameters, initialising the class, and using the fit and transform function, with the aid of the written documentation and demonstration file. The COMPAS dataset was chosen for this task, as the demonstration file already implemented the Adult dataset. Although users selected varying attributes for S, U and X, their selections were all appropriate. All users correctly selected the $\hat{Y}$ of if the individual will reoffend. As with the usability tests for Subgroup and Instantaneous Fairness algorithms, some users ignored the documentation page and implemented the algorithm based on the demonstration file alone. All users successfully implemented the algorithms in under 20 minutes, validating the quality of the documentation. Some users questioned if attributes belonged under the U or X label. As such, the documentation was updated to clarify that the X attributes were used by the model, being specific to the use case, not the dataset.

6.2.2 Evaluation Metrics for Distributional Repair

The Kullback-Leibler divergence *(KLD)* [56][8] was used to evaluate the performance of the Distributional Repair algorithm. The difference between the reweighted distributions where $s = 1$ and $s = 0$ within $\tilde{\mathcal{D}}$ is determined and used to measure conditional independence, as shown in Equation (6.1), utilising the Equation (3.6) and Equation (3.5).

$$\text{U-Mean KLD} = \frac{1}{2}\mathrm{D}_{\mathrm{KL}}\left[f(x \mid y=0,u) | f(x \mid y=1,u)\right] + \frac{1}{2}\mathrm{D}_{\mathrm{KL}}\left[f(x \mid y=1,u) | f(x \mid y=0,u)\right] \tag{6.1}$$

6.2.3 Reproducibility Testing

To ensure that my implementation and the original had the same behaviour, the algorithm was run on the Adult dataset under the same parameters to validate my implementation. 100 iterations of tests were run for both, with the results in Figure 6.2 showing that equivalent results were achieved, thus validating my work. The randomised data splits account for the slight deviation.

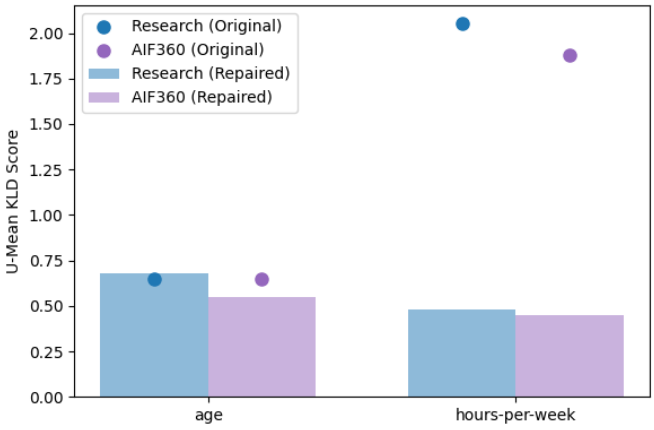


Figure 6.2: Comparison of results between the source algorithm and my implementation.

6.2.4 Evaluation of Distributional Repair Algorithm

Performance Against Different S

In the original work, $S =$ sex was used, which showed promising results for the effectiveness of increasing conditional independence. To ensure that the algorithm scaled well to other S attributes, a test was performed using $S =$ race. As before, the algorithm effectively improved conditional independence in both $\tilde{\mathcal{D}}_{research}$ and $\tilde{\mathcal{D}}_{archive}$. The results can be compared in Figure 6.3. For both tests, the following were used:

- U = college educated *(binary)*
- X = age, hours of work per week
- Y = annual income over $50,000 *(binary)*

200 tests were run, with randomised splits between $\tilde{\mathcal{D}}_{research}$ and $\tilde{\mathcal{D}}_{archive}$ to assess the robustness of the algorithm regardless of how it is split. $\tilde{\mathcal{D}}_{research}$ lengths of 10,000 were used. The low standard range, indicated by the bars in Figure 6.3, shows that regardless of the split, consistent results were achieved that increased conditional dependence for all X attributes across both datasets.

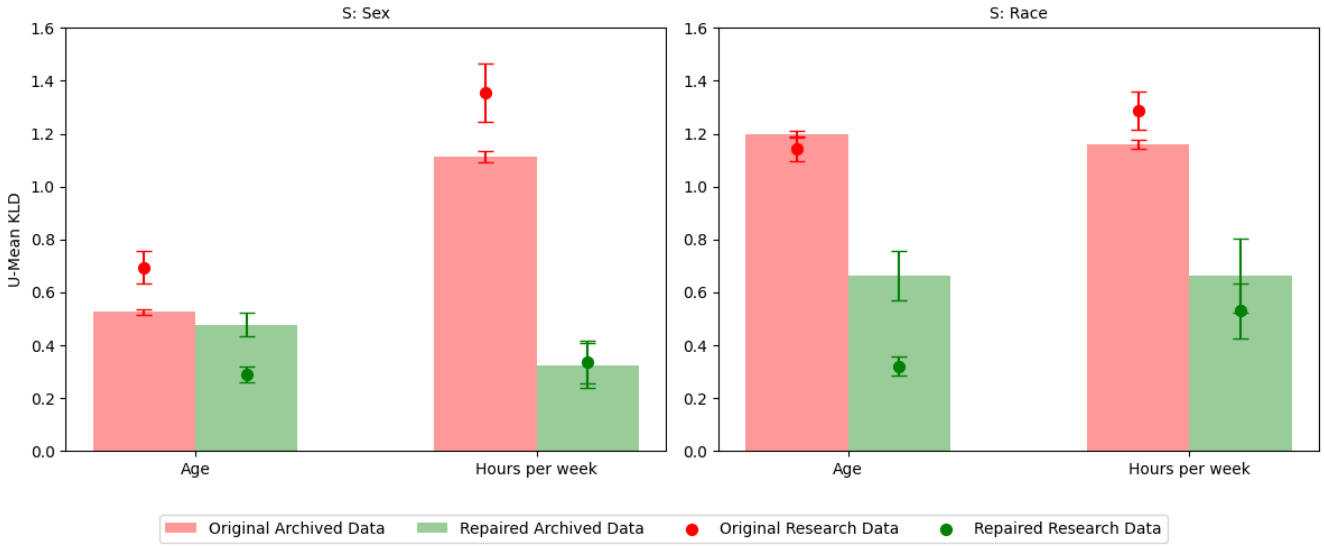


Figure 6.3: KLD fairness testing on $\tilde{\mathcal{D}}_{research}$ and $\tilde{\mathcal{D}}_{archvie}$ under varied S attributes.

