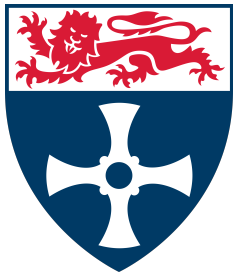
Fairness-Aware Data Preprocessing for Classification Tasks



Carlos Vladimiro González Zelaya

Supervisors: Prof. Paolo Missier
Dr. Dennis Prangle

School of Computing
Newcastle University

This thesis is submitted for the degree of
Doctor of Philosophy

July 2022

I dedicate this thesis to the memory of my dad, as well as to my dearest mom and aunts:
your love and unconditional support were an essential part of my PhD.

Dedico esta tesis a la memoria de mi papá, así como a mis queridas mamá y tías:
su amor y apoyo incondicional fueron parte esencial de mi doctorado.

Declaration

I declare that this thesis is my own work unless otherwise stated. No part of this thesis has previously been submitted for any other degree or qualification at Newcastle University or any other institution.

Carlos Vladimiro González Zelaya
July 2022

Publications

Significant portions of the work presented within this thesis have been documented in the following papers:

Conference

- Salas, J. and González-Zelaya, V. (2020). Fair-MDAV: An algorithm for fair privacy by microaggregation. In *Modeling Decisions for Artificial Intelligence. MDAI 2020*, volume 12256. Springer

- González-Zelaya, V., Salas, J., Prangle, D., and Missier, P. (2021b). Optimising Fairness Through Parametrised Data Sampling. In *Proceedings of the 2021 EDBT Conference*

Workshop

- González-Zelaya, V., Missier, P., and Prangle, D. (2019). Parametrised Data Sampling for Fairness Optimisation. Presented on the *2019 XAI Workshop at SIGKDD, Anchorage, AK, USA*

Submitted

- González-Zelaya, V., Salas, J., Megías, D., and Missier, P. (2021a). Fair Classification with Privacy Guarantees. Submitted to the *Data Mining and Knowledge Discovery (DAMI)* Journal

Acknowledgements

I wish to extend my appreciation to my supervisors Prof. Paolo Missier and Dr. Dennis Prangle for their help, comments and suggestions on the present work. Thanks to them, this thesis has become more understandable, concise, deeper, and generally a better work.

I would also like to thank my examiners Dr. Jaume Bacardit and Prof. H. V. Jagadish for their comments and corrections, as they greatly improved the quality of the final version of this thesis.

My gratitude towards Universidad Panamericana, that generously sponsored my studies in the UK. I would like to especially thank Dr. Héctor Xavier Ramírez Pérez, Dr. José Alberto Ross Hernández, Dr. Marisol Velázquez Salazar, Alma Rosa Limas Álvarez, Diana Alejandra Hernández García and Diana Aparicio González, who helped me a great deal whenever a situation arose, both personally and professionally.

My sincere thanks to Prof. Paul Watson, Jennifer Wood and Andrew Turnbull for their invaluable support and their continuous effort in improving our Centre in all regards.

I would like to thank everyone at the CDT in Cloud Computing for Big Data, for their friendship and good vibes made this process an incredibly fun and enriching life experience.

Most importantly, my love and gratitude to Pil: sharing this adventure with you made it better in every possible way.

Abstract

The prevalence of decision-making mechanisms in life-impacting decisions, ranging from bank loans and college admissions to probation decisions, makes understanding and controlling the *fairness* of algorithmically-generated decisions indispensable. This thesis presents an introduction to algorithmic fairness, focusing on classification tasks. A survey of state-of-the-art fairness-correcting methods is presented, emphasising data preprocessing solutions.

The thesis' research aim is *to design, implement and evaluate data preprocessing methods that correct unfair predictions in classification tasks*. Three such methods are presented, sharing the trait of being fairness-definition and classifier agnostic. For each of these methods, experiments are performed on widely used benchmark datasets.

FAIRPIPES is a genetic-algorithm method which optimises for user-defined combinations of multiple definitions of fairness and accuracy, providing flexibility in the fairness-accuracy trade-off. FAIRPIPES heuristically searches through a large space of pipeline configurations, achieving near-optimality efficiently and presenting the user with an estimate of the best attainable fairness/accuracy trade-offs. The optimal pipelines are shown to differ for different datasets, suggesting that no "universal best" pipeline exists and confirming that FAIRPIPES fills a niche in the *fairness-aware AutoML* space.

PARDS is a parametrised data sampling method by which it is possible to optimise the *fairness ratios* observed on a dataset, in a way that is agnostic to both the specific fairness definitions, and the chosen classification model. Given a dataset with one binary protected attribute and a binary label, PARDS' approach involves correcting the positive rate for both the *favoured* and *unfavoured* groups through resampling of the training set. PARDS is shown to produce fairness-optimal predictions with a small loss in predictive power.

FAIR-MDAV is a fairness correcting preprocessing method with privacy guarantees. It outperforms existing fairness correcting methods on its equalised odds/accuracy trade-off, and is competitive on its demographic parity/accuracy trade-off as well. FAIR-MDAV is modular, allowing for privacy guarantees to be set separately from fairness correction.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AB	AdaBoost 12
ACC	Accuracy 13
AHD	Averaged Hausdorff Distance 61
B2B	Best To Best Distance 62
DP	Demographic Parity 23
DPD	Demographic Parity Difference 38
DPR	Demographic Parity Ratio 69
DT	Decision Tree 11
EO	Equalised Odds 23
EOD	Equalised Odds Difference 38
EOp	Equality of Opportunity 23
EOpD	Equality of Opportunity Difference 38
EOpR	Equality of Opportunity Ratio 69
FN	False Negatives 13
FP	False Positives 13
FTU	Fairness Through Unawareness 25
GDPR	General Data Protection Regulation 4
GNB	Gaussian Naive Bayes 90
LOO	Leave One Out 12
LR	Logistic Regression 11
ML	Machine Learning 4
PA	Protected Attribute 8
PF	Proxy Fairness 23
PFR	Proxy Fairness Ratio 69
PR	Positive Ratio 22
SVM	Support Vector Machine 11
TN	True Negatives 13
TP	True Positives 12
WoE	Weight of Evidence 18

List of Symbols

F	The favoured PA group
U	The unfavoured PA group
$\hat{Y}$	A classifier function
$\mathbb{E}$	The expected value of a random variable
$\mathbb{P}$	The probability of an event
$\mathbb{R}$	The set of real numbers
Dom	The domain of a function

Chapter 1

Introduction

I have a dream
that my four little children
will one day live in a nation
where they will not be judged
by the color of their skin
but by the content of their character.
I have a dream today.

Martin Luther King Jr.

Contents

Summary

This chapter begins by outlining the need to research *algorithmic fairness*—this thesis' main topic—in Section 1.1 by presenting the archetypical case of COMPAS, a classifier used in the United States to predict whether convicted defendants are likely to re-offend after their release. Section 1.2 continues by discussing the meaning of fairness, first from a human perspective followed by the necessity to formalise its definition in order to quantify and improve it. In Section 1.3 the relevance of certifying and intervening automated decision-making systems to ensure the fairness of their decisions is discussed. Finally, Section 1.4 presents both our general and specific research goals, as well as the contributions presented in this thesis.

1.1 The COMPAS Controversy

COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is a decision-support tool developed by Northpointe (now Equivant) to assess the *recidivating* (or re-offending) risk of convicted defendants in several jurisdictions of the United States in the two years after their release. COMPAS' predictions are based on 137 features obtained from a defendant's personal information and their criminal record [48].

COMPAS has faced significant criticism from academia following ProPublica's 2016 article *Machine Bias* [4], where a dataset consisting of two years of COMPAS Scores from Broward County in Florida, USA, consisting of 18,610 records between 2014 and 2015. In their analysis, Propublica found that, even though the error rates in recidivism-prediction were similar for white and black defendants, the "direction" in which these errors happened were not consistent for both groups, and hence COMPAS predictions were discriminatory against black defendants. Specifically, they found that "black defendants were often predicted to be at a higher risk of recidivism than they actually were,... (while) white defendants were often predicted to be less risky than they were" [4].