Assessing Performance on the COMPAS Dataset

The COMPAS dataset assessed whether the algorithm was effective on alternate datasets. The COMPAS dataset provided a good test as it features well-formatted data, yet a smaller size of only 7,214 rows compared to the 48,842 in the Adult dataset. It was tested separately with S = race and S = sex. Other attributes were as follows:

- U = charge degree
- X = age, total priors count, decile score
- Y = will re-offend *(binary)*

Due to the smaller size, a smaller $\mathcal{D}_{research}$ size of 5,000 was used. The results, shown in Figure 6.4, show that the algorithm reduced the net U-Mean KLD; however, for some X attributes, U-Mean KLD increased. This indicated that the performance was worse, most likely due to the smaller $\mathcal{D}_{research}$ size - behaviour that matches both my testing in Section 6.2.4 and Langbridge's research [8].

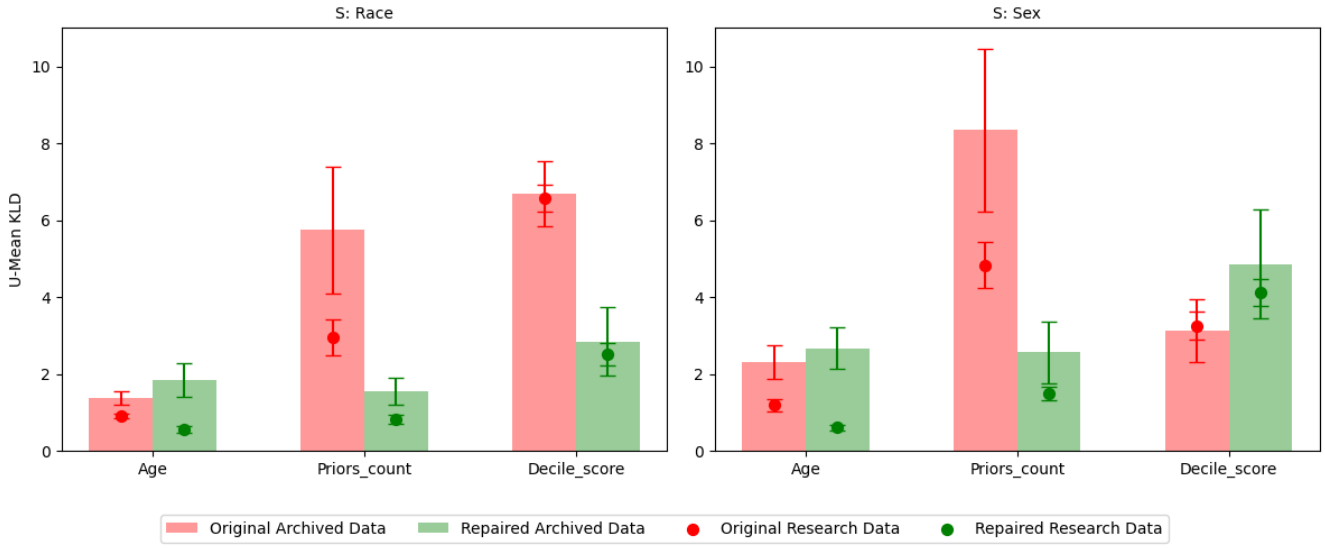


Figure 6.4: KLD fairness testing performed on the COMPAS dataset

Effect of Research Dataset Size

200 iterations of tests were performed on the Adult dataset for varied $\mathcal{D}_{research}$ sizes between 1,000 and 15,000. These tests used the same parameters as the $S =$ sex case in Figure 6.3.

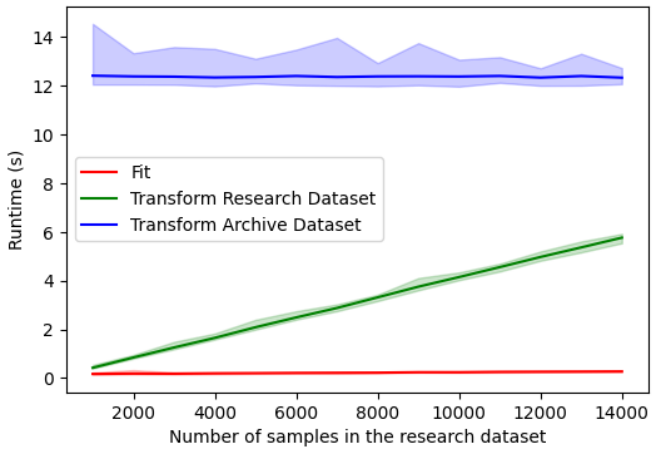


Figure 6.5: Runtime of each Distributional Repair function over varies $\mathcal{D}_{research}$ sizes

Observing the runtimes in Figure 6.5, it is shown that fit and transform functions have time complexities of $O(1)$ and $O(n)$, respectively, when considering the number of samples in the input dataset.

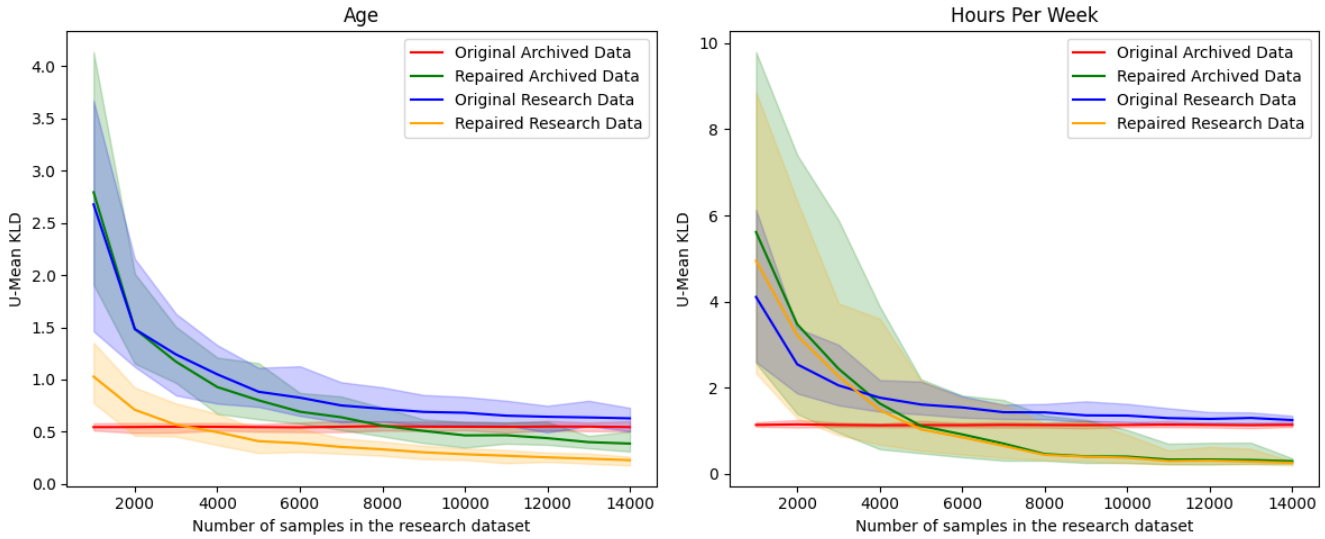
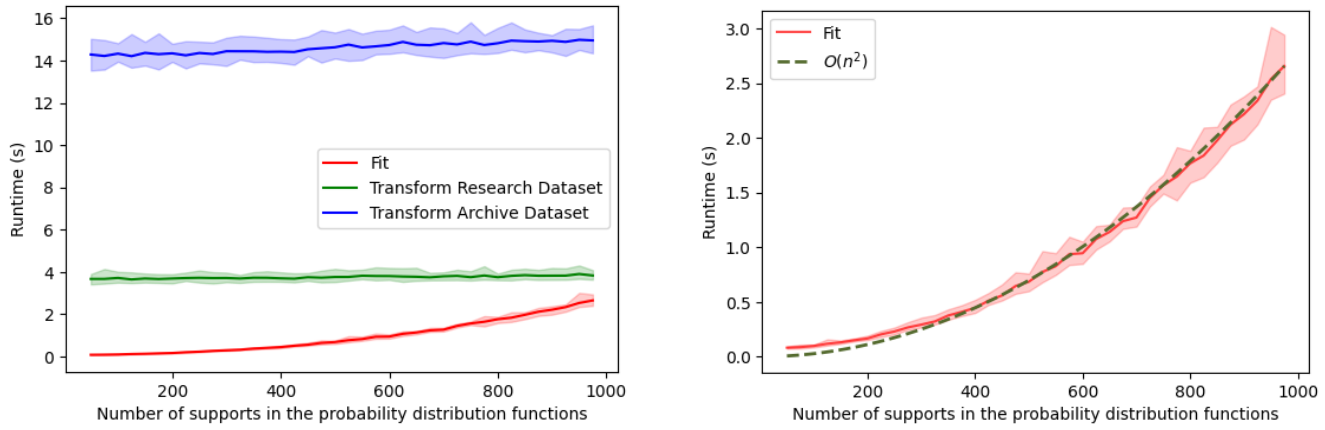


Figure 6.6: KLD fairness values for the original and repaired $\mathcal{D}_{research}$ and $\mathcal{D}_{archive}$

Figure 6.6, shows that the performance of the algorithm is proportional to $|\mathcal{D}_{research}|^{-1}$, with a larger $\mathcal{D}_{research}$ giving better results. Effective remediation begins when the repaired lines intersect the original lines for their respective dataset. For the given inputs, we can see that to improve the U-Mean KLD for both X values, a dataset size of at least 8,000 is needed for this specific configuration.

Effect of the Number of Probability Distribution Supports

Testing was performed to evaluate the impact of changing the number of supports, n_q, on the runtime and the U-Mean KLD performance. 200 iterations of tests were run for values of n_q between 50 and 1000. For these tests, the same parameters as the $S =$ sex case in Figure 6.3, with a $\mathcal{D}_{archive}$ size of 10,000.



(a) Fit and Transform runtimes

(b) Fit runtime with overlaid $O(n^2)$ curve

Figure 6.7: Runtimes of each Distributional Repair function

Figure 6.7 shows that the fit and transform functions have runtime complexities of $O(n^2)$ and $O(n)$ respectively, with regards to n_q.

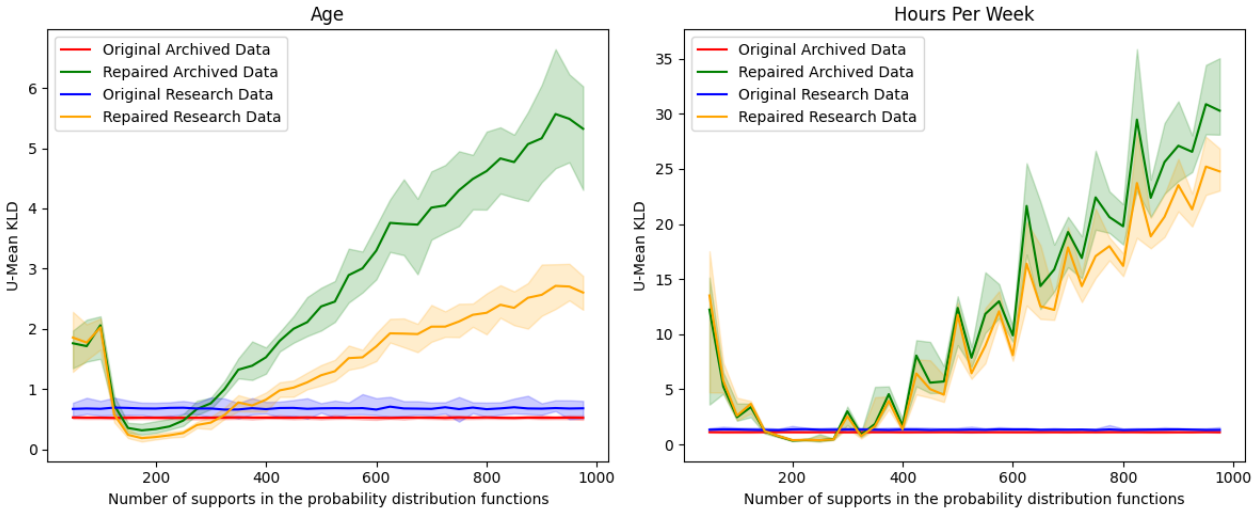


Figure 6.8: KLD fairness values for the original and repaired $\mathcal{D}_{archvive}$ and $\mathcal{D}_{research}$

Plotting the U-Mean KLD for the original and repaired $\mathcal{D}_{archvive}$ and $\mathcal{D}_{research}$, as shown in Figure 6.8, indicates an optimal number of supports for each feature in X. The algorithm currently allows the user to define a single n_q for all X attributes, or if undefined, a default value of 250 was used; however, this suggests that there is scope for future improvement to the algorithm by automatically finding an optimal value of n_q.