In response to this, Northpointe presented their own analysis [41], where they pointed out that COMPAS did not discriminate since "black defendants who were predicted to recidivate ... actually did recidivate at a higher rate than the white defendants ... (while) white defendants who were predicted not to recidivate actually did not recidivate at a higher rate than the black defendants" [41, p. 20–21].

Surprisingly, both claims are true on the analysed dataset. The point of contention is not the technical correctness of the analyses, but the fairness definition that was used: while ProPublica refers to *equalised odds*, Northpointe uses a different definition, known

as *predictive parity*. As discussed in Chapter 3, deciding which fairness definition to use on a decision task is subjective, and will ultimately depend on the user's *world-views* [55].

As an aftermath of the COMPAS controversy, in July 2016 the Supreme Court of Wisconsin, USA determined to continue allowing judges to consider risk scores during sentencing, but instructed the judges to warn about their "limitations and cautions" [95, p. 22]. Aside from this decision, further arguments against the use of recidivism prediction have been made by Dressel and Farid [43], who experimentally show that COMPAS is neither more accurate nor less biased than inexpert-human decisions; furthermore, they show that COMPAS' accuracy may be achieved by a linear predictor dependant on just two features.

1.2 The Meaning of Fairness

The COMPAS controversy is just an instance of a much larger problem, namely the lack of a unique definition of fairness. While outside the scope of this thesis, it is worth mentioning that this debate has existed in the social sciences for a long time, with notions of fairness coming from as far back as Aristotle's *Nicomachean Ethics*, where he defines fairness as "involving equitable distributions and the correction of what is inequitable" [32, Book V, Section 10]. While this definition lacks formality, in the sense that it is impossible to quantify whether a given decision was fair or not based on it, its spirit is closely related to the general aim of this thesis, presented in Section 1.4.

In the computer science community, in contrast, many formal definitions of fairness have been proposed [101], but with no definitive agreement [55]. A decision rule that satisfies one of the definitions may well prove to be very unfair for a different one [29]. For example, determining university admissions through gender quotas may achieve *demographic parity*, but it makes the acceptance rates for good students of different genders disparate.

A first divide in algorithmic fairness definitions appears in deciding whether fairness is an individual or a collective issue: a family of definitions, known as *individual* fairness, advocates for the principle of similar individuals being treated similarly. *Group* fairness, on the other hand, advocates for a similar predictive behaviour across population subgroups, even if this is achieved at the expense of individuals who might otherwise obtain a better prediction. This division, in turn, results in two types of fairness correction solutions: in the case of individual fairness, the first problem comes in defining an adequate distance between individuals that determines whether two individuals should be considered similar

or not. This problem is not shared by group fairness, where the interest is solely in predictive outcomes; this thesis focuses on group fairness definitions.

Within group fairness, a second divide comes from different notions of equality. Specifically, *equality of outcomes* strives for similar prediction distributions across population subgroups, as there are historical and societal biases that need to be corrected through *affirmative action*. On the other hand, *equality of opportunity* seeks to guarantee that deserving individuals be correctly evaluated in the same proportion across population subgroups. In general, it is not possible to determine whether one of these definitions is necessarily better or fairer than another. Instead, each of these "encode some belief about the world" [55]. Therefore, the works presented in this thesis are *agnostic* to most group fairness definitions: they allow the user to decide which definition encodes their world-view, or the spirit of the decision that is to be made.

1.3 The Importance of Having Fair Algorithms

The prevalence of decision-making mechanisms in life-impacting decisions, ranging from bank loans and college admissions to probation decisions, makes understanding and controlling the *fairness* of algorithmically-generated decisions indispensable, pushing fairness to the forefront of machine learning (ML) research.

Among the multiple potential consequences of deploying an unfair algorithm, perpetuating discriminatory decisions is one of the most problematic ones. Besides the COMPAS case discussed in Section 1.1, a showcase of representative cases where automated decision-making mechanisms have been shown to be discriminatory is presented next.

- The European Union's *General Data Protection Regulation (GDPR)* establishes the principles applicable to *any data processing activity* [160, Page 87]; this broad definition includes automated decision-making mechanisms. Article 5(1) of the GDPR states that "*personal data shall be processed lawfully, **fairly** and in a transparent manner in relation to the data subject*" [160, Page 88].

- In 2018 a Reuters article [37] showed that an Amazon automated recruiting tool discriminated against submitted resumes including the word *women's*, as in "member of the women's swimming team". The reason for this behaviour was that the tool was trained on a dataset of 10 years of previous applications, for which most of the successful candidates were male, thus causing the model to associate the word *women's* with unsuccessful candidates. Amazon eventually disbanded the team

responsible for the tool, and claimed that "it was never used by their recruiters to evaluate candidates". However, this example shows how an existing societal bias, namely the lack of women in the tech industry, can carry on as a consequence of an unfair mechanism.

- Apple's banking service, Apple Card, has been shown to assign disparate credit lines for males and females, notoriously with Apple co-founder Steve Wozniak and his wife, who share bank accounts and have credit cards with the same high limit, yet he was granted a credit line ten times larger than his wife's [70].

- Twitter's image-cropping algorithm, which is used to show tweets on users feeds, keeps white/female faces more often than black/male ones in the cropped image [86], an issue they have acknowledged and are taking measures to correct [30].

- Google Translate, one of the most popular translation engine on the Web, has a tendency to assign specific genders to professions when translating [92], and although they have partially corrected this problem [99], for many instances it persists, especially in the female case.

- Predictive policing trained on historical data has been shown to cause feedback loops on which zones to patrol [71], i.e. frequently patrolling a zone makes a detention likelier there, which in turn causes the zone to be frequently patrolled.

Regardless of the specific metric adopted to measure fairness, the principal—yet not the only [78]—factor leading to unfair automated decisions is biased training data [36, 95], either due to historical disparities and discrimination [111] or to improper data collection [103]. In this regard, the computing community is ideally positioned to help fix this problem [147, 146]. The following section specifies this thesis' research aim and particular objectives.

1.4 Research Aim and Objectives

This thesis' general research aim is *to design, implement and evaluate data preprocessing methods that correct unfair predictions in classification tasks*. In an attempt to fulfill this general aim, the following specific objectives were selected:

1. To understand and formalise the concept of *algorithmic fairness*.
2. To determine whether the choice and order of the non-fairness-specific preprocessing operators have an impact on the fairness of the predictions made by a classifier learnt from the resulting data.
3. To estimate the trade-off between fairness and performance for classifiers, generating a range of possible trade-offs from which to choose.
4. To design a fairness-definition and classifier agnostic preprocessing method to optimise the specified fairness definition for a classifier without incurring a big accuracy loss.
5. To analyse the connection between fairness and privacy in the classification context. In particular, to design a method capable of correcting fairness while providing privacy guarantees to the resulting dataset.

1.5 Thesis Outline and Contributions

This section presents an outline for the rest of the thesis and presents each chapter's contributions towards addressing Section 1.4's specific objectives.

Chapter 2 presents definitions on classifiers, performance metrics and data preprocessing, which are required in order to follow the rest of the thesis, followed by an introduction to algorithmic fairness and the fairness definitions used throughout the thesis. This chapter partly addresses **research objective 1**.

Chapter 3 presents the work related to Chapters 4, 5 and 6, including an explanation of the fairness/accuracy trade-off phenomenon and a survey on the state-of-the-art in fairness correction for classification tasks, with a focus on preprocessing methods. This chapter completes the addressing of **research objective 1**.

Chapter 4 analyses the effect of several data preprocessing operators over different fairness definitions, addressing **research objective 2**. This is followed by the introduction of FAIRPIPES, a genetic algorithm with mutation and cross-over operations that optimises preprocessing pipelines through preprocessor-and-order selection from sets of operator options. FAIRPIPES presents the user with an estimate of the fairness/performance Pareto front for the selection of preprocessing operators, addressing **research objective 3**.