Exploring the Relationship Between the Size of the Research Dataset and the Number of Probability Distribution Supports

The relationship between the number of supports and the size of the research dataset was evaluated. As shown in Figure 6.9, the optimal number of supports for a given research dataset size increases with the number of samples; however, as in Figure 6.8, this relationship is inconsistent across X attributes.

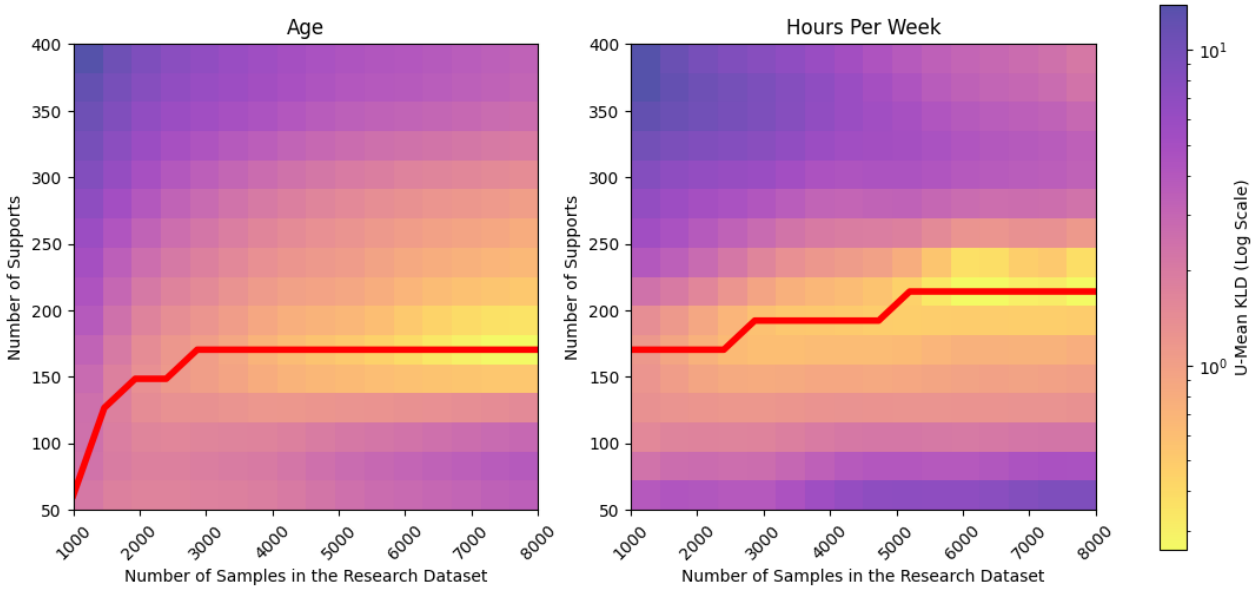


Figure 6.9: KLD Fairness Values of $\tilde{\mathcal{D}}_{archive}$. The red line shows the optimal n_q for each $\mathcal{D}_{archive}$.

As before, this shows future scope to automatically determine optimal n_q values for each X attribute independently.

6.3 Discussion

My work successfully implemented the Distributional repair algorithm, making the tool accessible to a broad audience of AI practitioners. Generalising and adapting the algorithm enabled me to understand better the underlying concepts of optimal transport and probability functions.

The Distributional repair algorithm was deemed a highly effective means of remediating unfairness within a dataset. Its preprocessing nature aids in increasing the explainability of the models, as practitioners do not need to understand any processes within black box models. This approach is also beneficial when developing AI systems, as the process needs to be applied only to the dataset once rather than to the model or result at each trial.

The model performed highly across all tests, delivering solid results regardless of the dataset or attributes used. Weaknesses of the model regarding small dataset sizes or incorrect n_q values as outlined in [8] were also reinforced, with these caveats made clear to users within the demonstration document.

As proposed, there is scope for future algorithm development through the automated optimisation of independent n_q values for each X attribute. As the whole process must be applied to determine the KLD values when applied to $\mathcal{D}_{archive}$, an iterative approach would be a suitable solution. One such method may be to use stochastic gradient descent with momentum[57] due to its computational efficiency and ability to overcome local minima as present in Figure 6.8 and Figure 6.9.

Overall, I found the distributional repair algorithm to be a strong approach to remediating unfairness within a dataset. It provides consistent and positive results throughout tests whilst incurring a relatively low computational cost.

Chapter 7

Conclusion

7.1 Summary of Work

In conclusion, my work successfully met the goals outlined in Chapter 1.

- Accessible explanations of the algorithms concepts and processes were effectively detailed within this report and the code documentation.
- The research algorithms were converted into general-purpose tools within the AIF360 framework, allowing them to be used by AI practitioners within their own AI fairness pipelines and successfully validated against the research algorithms.
- Clear documentation of these tools was created, and their clarity and effectiveness were validated through user testing.
- I effectively evaluated parameter variance on all algorithms through informative data visualisations, providing new insights into the parameters' impact on model performance and computational cost.

As a result of my contribution to the field of AI fairness, Subgroup Fairness, Instantaneous Fairness, and Distributional Repair algorithms are now accessible to a broad audience of AI practitioners.

7.2 Recommendations For Future Work

In addition to the future scope of the implemented algorithms detailed in Section 5.3 and Section 6.3, I have identified recommendations for future AI fairness algorithm researchers. I recommend that strong coding practices be used when developing new algorithms. Object-orientated structures should be used, along with well-named and sufficiently documented subfunctions, to create both readable and maintainable code. Hard coded values should be avoided, as they limit the algorithm's real-world usage and the evaluation that can be performed on such algorithms, leaving many assumptions regarding scalability unanswered. Additionally, algorithm designers should aim to work within an established toolkit such as AIF360, allowing them to utilise the pre-existing methods for tasks such as metric measurement, thus reducing the development time of new algorithms.

Bibliography

[1] Janine S. Hiller et al. “Fairness in the Eyes of the Beholder: AI; Fairness; and Alternative Fairness in the Eyes of the Beholder: AI; Fairness; and Alternative Credit Scoring Credit Scoring”. In: 2021. URL: https://www.semanticscholar.org/paper/Fairness-in-the-Eyes-of-the-Beholder%3A-AI%3B-Fairness%3B-Hiller-Tsamados/12b1de223c84b90c7c1fe096f2a77aacd4941114 (visited on 06/06/2024) (cit. on p. 4).

[2] Lou Therese Brandner et al. “How Data Quality Determines AI Fairness: The Case of Automated Interviewing”. en. In: () (cit. on p. 4).

[3] Emilio Ferrara. “Fairness And Bias in Artificial Intelligence: A Brief Survey of Sources, Impacts, And Mitigation Strategies”. In: *Sci* 6.1 (Dec. 2023). arXiv:2304.07683 [cs], p. 3. ISSN: 2413-4155. DOI: 10.3390/sci6010003. URL: http://arxiv.org/abs/2304.07683 (visited on 06/06/2024) (cit. on p. 4).

[4] *AI Fairness 360 documentation aif360 0.6.1 documentation*. URL: https://aif360.readthedocs.io/en/latest/index.html (visited on 05/22/2024) (cit. on pp. 4, 8, 14, 15).

[5] *FATE: Fairness, Accountability, Transparency & Ethics in AI*. en-US. URL: https://www.microsoft.com/en-us/research/theme/fate/ (visited on 05/22/2024) (cit. on p. 4).

[6] *Google Responsible AI Practices*. en. URL: https://ai.google/responsibility/responsible-ai-practices/ (visited on 05/22/2024) (cit. on p. 4).

[7] Quan Zhou, Jakub Marecek, and Robert N. Shorten. “Fairness in Forecasting of Observations of Linear Dynamical Systems”. en. In: *Journal of Artificial Intelligence Research* 76 (Apr. 2023). arXiv:2209.05274 [cs, eess, math, stat], pp. 1247–1280. ISSN: 1076-9757. DOI: 10.1613/jair.1.14050. URL: http://arxiv.org/abs/2209.05274 (visited on 09/26/2023) (cit. on pp. 4, 9, 15, 20, 22).

[8] Abigail Langbridge, Anthony Quinn, and Robert Shorten. *Optimal Transport for Fairness: Archival Data Repair using Small Research Data Sets*. en. arXiv:2403.13864 [cs, math, stat]. Mar. 2024. URL: http://arxiv.org/abs/2403.13864 (visited on 05/09/2024) (cit. on pp. 4, 12, 23, 26, 27, 31).

[9] Richard Benjamins, Alberto Barbado, and Daniel Sierra. “Responsible AI by Design in Practice”. en. In: () (cit. on p. 5).

[10] *Equality and Non-discrimination - United Nations and the Rule of Law*. URL: https://www.un.org/ruleoflaw/thematic-areas/human-rights/equality-and-non-discrimination/ (visited on 11/29/2023) (cit. on p. 5).

[11] Surbhi Rathore. “MODEL AGNOSTIC FEATURE SELECTION FOR FAIRNESS”. en. PhD thesis. Kingston, RI: University of Rhode Island, 2022. DOI: 10.23860/thesis-rathore-surbhi-2022. URL: https://digitalcommons.uri.edu/theses/2291 (visited on 06/06/2024) (cit. on p. 5).

[12] *Discrimination: your rights*. en. URL: https://www.gov.uk/discrimination-your-rights (visited on 11/29/2023) (cit. on p. 5).

[13] Pranjal Awasthi et al. *Beyond Individual and Group Fairness*. en. arXiv:2008.09490 [cs, stat]. Aug. 2020. URL: http://arxiv.org/abs/2008.09490 (visited on 06/04/2024) (cit. on p. 6).

[14] Candice Schumann et al. “We Need Fairness and Explainability in Algorithmic Hiring”. en. In: *New Zealand* (2020) (cit. on p. 6).

[15] Aislinn Kelly-Lyth. "Challenging Biased Hiring Algorithms". In: *Oxford Journal of Legal Studies* 41.4 (Dec. 2021), pp. 899–928. ISSN: 0143-6503. DOI: 10.1093/ojls/gqab006. URL: https://doi.org/10.1093/ojls/gqab006 (visited on 05/22/2024) (cit. on p. 6).

[16] Kevin Credit. "Neighbourhood inequity: Exploring the factors underlying racial and ethnic disparities in COVID19 testing and infection rates using ZIP code data in Chicago and New York". en. In: *Regional Science Policy & Practice* 12.6 (Dec. 2020), pp. 1249–1271. ISSN: 1757-7802, 1757-7802. DOI: 10.1111/rsp3.12321. URL: https://rsaiconnect.onlinelibrary.wiley.com/doi/10.1111/rsp3.12321 (visited on 05/22/2024) (cit. on p. 6).

[17] *Article 8 - Law No. 78-17 of January 6, 1978 relating to computing, files and freedoms - Légifrance*. URL: https://www.legifrance.gouv.fr/loda/article_lc/LEGIARTI000037090124/2019-03-18 (visited on 05/23/2024) (cit. on p. 7).

[18] *Race Policy in France*. en-US. URL: https://www.brookings.edu/articles/race-policy-in-france/ (visited on 05/23/2024) (cit. on p. 7).

[19] Arthur Charpentier. "Group Fairness". en. In: *Insurance, Biases, Discrimination and Fairness*. Ed. by Arthur Charpentier. Cham: Springer Nature Switzerland, 2024, pp. 309–355. ISBN: 978-3-031-49783-4. DOI: 10.1007/978-3-031-49783-4_8. URL: https://doi.org/10.1007/978-3-031-49783-4_8 (visited on 05/22/2024) (cit. on p. 7).