Chapter 5 presents PARDS, a fairness-and-classifier agnostic fairness-correcting preprocessing solution based on data resampling. In the simpler setting, instances in the training set are divided into *favoured* (F) and *unfavoured* (U) with respect to a binary attribute for which discrimination is to be prevented, e.g. the gender. The groups are further divided by their class label as *positive* ($+$) and *negative* ($-$). Each of the four resulting groups—F^+, F^-, U^+ and U^-—are then independently resampled to balance F and U with respect to a particular fairness definition. After describing PARDS, theoretical results on its effectiveness for the univariate case are presented, and the method is benchmarked and compared with state-of-the-art solutions, producing better fairness/accuracy trade-offs than most of them. This chapter addresses **research objective 4**.

Chapter 6 borrows the *fairlet* concept from *fair clustering* and introduces FAIR-MDAV, a fairness correction method that works through micro-aggregation, i.e. aggregating most attributes of each n nearby-instances cluster through an aggregation function, e.g. mean or mode, making n copies of this aggregate while keeping the original class label and assigning the *protected attribute* values to each copy in accordance to its distribution over the whole training set; FAIR-MDAV provides the user with privacy guarantees, specifically k-anonymity and t-closeness. This addresses **research objective 5**.

Chapter 7 presents a summary of the thesis' contributions, final remarks and future research directions.

Research Limitations

In Chapters 5 and 6, experiments were performed on three standard fairness-benchmark datasets: *Adult Income*, *COMPAS* and *German Credit*; experiments were also performed over the *Titanic* dataset in Chapter 4. All these datasets, described in Subsection 2.6.3, were chosen due to their *binary* label, as this is a requirement of the employed fairness

definitions and therefore of the presented fairness-correcting methods as well. Depending on the problem at hand, the binary label limitation may be dealt with by grouping the possible label values into two categories, usually referred to as the *positive* and the *negative* outcomes.

A second limitation of the analysed fairness definitions and correcting methods is the need for a single binary *protected attribute (PA)*, i.e. the feature for which fairness is measured and corrected. This restriction can be relaxed in both the number of PAs and the number of possible values of each PA if necessary: an example of how to do this is presented in Chapter 5.

1.5.1 Key Insights

The following are this thesis' key insights and the chapters supporting them.

- The way in which data is preprocessed directly affects a classifier's performance and fairness; see Chapter 4.
- Fairness and performance metrics are in conflict with each other; see Chapters 3, 4, 5 and 6.
- It is possible to correct classification fairness through data preprocessing; see Chapters 5 and 6.
- Different fairness definitions require specific amounts of correction. Enforcing a particular fairness definition does not cause—and generally prevents—a different definition from being satisfied; see Chapters 4, 5 and 6.
- Certain definitions of fairness and privacy are compatible with each other; see Chapter 6.

Chapter 2

Preliminaries

Contents

Summary

This chapter describes standard background material which will be used throughout the thesis, namely classifiers, performance metrics and data preprocessing. Section 2.2 describes basic notions of classifiers, as well as the standard notation this work adheres to. Since most of this work is focused on binary classification problems, Section 2.3 presents five standard binary classification performance metric definitions. Section 2.5 presents five of the most common data preprocessing steps, with a showcase of options to perform each of these steps. Finally, Section 2.6 presents the fairness definitions that are used throughout the rest of the thesis.

2.1 Supervised Machine Learning

According to Goodfellow et al. [66], ML refers to the capability of artificial intelligence systems "to acquire their own knowledge, by extracting patterns from raw data"; ML methods may be divided into two categories: *unsupervised* and *supervised*.

Unsupervised learning refers to methods that look for relationships in data without having a measured outcome, with the usual goal of finding clusters, components or trajectories in the data structure [85]. Supervised methods, on the other hand, aim at *predicting* a target outcome on instances of a data structure for which such target is unknown. The nature of the target may be *continuous*, e.g. a real number, or *discrete*, e.g. a category from a finite set of options. Supervised methods aimed at predicting a continuous value are known as *regression* methods, while supervised methods aimed at predicting a category are known as *classification* methods. This thesis is about the latter, and therefore an introduction to ML classification methods now follows.

2.2 Classification

Classification is one of the most common supervised learning tasks, along with regression. In classification, the purpose is to create a model that predicts a data instance's *class*, i.e. a particular category among a finite set of options. Some examples of class variables are a person's sex, a medical diagnosis among a set of possible diseases or whether a bank loan will be paid or not.

Formally, a dataset D is a rectangular array

$$D = \begin{array}{c} \begin{matrix} X_1 & X_2 & \cdots & X_p & Y \end{matrix} \\ \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} & y_1 \\ x_{21} & x_{22} & \cdots & x_{2p} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} & y_n \end{pmatrix} \end{array}$$

where x_{ij} represents the value of the j-th variable for the i-th observation, with $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, p$. D's variables are divided in $X = (X_1, X_2, \ldots, X_p)$, the *feature* variables, which may be numerical (or *quantitative*) or categorical (or *qualitative*), and Y, the *class* or *target* label, the categorical variable which is to be predicted. Each row in D will be an *instance* or *record* [83].

In this context, a classifier may be defined as a function from the domain of the feature variables to the domain of the class $\hat{Y} : \mathrm{Dom}(X) \to \mathrm{Dom}(Y)$, i.e. $\hat{Y}$ assigns a label to every possible data instance. When $\mathrm{Dom}(Y)$ consists of only two labels it will be referred to as *binary* classification; most of the classification tasks dealt with in this thesis are binary.

Since the goal of a classification task is usually to generate a classifier which produces a set of predictions $\hat{Y}(X)$ that is as similar to Y as possible, a set of performance metrics which will allow us to quantify the predictive performance of a classifier is defined in Subsection 2.3.

There are several families of classifiers, which vary in both the way they operate as well as on the way the models are fitted to the data. Common classifiers range from logistic regression (LR) [79], a binary classifier which is based on fitting the coefficients of a logistic function, to decision trees (DTs) [130], which work through a set of nested if-else statements, to margin-maximising methods such as support vector machines (SVMs) [67], which aim at maximising the distance between instances and a separating hyperplane (the *margin*) on a high-dimensional embedding of the data.

Although some of these models only work for binary classification in their simplest form, e.g. LR and SVM, they can handle multi-class problems by considering several binary classification problems, and then applying a combining rule on the set of decisions, e.g. voting. Two approaches are commonly applied to do this:

1. Classifiers between each class and the union of *all* the other classes are trained. This is called the one-versus-rest approach [150].

2. Classifiers between every pair of classes are trained. This is known as a *one-versus-one* or *all-pairs* approach [83].

It is also possible to combine multiple classifiers into an ensemble by *bagging*, i.e. aggregating the predictions of classifiers learnt from different training-set samples through majority voting, or by *boosting*, i.e. fitting a sequence of classifiers on repeatedly modified versions of the data and then combining them through a weighted vote. The aim of these ensemble methods is to obtain a more powerful prediction model, with a reduced variance of the resulting classifier's predictions [83]. Typical ensemble-method classifiers are random forests [16] for bagging and AdaBoost (AB) [140] for boosting.

2.3 Performance Metrics

In order to quantify the quality of a classifier, several possible performance metrics may be considered. It is a standard practice to separate the data into a *training* set, which will be used to fit the model, and a *test* set, which is used to measure these metrics. Furthermore, if the computational budget allows for it, performing k-fold cross-validation, i.e. splitting the dataset into k evenly-sized subsets, or *folds*, treating each of them as test and the rest as training to evaluate the performance metrics and finally averaging out the resulting metrics, will result in more accurate performance estimates than a single split [119].

The selection of k ranges from 2 to the number of instances in the data (this particular case is known as *leave one out (LOO)*), and it determines the relative sizes of the training and test sets. Even though this value is arbitrary, the literature [83] suggests the values of $k = 5$ or $k = 10$ (resulting in 80/20 or 90/10 train-test splits, respectively), which are considerably less computationally-expensive, yet produce a smaller variance in test error rates than LOO. The cross-validation folds may be sampled in different ways: this may happen either *consecutively*, i.e. in the order of appearance of the instances in the dataset, *shuffling* the instance-order before splitting the data, or through *stratified* random sampling; this last case means that the sampling is done so that the class proportions for each fold are roughly equal to the class proportions in the whole training set [13].

In a binary classification problem, i.e. one where there are only two possible outcomes (negative and positive, codified as 0 and 1 respectively), and given a set of predictions $\hat{Y}(T)$ over test set T, it is possible to aggregate the instances of T into one of the following four categories, and represent them in a *confusion matrix*, as shown in Table 2.1:

True positives (TP) Positive instances that are correctly classified as positive.

Table 2.1 The confusion matrix of a binary classifier's predictions.

	True Class	
Predicted Class	Positive	Negative
Positive	TP	FP
Negative	FN	TN

True negatives (TN) Negative instances that are correctly classified as negative.

False positives (FP) Negative instances that are incorrectly classified as positive.

False negatives (FN) Positive instances that are incorrectly classified as negative.

Based on these four categories, the most common performance metrics in binary classification are now defined, along with the advantages and disadvantages inherent to each of them. All of these metrics range between 0 and 1, with higher values being better.

Accuracy (ACC) The ratio of correctly classified instances in the test set:

$$ACC = \frac{|TP| + |TN|}{|T|}.$$

Precision The ratio of correctly classified positive instances over the total number of instances classified as positive, i.e. how good a classifier is at *avoiding* false positives:

$$\text{precision} = \frac{|TP|}{|TP| + |FP|}.$$

Recall The ratio of correctly classified positive instances over the total number of positive instances, i.e. how good a classifier is at *detecting* positive instances:

$$\text{recall} = \frac{|TP|}{|TP| + |FN|}.$$

F_1 **Score** The harmonic mean of precision and recall:

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

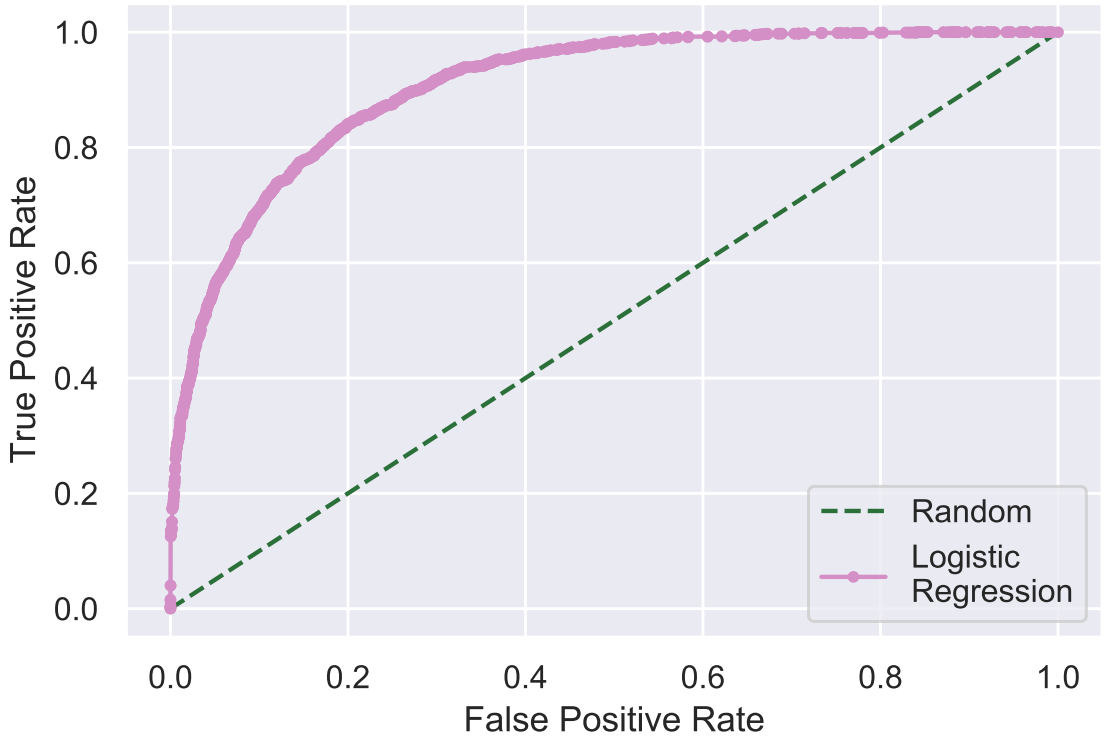


Figure 2.1 The ROC curve for a LR classifier. The ROC AUC Score is the area below the pink curve. The green line represents the ROC curve for random classifiers that assign a positive label with probability $p \in [0, 1]$.

ROC AUC Score Area under the Receiver Operating Characteristic Curve (ROC AUC), which plots $|FP|/|T|$ on the x-axis and $|TP|/|T|$ on the y-axis for different discrimination thresholds, i.e. the value at which a classifier will change an instance's class. For example, in LR predictions are determined by evaluating a fitted logistic function's values on an instance and comparing this functional value with a preset threshold, e.g. 0.5 to determine the instance's class, and changing this threshold value will result in different TP and FP values for the resulting classifier. A plot of the ROC curve of LR classifiers with different classification thresholds is shown in Figure 2.1.

It is possible to generalise these metrics to the multi-class case by first considering each class i as "positive" and the combined rest of the classes as "negative", evaluating TP_i, FP_i, TN_i and FN_i and the corresponding metrics for each class, and finally averaging out the resulting metrics.

2.4 Genetic Algorithms

This section is based on the genetic algorithm tutorial by Whitley [162]. Genetic algorithms as a family of computational models, inspired by evolution, that may be used to search through a potentially large search-space for a function's optimal value. The original genetic algorithm model was introduced in Holland [77] in 1975. As genetic algorithms do not use gradient information, they may be used to optimise many different classes of problems, ranging from robot path planning, image processing and gaming to scheduling problems [100].

Informally, genetic algorithms operate on a *population* of individuals, whose relevant information is encoded into *chromosomes* consisting of a set of values—known as *genes*—for each relevant feature of the data. The population "evolves" across generations by recursively applying recombination and transformation operators to the previous generation, with the aim of gradually obtaining *fitter* individuals with respect to the objective function.

The Basic Genetic Algorithm

The basic genetic algorithm consists of the following steps:

1. **Initialisation:** Typically, a random population will be generated with chromosomes, i.e. an ordered tuple of valid gene-values, where each gene represents a different data feature.

2. **Fitness or Evaluation:** The objective function is evaluated on each chromosome of the population; this evaluation will determine the *fitness* of the chromosomes, i.e. how good they are in comparison to the other chromosomes with respect to the objective function.

3. **Selection:** An intermediate population is built by copying chromosomes from the current population—possibly more than once—where the probability of chromosomes being copied is proportional to their respective fitness, i.e. fitter chromosomes are more likely to be copied.

4. **Crossover:** A fraction of the intermediate population chromosomes—determined by a *crossover rate* parameter—are selected to be *parents* of new chromosomes according to their fitness (where fitter chromosomes are more likely to be selected) to form new *offspring* (or *children*) to replace them. Crossover is a recombination operator that may happen in several ways. Two standard crossover operators are:

Single-Point Crossover: A recombination point, i.e. a gene-index, is randomly chosen. Offspring are then generated by *swapping* all the genes after the recombination point for two parents. For example, consider chromosomes `abcde` and `12345`. If the recombination point were $i = 3$, the resulting offspring would be chromosomes `ab345` and `12cde`.

Two-Point Crossover: A *start* and an *end* recombination points are randomly chosen, and only the genes between those two points are swapped. If we think again of chromosomes `abcde` and `12345` as parents, and assume that $start = 2$ and $end = 4$, the resulting offspring would be chromosomes `a234e` and `1bcd5`.

5. **Mutation:** Each *gene* in the intermediate population may independently mutate with a small probability, determined by a *mutation rate* parameter. This allows for additional variability in the generated chromosomes, which in turn permits the exploration of previously unknown regions in the search-space.

Steps 2–5 are repeated over the resulting populations until a stop-condition is met. This could be, for example, finding a sufficiently fit individual, reaching a set number of iterations or exhausting a computational budget. Each consecutive execution of steps 2–5 will be referred to as a *generation* of the genetic algorithm. FAIRPIPES, the pipeline optimisation algorithm presented in Chapter 4, is inspired by the basic genetic algorithm, although it has many particularities described in the aforementioned chapter.