[20] Thomas E. Weisskopf. "Is Positive Discrimination a Good Way to Aid Disadvantaged Ethnic Communities?" en. In: *Handbook on Economics of Discrimination and Affirmative Action*. Ed. by Ashwini Deshpande. Singapore: Springer Nature Singapore, 2023, pp. 699–717. ISBN: 978-981-19416-5-8 978-981-19416-6-5. DOI: 10.1007/978-981-19-4166-5_44. URL: https://link.springer.com/10.1007/978-981-19-4166-5_44 (visited on 05/23/2024) (cit. on p. 7).

[21] Mike Noon. "The shackled runner: time to rethink positive discrimination?" In: *Work, Employment and Society* 24.4 (Dec. 2010). Publisher: SAGE Publications Ltd, pp. 728–739. ISSN: 0950-0170. DOI: 10.1177/0950017010380648. URL: https://doi.org/10.1177/0950017010380648 (visited on 05/23/2024) (cit. on p. 7).

[22] Zhimeng Jiang et al. "GENERALIZED DEMOGRAPHIC PARITY FOR GROUP FAIRNESS". en. In: (2022) (cit. on p. 7).

[23] Richard Berk et al. "Fairness in Criminal Justice Risk Assessments: The State of the Art". en. In: *Sociological Methods & Research* 50.1 (Feb. 2021), pp. 3–44. ISSN: 0049-1241, 1552-8294. DOI: 10.1177/0049124118782533. URL: http://journals.sagepub.com/doi/10.1177/0049124118782533 (visited on 05/22/2024) (cit. on p. 7).

[24] *Linux Foundation - Decentralized innovation, built with trust*. en. URL: https://www.linuxfoundation.org (visited on 05/22/2024) (cit. on p. 8).

[25] Michael Feldman et al. *Certifying and removing disparate impact*. arXiv:1412.3756 [cs, stat]. July 2015. DOI: 10.48550/arXiv.1412.3756. URL: http://arxiv.org/abs/1412.3756 (visited on 05/22/2024) (cit. on p. 8).

[26] Rich Zemel et al. "Learning Fair Representations". en. In: *Proceedings of the 30th International Conference on Machine Learning*. ISSN: 1938-7228. PMLR, May 2013, pp. 325–333. URL: https://proceedings.mlr.press/v28/zemel13.html (visited on 05/22/2024) (cit. on p. 8).

[27] Flavio Calmon et al. "Optimized Pre-Processing for Discrimination Prevention". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/9a49a25d845a483fae4be7e341368e36-Abstract.html (visited on 05/22/2024) (cit. on p. 8).

[28] Faisal Kamiran and Toon Calders. "Data preprocessing techniques for classification without discrimination". en. In: *Knowledge and Information Systems* 33.1 (Oct. 2012), pp. 1–33. ISSN: 0219-1377, 0219-3116. DOI: 10.1007/s10115-011-0463-8. URL: http://link.springer.com/10.1007/s10115-011-0463-8 (visited on 05/22/2024) (cit. on p. 8).

[29] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. "Mitigating Unwanted Biases with Adversarial Learning". In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '18. New York, NY, USA: Association for Computing Machinery, Dec. 2018, pp. 335–340. ISBN: 978-1-4503-6012-8. DOI: 10.1145/3278721.3278779. URL: https://dl.acm.org/doi/10.1145/3278721.3278779 (visited on 05/22/2024) (cit. on p. 8).

[30] *Home*. en-US. URL: https://adversarial-robustness-toolbox.org/ (visited on 05/22/2024) (cit. on p. 8).

[31] Alekh Agarwal et al. *A Reductions Approach to Fair Classification*. arXiv:1803.02453 [cs]. July 2018. DOI: 10.48550/arXiv.1803.02453. URL: http://arxiv.org/abs/1803.02453 (visited on 05/22/2024) (cit. on p. 8).

[32] *An Empirical Study of Rich Subgroup Fairness for Machine Learning | Proceedings of the Conference on Fairness, Accountability, and Transparency*. URL: https://dl.acm.org/doi/10.1145/3287560.3287592 (visited on 05/22/2024) (cit. on p. 8).

[33] L. Elisa Celis et al. *Classification with Fairness Constraints: A Meta-Algorithm with Provable Guarantees*. arXiv:1806.06055 [cs, stat]. Apr. 2020. DOI: 10.48550/arXiv.1806.06055. URL: http://arxiv.org/abs/1806.06055 (visited on 05/22/2024) (cit. on p. 8).

[34] Toshihiro Kamishima et al. "Fairness-Aware Classifier with Prejudice Remover Regularizer". en. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Peter A. Flach, Tijl De Bie, and Nello Cristianini. Berlin, Heidelberg: Springer, 2012, pp. 35–50. ISBN: 978-3-642-33486-3. DOI: 10.1007/978-3-642-33486-3_3 (cit. on p. 8).

[35] Geoff Pleiss et al. "On Fairness and Calibration". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/b8b9c74ac526fffbeb2d39ab038d1cd7-Abstract.html (visited on 05/22/2024) (cit. on p. 8).

[36] Moritz Hardt et al. "Equality of Opportunity in Supervised Learning". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016. URL: https://papers.nips.cc/paper_files/paper/2016/hash/9d2682367c3935defcb1f9e247a97c0d-Abstract.html (visited on 05/22/2024) (cit. on p. 8).

[37] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. "Fairness-Aware Ranking in Search & Recommendation Systems with Application to LinkedIn Talent Search". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. arXiv:1905.01989 [cs]. July 2019, pp. 2221–2231. DOI: 10.1145/3292500.3330691. URL: http://arxiv.org/abs/1905.01989 (visited on 05/22/2024) (cit. on p. 8).

[38] *Decision Theory for Discrimination-Aware Classification | IEEE Conference Publication | IEEE Xplore*. URL: https://ieeexplore.ieee.org/document/6413831 (visited on 05/22/2024) (cit. on p. 8).

[39] *COMPAS Recidivism Risk Score Data and Analysis*. URL: https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis (visited on 05/22/2024) (cit. on p. 8).

[40] *DOC COMPAS*. URL: https://doc.wi.gov/Pages/AboutDOC/COMPAS.aspx (visited on 05/22/2024) (cit. on p. 8).

[41] Ronny Kohavi Barry Becker. *Adult*. 1996. DOI: 10.24432/C5XW20. URL: https://archive.ics.uci.edu/dataset/2 (visited on 05/22/2024) (cit. on p. 8).