2.5 Data Preprocessing

Data preprocessing will be defined as the set of steps that transform the raw input data into its final form as a training set. Several catalogues and classifications have been proposed for data preprocessing operators, e.g. ML Bazaar [145], Orange [40] and others [58].

Some of these steps are required by the classification framework, e.g. encoding categorical variables and imputing missing data, while others may optionally be deployed, e.g. class balancing, scaling and feature selection. These steps are generally selected and combined into pipelines based on best practice considerations, with model performance, e.g. one or more of the metrics discussed in Section 2.3 as the main objective [129]. Some of the most common preprocessing steps are now described, exemplifying different ways in which each of these steps may be performed. It is important to note that all the parameter-tuning decisions made for each preprocessor must come, i.e. be *fitted*, on the training set alone.

2.5.1 Feature Encoding

Data may come in different formats [58], such as discrete numerical, e.g. the number of rooms in a house, continuous numerical, e.g. the surface area of the house and categorical, e.g. the neighbourhood where the house is. Classifiers have distinct requirements regarding the way they understand data. Therefore, a raw dataset will most of the time require transforming the different variables into a format that may be processed by the classifier of choice. The most common transformation is to convert a categorical variable into numerical. There are several ways in which this may be achieved, each having its set of pros and cons. Across this thesis, the following categorical encoding methods were used. For all of them, the assumption will be that a categorical feature x with n possible values is to be encoded.

One-Hot The original categorical feature x is dropped and n new zero-columns—one per category—are created. For any instance, the column corresponding to the instance's x-value gets assigned the value 1, leaving the rest unmodified. This method is highly interpretable and yields binary new features, both desirable qualities in a dataset. However, this comes with the associated cost of a high dimensionality in the resulting dataset, especially when x has many possible values, e.g. the nationality of an instance. This in turn may greatly increase the volume of a dataset, as well as the number of parameters to be learned by the classifier [143]. Another problem with this method is that it provokes multicollinearity among the resulting one-hot encoded features, as their sum will always equal 1, preventing certain linear models, such as non-regularised LR, from working [49].

Ordinal Each value gets mapped to an integer in $\{0, 1, \ldots, n-1\}$. This simple encoding may work well when the categorical feature has an underlying order, e.g. low/medium/high and it does not increase the dimensionality of the training set [53], but may otherwise cause artificial trends to be detected by a classifier.

Target Each categorical value x_i gets replaced with the posterior probability of a positive classification given x_i [117]. This method does not increase the dimensionality of a dataset, but given that it correlates the categorical feature with the classification, the resulting model may be prone to overfitting [105].

LOO A slight variation of *Target* encoding, where the current data instance is excluded from the posterior calculations, to reduce the effect of outliers [113]. Like *Target*, with LOO the dimensionality is not increased, but there is a risk of overfitting the resulting model.

Weight of Evidence Each categorical value x_i gets replaced by its *weight of evidence (WoE)*: $\text{WoE}(x_i) = \ln \frac{\mathbb{P}(Y=1|x=x_i)}{\mathbb{P}(Y=1|x\neq x_i)}$ [60]. Like *Target* and *LOO*, the dimensionality is not increased, but there is a risk of overfitting, and the calculation may be undefined when either the numerator or the denominator become zero.

Count Replaces each x_i value with its number of occurrences in the training set [113]. Again, no dimensionality is added. However, there is the risk of confounding two different x-values if their counts are the same.

2.5.2 Dealing with Missing Values

For most real-world datasets, there will be missing data for some instances of it. There are many different reasons for this, ranging from problems in data collection, e.g. a sensor did not register a particular observation, up to a feature not being applicable for a given instance [96]. Missing values may occur in three different ways [120]:

Missing at Random (MAR) means that missing data in variable X_i depends on the value of variable X_j, with $i \neq j$. For example, school grades obviously depend on whether an individual actually attended school or not.

Missing Not at Random (MNAR) means that missing data in variable X_i depends on X_i itself. For example, a sensor might not record temperatures below or above a certain threshold.

Missing Completely at Random (MCAR) means that missing data occurrences are independent of all variables in the data. These could happen, for example, because there was a human error while capturing the data.

Four standard ways to deal with missing data are:

1. Removing the features having missing values. This is the most drastic—and arguably the worst—possible solution, as it may cause a big proportion of the data to disappear.

2. Removing the instances with missing values. This solution is not as drastic, as only the problematic instances get deleted. However, if there are many missing values for different features, the number of deleted instances may still not be insignificant.

3. Treating missing (or NA) as a special value, i.e. assigning missing values their own category. This solution can work well on unordered categorical features, but on ordinal features mapping a particular number to missing data, e.g. 0 can be problematic [156].

4. Imputing the missing values with existing or synthetic values extracted from the dataset; Janssen et al. [84] argue this is usually preferable. There are many different ways in which this solution can be implemented. The most common ones use a statistical measure of the feature—e.g. mean, median or mode—either over the whole dataset or over the instance's corresponding class. It is also possible to apply statistical learning methods such as MICE [157] or KNN [34] in order to produce the imputed value, using the rest of the dataset's features as predictors [104].

In the Fair-ML context, missing data may not be MCAR due to different reasons, such as unfavoured individuals being reluctant to answer questions that could be used against them, or data may be less complete for minority groups [51]. Therefore, missing data correction methods, such as removing instances with missing values, could have a disparate impact across population groups, making it an even less suitable preprocessing choice.

2.5.3 Class Balancing

Real-world datasets will often be imbalanced with respect to their class distribution [107], i.e. some class values (or values of particularly important features) are substantially more frequent than others. There are several possible reasons for this imbalance, ranging from the actual class distribution—where some classes may appear less frequently than others, e.g. people with a rare blood type—to improper data collection, e.g. an Internet survey may underrepresent the poorer sectors of the population, since their access to the Web may be more limited. In any case, class imbalance may produce the undesirable effect of classifiers with low predictive performance *for the minority classes* by focusing solely on correctly predicting the majority classes; in an extreme scenario with very few minority instances, even the trivial classifier labelling every instance as members of the majority class will have a great overall predictive performance.

In order to correct class imbalance, three possible solutions were considered:

1. To *undersample* the majority classes. This may be done by dropping instances belonging to the majority class, either randomly or systematically [98]. The main problem with this solution is that valuable data is discarded, which will have a negative impact on the resulting classifier's performance for the majority class.

2. To *oversample* the minority classes. Oversampling may be performed either by randomly duplicating existing minority class instances or by generating synthetic minority examples that approximate real minority instances [24].

3. A combination of undersampling the majority classes and oversampling the minority classes [158].

An alternative solution to class imbalance is *cost-sensitive classification*, in which classes are given different misclassification costs, and the goal is to minimise the total misclassification cost [75].

2.5.4 Feature Scaling

Classifiers based on distances or similarities, such as K Nearest Neighbours and Support Vector Machines, are sensitive to the scale of feature values [144]. For example, a classifier predicting whether individuals are healthy or not may assign a greater importance to the person's salary than to their weight, solely based on the different scales of both features. For this reason, transforming the data so that all the features share a common scale is often desirable. There are many different ways in which this can be achieved, and selecting the optimal transformation will be data-and-problem specific. The following are the most common scaling techniques [142]:

z-Score Scaler Also known as *standard scaler*, the z-score scaling of feature X is given by:

$$z = \frac{x - \mu}{s}$$

where μ is the mean and s is the standard deviation of X. This centers X around 0, and makes its values commensurable with the standard normal distribution.

MinMax Scales each feature X into the $[0,1]$ range [124]. This transformation is given by:

$$x_{\text{MinMax}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}.$$

MaxAbs Scaler Scales each feature X by its maximum absolute value, so that for each feature the resulting values will be in the range $[-1,1]$. This transformation is given by:

$$x_{\text{MaxAbs}} = \frac{x}{|x|_{\max}}.$$

Quantile Transformer Transforms the features to follow a discrete uniform distribution so that each quantile (a user-defined parameter specifying the number of bins in which to accommodate the data) has the same number of instances, reducing the effect of outliers on the resulting classifiers. However, the non-linear nature of this transformation may distort correlations between features [142].

Normaliser Scales each instance to have unit norm, i.e.

$$\left(\sum_{i \in I} x_i^2\right)^{\frac{1}{2}} = 1,$$

where I are the indices for all feature variables. Data normalisation is a common operation for text classification [152]. Importantly, this method will convert categorical features into continuous, with its new values being impossible to interpret.

2.5.5 Feature Selection

Feature selection refers to discarding unnecessary or redundant variables from data, in order to reduce both the computational cost of training a classifier as well as the data noise resultant from these redundancies [23]. There are many different criteria to select which features should be discarded, yet they can can be grouped into three main categories: filters, wrappers and embedded methods [154, 68].

Filters are algorithms that select variables without evaluating the metric of interest—e.g. the accuracy—on feature subsets, but instead rely on auxiliary statistical metrics, discarding the features that do not meet a predefined threshold, e.g. features with very low variance.

Wrappers perform classifier evaluations on feature subsets, and preserve either a fixed number—e.g. the k best—or a percentage of the total number of features.

Embedded Methods perform variable selection during the learning process, and are usually classifier-specific.

The presented preprocessing operators, either in their original form or modified ad hoc, are the base of the correction and optimisation algorithms presented in Chapters 4, 5 and 6. The following section introduces the basic algorithmic fairness notions used in the subsequent chapters.

2.6 Algorithmic Fairness

There is a major division in fairness definitions, namely the one between group and individual fairness. While the former refer to a statistical analysis of a classifier's behaviour

across data groups, e.g. gender or race, the latter strives for the ideal of "similar" individuals receiving similar classifications [46]. The notion of individual fairness, although intuitively easier to grasp, brings forward the hard follow-up problem of determining whether two individuals are similar or not, i.e. of determining a fair metric for the data. Although some research has been done with respect to individual fairness [170, 102, 87], most of the existing work on fair classification—including the works presented in this thesis—has to do with group fairness.

The basic group fairness definitions used are now presented, together with the reasoning behind the focus on the choice of these metrics over other existing ones.

2.6.1 Basic Definitions

A binary classifier's class label can be *positive* or *negative* referring to the desirable and non-desirable outcome of a prediction, respectively. For example, in a classification task deciding whether to grant a user a bank loan, the positive label would refer to getting the loan, and the negative one to being rejected.

A dataset's *protected attribute (PA)* refers to a variable that may be the object of discrimination, due to historical bias or otherwise. Although both the number of PAs and the number of groups in a PA may and sometimes will be greater than two, the following definitions will assume a single binary PA, meaning there will only be two PA groups, every instance in the dataset belonging to one of those.

The ratio of the number of positive instances divided by the total number of instances in a specific group will be called the *positive ratio (PR)* of the group.

Among the two PA groups, the one having the highest PR will be referred to as the *favoured* group F, while the other one will be referred to as the *unfavoured* group U. When required, the positive and negative instances of F and U will be referred to as F^+, F^-, U^+ and U^-, respectively.

2.6.2 Group Fairness Definitions

This thesis' analyses are based on four fairness definitions, formalised next: *demographic parity* [46], *equality of opportunity* [72], *equalised odds* [72] and *proxy fairness* [94]. In these definitions, the positive label is identified with $Y = 1$ and the negative label with $Y = 0$.

Definition 1 (Demographic Parity)**.** A classifier satisfies *demographic parity (DP)* if the probability of being classified as positive is the same across PA subgroups:

$$\mathbb{P}(\hat{Y} = 1 \mid \text{PA} = U) = \mathbb{P}(\hat{Y} = 1 \mid \text{PA} = F).$$

Definition 2 (Equality of Opportunity)**.** A classifier satisfies *equality of opportunity (EOp)* if the probability of being classified as positive *for actual positives* is the same across PA subgroups:

$$\mathbb{P}(\hat{Y} = 1 \mid \text{PA} = U, Y = 1) = \mathbb{P}(\hat{Y} = 1 \mid \text{PA} = F, Y = 1).$$

Definition 3 (Equalised Odds)**.** A classifier satisfies equalised odds (EO) if the probability of being classified correctly is the same across PA subgroups:

$$\mathbb{P}(\hat{Y} = y \mid \text{PA} = U, Y = y) = \mathbb{P}(\hat{Y} = y \mid \text{PA} = F, Y = y), \;\; y \in \{0, 1\}.$$

Definition 4 (Proxy Fairness)**.** A classifier $\hat{Y}$ satisfies proxy fairness (PF) if the interventions $do\,(\text{PA} = U)$ and $do\,(\text{PA} = F)$ result in the same probability of being classified as positive:

$$\mathbb{P}(\hat{Y} = 1 \mid do\,(\text{PA} = U)) = \mathbb{P}(\hat{Y} = 1 \mid do\,(\text{PA} = F)).$$

The *do* operator is defined as an *intervention* on the test set T. To evaluate PF, this intervention is performed independently two times, assigning every individual in T the PA-values U and F, resulting in $T_{\text{PA}=U}$ and $T_{\text{PA}=F}$, respectively. The process is illustrated in Figure 2.2.

These definitions, as well as others [101], have been proposed in the literature, and they are sometimes in conflict with one another: a decision rule that satisfies one of the definitions may well prove to be unfair for a different one [29]. For example, determining university admissions through gender quotas may achieve DP, but it makes the acceptance rates for good students of different genders disparate, i.e. EOp is not satisfied.

The reasons to focus the analyses on these particular metrics are four-fold:

1. These definitions align with the prevalent world-views of equality in the social sciences [133]. DP aligns with *equality of outcomes*, an often referred to world-view

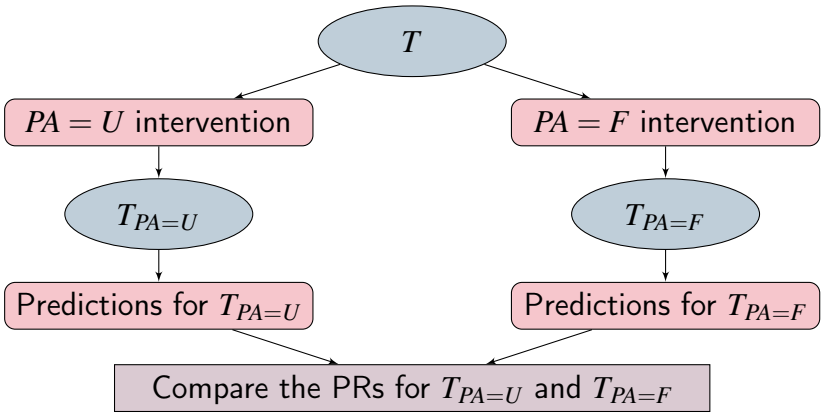


Figure 2.2 Computing the proxy fairness of test set T.

in the social equality literature [127]. According to this, aggregated differences in skill or condition among individuals belonging to different population groups, e.g. gender or race, are the consequence of historical and social injustices. Therefore, in an ideal world the probabilities of every outcome should be the same across groups [46]. In the United States labor law, the *80% rule* [14] is presented as a simple rule to determine whether a company's hiring procedures have an adverse impact on a PA subgroup. This rule specifies that the ratio of the hiring rates among applicants belonging to different PA subgroups, e.g. $\text{PR}(U)/\text{PR}(F)$ should be higher than 0.8 [115]. An opposing social view of fairness is EOp [133], where individuals *deserving* a positive outcome have the same probability of obtaining it, regardless of their group adherence. This view, however, allows for the possibility of different population groups performing differently with respect to the outcome of interest. An even more severe restriction to this is EO, where, additionally to deserving individuals, *undeserving* individuals share the same probability of a positive outcome (not necessarily equal to the probability for the deserving individuals).