[42] Hans Hofmann. *Statlog (German Credit Data)*. 1994. DOI: 10.24432/C5NC77. URL: https://archive.ics.uci.edu/dataset/144 (visited on 05/22/2024) (cit. on p. 8).

[43] Paula Gordaliza et al. "Obtaining Fairness using Optimal Transport Theory". en. In: *Proceedings of the 36th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, May 2019, pp. 2357–2365. URL: https://proceedings.mlr.press/v97/gordaliza19a.html (visited on 05/22/2024) (cit. on p. 11).

[44] Victor M. Panaretos and Yoav Zemel. “Statistical Aspects of Wasserstein Distances”. en. In: *Annual Review of Statistics and Its Application* 6.1 (Mar. 2019). arXiv:1806.05500 [stat], pp. 405–431. ISSN: 2326-8298, 2326-831X. DOI: 10.1146/annurev-statistics-030718-104938. URL: http://arxiv.org/abs/1806.05500 (visited on 05/22/2024) (cit. on p. 11).

[45] *pandas - Python Data Analysis Library*. URL: https://pandas.pydata.org/ (visited on 05/22/2024) (cit. on p. 13).

[46] *PyPI ů The Python Package Index*. en. URL: https://pypi.org/ (visited on 05/22/2024) (cit. on p. 13).

[47] *Contributing to AIF360 aif360 0.6.1 documentation*. URL: https://aif360.readthedocs.io/en/latest/CONTRIBUTING.html (visited on 05/22/2024) (cit. on p. 13).

[48] *Welcome Sphinx documentation*. URL: https://www.sphinx-doc.org/en/master/ (visited on 05/22/2024) (cit. on p. 13).

[49] *Project Jupyter*. en. URL: https://jupyter.org (visited on 05/22/2024) (cit. on p. 14).

[50] *PEP 8 Style Guide for Python Code | peps.python.org*. en. URL: https://peps.python.org/pep-0008/ (visited on 05/22/2024) (cit. on p. 14).

[51] “Inspiron 16 7610 Setup and Specifications”. en. In: () (cit. on p. 14).

[52] *SDPA Official Page*. URL: https://sdpa.sourceforge.net/ (visited on 05/22/2024) (cit. on p. 18).

[53] *Welcome to CVXPY 1.5 CVXPY 1.5 documentation*. URL: https://www.cvxpy.org/ (visited on 05/22/2024) (cit. on p. 18).

[54] *Quan-Zhou/Fairness-in-Learning-of-LDS*. URL: https://github.com/Quan-Zhou/Fairness-in-Learning-of-LDS (visited on 05/22/2024) (cit. on p. 19).

[55] *Algorithms aif360 0.6.1 documentation*. URL: https://aif360.readthedocs.io/en/stable/modules/algorithms.html#module-aif360.algorithms.preprocessing (visited on 05/22/2024) (cit. on p. 23).

[56] *Distributions of the KullbackLeibler divergence with applications - Belov - 2011 - British Journal of Mathematical and Statistical Psychology - Wiley Online Library*. URL: https://bpspsychub.onlinelibrary.wiley.com/doi/abs/10.1348/000711010X522227?casa_token=gHwChikjVs8AAAAA%3A_YKB4tHp4jNDe2Idg7fAC87OGyVGtTyAKiDSbmPWGJDyDNXnb6gSkmJWWh-uavADVk1gNwrb5ZtwJn-5 (visited on 05/22/2024) (cit. on p. 26).

[57] Nicolas Loizou and Peter Richtárik. “Momentum and stochastic momentum for stochastic gradient, Newton, proximal point and subspace descent methods”. en. In: *Computational Optimization and Applications* 77.3 (Dec. 2020), pp. 653–710. ISSN: 1573-2894. DOI: 10.1007/s10589-020-00220-z. URL: https://doi.org/10.1007/s10589-020-00220-z (visited on 06/04/2024) (cit. on p. 31).

Appendix A

Github Repository Links

Project fork of the AIF360 Repository

- Subgroup and Instantaneous Algorithms
- Subgroup and Instantaneous Demonstration File
- Distributional Repair Algorithm
- Distributional Repair Demonstration File

Appendix B

Additional Algorithms

B.1 Subgroup and Instantaneous Fairness

Algorithm 7 CalculateBaseRate

procedure CALCULATEBASERATE($\mathcal{D}, S, \hat{Y}$)
$\mathcal{D}^{privileged}, \mathcal{D}^{unprivileged} \leftarrow$ SPLITDATA($\mathcal{D}, S$)
$privilegedCount \leftarrow |\mathcal{D}^{privileged}|$
$privilegedPosCount \leftarrow |d \in \mathcal{D}^{privileged} : d_{\hat{Y}} = 1|$
$unprivilegedCount \leftarrow |\mathcal{D}^{unprivileged}|$
$unprivilegedPosCount \leftarrow |d \in \mathcal{D}^{unprivileged} : d_{\hat{Y}} = 1|$
$baseRatePrivileged \leftarrow 1 - \frac{privilegedPosCount}{privilegedCount}$
$baseRateUnprivileged \leftarrow 1 - \frac{unprivilegedPosCount}{unprivilegedCount}$
return $baseRatePrivileged, baseRateUnprivileged$
end procedure

Algorithm 8 SplitData

procedure SPLITDATA($\mathcal{D}, S$)
$\mathcal{D}^{privileged} \leftarrow d \in \mathcal{D} : d_S = 1$
$\mathcal{D}^{unprivileged} \leftarrow d \in \mathcal{D} : d_S = 0$
return $\mathcal{D}^{privileged}, \mathcal{D}^{unprivileged}$
end procedure

Algorithm 9 CreateOptimization

procedure CREATEOPTIMIZATION($\mathcal{D}^{privileged}, \mathcal{D}^{unprivileged}, X, \hat{Y}$)
$xVars, e, z \leftarrow$ CREATEDECISIONVARIABLES($|X|$)
$constraints, objD \leftarrow$ GETCONSTRAINTS($\mathcal{D}^{privileged}, \mathcal{D}^{unprivileged}, X, \hat{Y}, xVars, e, z$)
return $constraints, objD, xVars, e, z$
end procedure