2. A more formal approach to the connection between fairness definitions and world-views is presented in Friedler et al. [55], where world-views are defined as the set of assumptions made about a *construct*—a metric space including all the features that are relevant to a decision task, i.e. the data plus possibly additional beliefs. Three notable world-views are introduced in this paper: *what you see is what you get (WYSIWYG)*, *we are all equal (WAAE)* and *structural bias*. WYSIWYG refers to the assumption of the construct space being the same as the available data. WAAE, on the other hand, assumes that all groups of individuals with respect to a protected attribute should perform equally well with respect to the classification task. Finally,

Table 2.2 Datasets used for this thesis' experiments.

Dataset	PA	Favoured	Positive Class	Attributes	Instances
Income	Sex	Male	Over $50k income	14	48842
COMPAS	Race	White	Will not recidivate	27	6907
German	Sex	Male	Will repay loan	22	1000
Titanic	Sex	Female	Survived	10	891

structural bias assumes that protected attribute groups may play a more relevant role in the decision rule than they should. This calls for a modified decision rule to correct this distortion, i.e. *taking affirmative action (AA)*.

3. Proxy fairness is closely related to fairness through unawareness (FTU) through a simple intervention: assign the same PA value to *every* instance in the test dataset, once per possible PA-value, and compare the resulting classification PRs of these modified test sets. If the PRs are the same, this means that the PA has no impact over the obtained label, i.e. the classifier is "PA-blind".

4. These definitions are easy to measure and intepret on real-world data and predictions: in most cases all that needs to be done is to evaluate the positive ratios for the whole test set as well as for the test set subgroups of interest, namely F^+, F^-, U^+ and U^-.

2.6.3 Benchmark Datasets

The experiments presented in Chapters 5 and 6 were performed on three datasets commonly used in the ML fairness research literature: *Adult Income (Income)* [44], *COMPAS* [106] and *German Credit (German)* [44]. The experiments in Chapter 4 were additionally performed on the *Titanic* [21] dataset. A summary of the main features of each of these datasets may be seen in Table 2.2.

Sections 2.2 and 2.3 introduced the basic ML concepts necessary to talk about algorithmic fairness in classification tasks. Afterwards, in Section 2.6 the basic fairness definitions employed in the subsequent chapters were presented. In Chapter 3, a survey of works related to Chapters 4, 5 and 6 is presented.

Chapter 3

Related Work

Contents

Summary

This chapter seeks to provide context for the contributions presented in this thesis by examining the relevant background material and related work in algorithmic fairness. Existing tools to detect bias in data that may lead to unfairness are discussed in Section 3.1 and the main problem with enforcing a particular fairness definition, the *fairness/accuracy* trade-off, is discussed in Section 3.2.

Existing AutoML solutions that perform automatic preprocessing pipeline optimisation with performance as their objective are presented. These methods are related to the FAIRPIPES algorithm, presented in Chapter 4. Afterwards, a discussion on the existing fairness-correcting methods for classification tasks and focusing on preprocessing methods is presented in Section 3.4, as these are related to the PARDS algorithm presented in Chapter 5. Finally, papers dealing with the relation between fairness and privacy are considered, as these are related to the FAIR-MDAV algorithm, presented in Chapter 6.

3.1 Detecting Unfairness

A first step towards algorithmic fairness is being able to detect whether a dataset is biased regarding the PAs of interest. This may be detected directly from the training data for many fairness definitions, e.g. DP, EOp or EO. Many unfairness detection tools are available [169, 137, 9], with IBM's AI Fairness 360 [10] being a popular open source solution that is continuously being updated.

It is important to note that having a dataset satisfy a fairness definition does not necessarily imply that the predictions of any classifier learnt from it will also be fair. In Chapter 5 it is shown that the fairness of a dataset D and the fairness of predictions by a classifier learnt from D are directly correlated, with different datasets and classifiers requiring specific amounts of fairness correction in the dataset to achieve optimal predictive fairness.

According to Mitchell et al. [118], there are two main sources of algorithmic unfairness: *statistical* bias and *societal* bias. Statistical bias refers to a mismatch between the dataset used to train a classifier and the true data relationship being modelled across the whole population due to inadequate sampling [103]. Societal bias, on the other hand, represents an additional layer to the fairness problem: the desire to model an ideal world in which a particular fairness world-view [55] is satisfied, e.g. WAAE. In both cases, it may be necessary to *correct* a learnt classifier to satisfy the desired fairness behaviour.

3.2 The Fairness/Accuracy Trade-Off

A common assumption in the algorithmic fairness literature is the existence of a trade-off between the performance and the fairness of a classifier [116, 26, 172, 31]: the cost of enforcing the latter is a penalty with respect to the former. This trade-off is particularly severe in the case of DP correction over a biased dataset: in order to achieve similar positive ratios across both PA groups, it becomes necessary to classify U^- instances as positive or F^+ instances as negative, therefore diminishing the overall predictive performance of the classifier. In the case of other fairness definitions, e.g. EOp and especially EO, however, the trade-off is less dramatic—yet still existent—as these definitions are centered around making correct classifications anyway.

Having two or more metrics in tension with each other, e.g. DP and ACC, means that fairness correction may be thought of as a multi-objective optimisation problem. Given a classification problem there must be a collection of optimal solutions consisting of different levels of fairness and accuracy, ranging from optimal predictive performance with a relatively poor level of fairness to the opposite: optimal fairness with a relatively poor predictive performance. This set of solutions, for which no metric may be improved without a negative impact on the other, is known as a *Pareto front* [121], consisting of *Pareto-optimal* solutions. A set of classifiers and their Pareto front (taken from a Chapter 4 experiment) is shown in Figure 3.1.

In Chapter 4 a genetic-algorithm based solution that explicitly assigns relative importances to a combination of fairness and performance objectives is presented, and in chapters 5 and 6, fairness correction solutions that indirectly control the fairness/accuracy trade-off through a continuous correction parameter are presented as well.

3.3 Pipeline Optimisation

This section presents work related to the FAIRPIPES algorithm briefly described in Section 1.5 and presented in detail in Chapter 4. FAIRPIPES is a genetic algorithm aimed at optimising data preprocessing pipelines, both through the selection of specific preprocessing operators as well as by applying the resulting operators in the best possible order.

Fairness-aware preprocessing is usually attained by applying fairness-specific methods, like the ones that will be described in Section 3.4. However, these are not always readily available, and they may cause undesired side-effects such as loss of accuracy, since they involve the introduction of synthetic data into the original dataset, e.g. by synthetic