Algorithm 10 SolveOptimization

procedure SOLVEOPTIMIZATION($constraints, objD, xVars, e, z$)
$xVars_{solved}, e_{solved}, z_{solved} \leftarrow$ SOLVE($constraints, objD, xVars, e, z$)
return $xVars_{solved}, e_{solved}, z_{solved}$
end procedure

Algorithm 11 ReweighData

procedure REWEIGHDATA($\mathcal{D}, X, \hat{Y}, xVars_{solved}, e_{solved}$)
  $\mathcal{D}$**reweighted** $\leftarrow \mathcal{D}$
  for $d \in \mathcal{D}$**reweighted**
    if $d_S = 0$ **then**
      $score \leftarrow \sum_{i=0}^{|X|-1} xVars_{solved}[2i] \cdot d_{X_i} + e_{solved}[0]$
    else
      $score \leftarrow \sum_{i=0}^{|X|-1} xVars_{solved}[2i+1] \cdot d_{X_i} + e_{solved}[1]$
    end if
    $d_{\hat{Y}} \leftarrow score$
  end for
  return $\mathcal{D}_{\textbf{reweighted}}$
end procedure

Algorithm 12 Normalize

procedure NORMALIZE($\mathcal{D}, \hat{Y}$)
  $\mathcal{D}$**normalized** $\leftarrow \mathcal{D}$
  $\hat{Y}$**min** $\leftarrow \min_{d \in \mathcal{D}} d_{\hat{Y}}$
  $\hat{Y}$**max** $\leftarrow \max d \in \mathcal{D} d_{\hat{Y}}$
  for $d \in \mathcal{D}$**normalized**
    $d\hat{Y} \leftarrow$ **round** $\left(\frac{d_{\hat{Y}} - \hat{Y}\textbf{min}}{\hat{Y}\textbf{max} - \hat{Y}\textbf{min}}, 1 \right)$
    $d\hat{Y} \leftarrow \min(\max(d_{\hat{Y}}, 0), 1)$
  end for
  return $\mathcal{D}_{\textbf{normalized}}$
end procedure

Algorithm 13 ApplyThreshold

procedure APPLYTHRESHOLD($\mathcal{D}, baseRates, S, \hat{Y}$)
  $\mathcal{D}$**classified** $\leftarrow \mathcal{D}$
  $thPrivileged \leftarrow$ PERCENTILE$(d \in \mathcal{D} : d_S = 1, \hat{Y}, baseRates[0] \cdot 100)$
  $thUnprivileged \leftarrow$ PERCENTILE$(d \in \mathcal{D} : d_S = 0, \hat{Y}, baseRates[1] \cdot 100)$
  for $d \in \mathcal{D}$**classified**
    if $d_S = 1$ **then**
      $d_{\hat{Y}} \leftarrow 1[d_{\hat{Y}} \geq thPrivileged]$
    else
      $d_{\hat{Y}} \leftarrow 1[d_{\hat{Y}} \geq thUnprivileged]$
    end if
  end for
  return $\mathcal{D}_{\textbf{classified}}$
end procedure

B.2 Distributional Repair

Algorithm 14 GetSupport

procedure GETSUPPORT($feat, u_{val}, n_q$)
$min_{val} \leftarrow \min$(values of $feat$ where $u = u_{val}$) $- 0.1 \cdot$ range(values of $feat$ where $u = u_{val}$)
$max_{val} \leftarrow \max$(values of $feat$ where $u = u_{val}$) $+ 0.1 \cdot$ range(values of $feat$ where $u = u_{val}$)
return evenly spaced sequence of length n_q from min_{val} to max_{val}
end procedure

Algorithm 15 GetContinuousProbabilityDensityFunctions

procedure GETCONTINUOUSPROBABILITYDENSITYFUNCTIONS($feat, u_{val}$)
$kde_0 \leftarrow$ Kernel density estimation fit to values of $feat$ where $u = u_{val}$ and $s = 0$
$pdf_0 \leftarrow \exp$(KDE scores of $support$ using kde_0)
$kde_1 \leftarrow$ Kernel density estimation fit to values of $feat$ where $u = u_{val}$ and $s = 1$
$pdf_1 \leftarrow \exp$(KDE scores of $support$ using kde_1)
normalize pdf_0 and pdf_1 ▷ Raise error if sum is 0
return pdf_0, pdf_1
end procedure

Algorithm 16 GetBarycenter

procedure GETBARYCENTER($pdf_0, pdf_1, feat, u_{val}$)
$M \leftarrow$ pairwise distances between points in $support$
$A \leftarrow$ stack pdf_0 and pdf_1 as columns
$barycenter \leftarrow$ Bregman barycenter of A using M
return $barycenter$ ▷ Raise error if invalid
end procedure

Algorithm 17 GetContinuousTransportPlans

procedure GETCONTINUOUSTRANSPORTPLANS($pdf_0, pdf_1, barycenter, feat, u_{val}$)
$M \leftarrow$ pairwise distances between points in $support$
$T_0 \leftarrow$ Earth mover's distance from pdf_0 to $barycenter$ using M
$T_1 \leftarrow$ Earth mover's distance from pdf_1 to $barycenter$ using M
return T_0, T_1
end procedure

Algorithm 18 GetDiscreteProbabiltyMassFunctions

procedure GETDISCRETEPROBABILTYMASSFUNCTIONS($feat, u_{val}$)
$pmf_0 \leftarrow$ value counts of $feat$ where $u = u_{val}$ and $s = 0$
$pmf_1 \leftarrow$ value counts of $feat$ where $u = u_{val}$ and $s = 1$
return pmf_0, pmf_1
end procedure

Algorithm 19 GetDiscreteTransportPlan

procedure GETDISCRETETRANSPORTPLAN(pmf_0, pmf_1)
$M \leftarrow$ pairwise distances between unique values of pmf_0 and pmf_1
$weights_0 \leftarrow pmf_0$ values normalized by sum
$weights_1 \leftarrow pmf_1$ values normalized by sum
$T \leftarrow$ Earth mover's distance from $weights_0$ to $weights_1$ using M
return T
end procedure

Appendix C

Word Count

```
Total
Words in text: 7160
Words in headers: 215
Words outside text (captions, etc.): 264
Number of headers: 60
Number of floats/tables/figures: 20
Number of math inlines: 294
Number of math displayed: 20
```

The word count above does not include the following:

- Title Page
- Acknowledgements
- Contents
- Bibliography
- Appendix