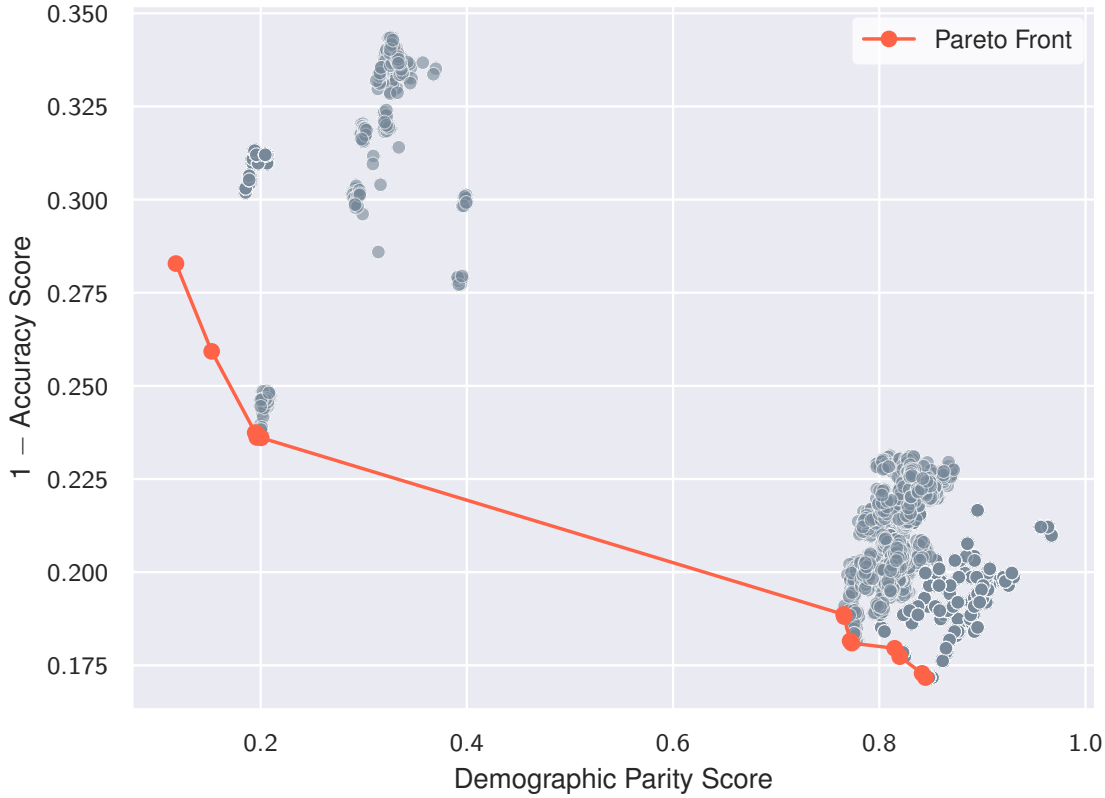


Figure 3.1 Fairness (DP) and performance ($1 - \text{ACC}$) metrics for a set $\mathscr{C}$ of classifiers used on the *Adult Income* dataset (gray scatter points). The fairness/accuracy trade-off may be observed on the Pareto front (shown in orange), as better ACC scores (lower points) are further to the right (worse DP score).

oversampling [25]. It is important to note that FAIRPIPES *does not* directly compete with these fairness-correcting methods. Instead, FAIRPIPES can easily be adapted to work on top of existing preprocessing methods, incorporating them to its pipeline search-space.

Evolutionary computation search, the approach used by FAIRPIPES, has been used before to optimise specific data preprocessing tasks, e.g. for feature selection [149, 5, 123, 153, 136] and data correction [3], as well as to build full preprocessing pipelines, e.g. TPOT [122], which is commented on next.

Automating the process of tuning and training classifiers is an active research topic. Notable examples are TPOT [122], ML Bazaar [145], H2O AutoML [108], and auto-sklearn [52]. These tools take different approaches for their optimisations: TPOT, one of the most popular AutoML solutions, is the closest in spirit to FAIRPIPES, as they are both based on evolutionary computation; while TPOT uses *genetic programming*, FAIRPIPES is a genetic algorithm. In contrast, ML Bazaar and auto-sklearn are based on Bayesian optimisation. H2O AutoML is based on stacked ensembles, where a meta-learner is trained to find the optimal combination of the base learners, combined with random search. While some of these methods do some preprocessing optimisation, for none of them it is the main component of their optimisation, as they are mostly focused on model selection and hyperparameter tuning. Table 3.1 compares the preprocessor options offered by FAIRPIPES and the aforementioned packages.

3.4 Fairness Correction for Classification Tasks

A classifier's predictive fairness may be adjusted by the combination of one or more of the following approaches: *wrangling* multiple data sources adequately into a training dataset [112], *preprocessing* the training data, *in-processing* the learning algorithm [1, 22, 168, 167, 170] or *post-processing* a classifier's predictions [72]. The works presented in this thesis' subsequent chapters adhere to the preprocessing approach, since it presents two distinct advantages over in-processing and post-processing correction:

- Preprocessing methods are classifier-agnostic, i.e. they will work regardless of the chosen classifier. In contrast, most in-processing approaches consist of a modification over an existing classifier, e.g. fairness-aware LR and SVM [167] or Naive Bayes [19], or by adding a fairness regularisation term, a strategy that would only apply to certain classifiers such as LR [91].

- The introduced corrections are transparent and auditable by the user: it is possible to quantify and report the changes introduced into data.

Table 3.1 Comparison of preprocessors offered by FAIRPIPES with popular AutoML packages.

Package	Encoders	Imputers	Scalers	Samplers	Feature Selectors
FAIRPIPES	One-Hot, LOO, Target, Count, WoE, Ordinal	Mean, Median, Most Frequent	None, MinMax, MaxAbs, Quantile, Normaliser	None, Over, Under	None, K-Best
TPOT 0.9	One-Hot	Median	None, MinMax, MaxAbs, Robust, Standard	None	None, RFE, SelectFromModel, SelectFweSelectFwe, SelectPercentile, VarianceThreshold
ML Bazaar 0.3.2	Categorical, Label, One Hot	Mean	None, MinMax, MaxAbs, Robust	None	None, Extra Trees, Lasso
auto-sklearn 0.14.7	One-Hot	Mean, Most Frequent	None, Normaliser	None	None, Polynomial, PCA, Extra Trees
H2O 3.32.1.7	Target	Mean, Median, Most Frequent	None	None, Over, Under	None

Therefore, the following section focuses on preprocessing correction.

3.5 Fairness-Aware Preprocessing

The works discussed in this section are closely related to PARDS, the *data resampling* algorithm briefly described in Section 1.5 and presented in detail in Chapter 5. Fairness-aware preprocessing is defined by Friedler et al. [56] as a set of techniques that modify the input data so that any classifier trained on such data will be fair.

According to Kamiran and Calders [90], there are four main ways in which to make appropriate adjustments to data in order to enforce fairness: suppressing certain features, e.g. FTU [57], *massaging* variable values [27, 50, 20], reweighing features [97, 81], and resampling data instances [132, 89, 138, 25]. Further elaboration and examples of each of these four preprocessing strategies now follows.

Feature Suppression Feature suppression is closely related to *FTU*. This technique relies on the idea of making the PAs "invisible" to a classifier. This is the simplest possible intervention, as all that is needed is to remove the relevant variables from the training set. The problem with this approach is that often other variables may be strongly correlated with the PAs, and hence the discriminatory behaviour of a classifier will persist even without considering the PA in the model. Even more aggravating is the fact that some fairness definitions, such as DP, require an *affirmative action* to improve the classifications of the unfavoured group's instances, and removing the PA makes this sort of correction unfeasible.

Massaging Variable Values This method consists in modifying the class labels for specific instances in the data, e.g. moving instances from F^+ to F^- and from U^- to U^+, in order to modify the behaviour of the learnt classifier. It is generally a good idea to change the labels of instances located near the decision boundary [90]; in order to detect these instances, a ranker that sorts the instances according to their probability of belonging to the positive class may be used [90].

Reweighing Instances Instead of modifying class labels, another solution is to assign different weights to specific instances in the training set, so that the regularisation term penalises wrong classifications differently for each of the PA-label subgroups [18].

Data Resampling Data resampling is less invasive in nature than FTU or massaging, since the original data is preserved and only the frequency with which the instances are represented is modified. In contrast, FTU disposes of large amounts of data without a guarantee on the effect of said intervention and massaging effectively creates synthetic data, which does not necessarily reflect the ground truth. *Preferential Sampling (PS)* [89] resamples the F/U and positive/negative combinations separately in order to equalise F's and U's PRs. In Chapter 5 it is empirically shown that the optimal fairness correction depends on the selected sampling method, classifier and fairness definition. Therefore, equalising the positive ratios across protected attribute groups is not necessarily the best approach. *SMOTEBoost* [25] over-samples the minority group through synthetic data based on real data instances, with a focus on improved minority predictions and indirectly improving fairness. Since it over-samples with synthetic data, it may either be considered a massaging or a resampling method. *SMOTEBoost* has a correcting parameter k which modulates the amount of correction introduced into the dataset. Other related methods include *Capuchin* [138], a causal-fairness non-parametric resampling method and Feldman et al. [50], a massaging method where parameter λ is used to create linear interpolations of the original dataset and a *repaired* copy to find the optimal combination.

Dealing with Multiple PAs

While most of the work presented in Chapters 4, 5 and 6—and indeed, most of the existing work on fair classification—is focused on the single binary PA classification problem, it is possible to generalise this to a multi-class PA, multiple binary PAs and even multiple multi-class PAs. In Chapter 5 one way to deal with the third, more general, situation is proposed, by generating a *combined* PA out of many multi-class PAs by comparing the positive ratio for each instance's PA values against the overall dataset's positive ratio and aggregating the resulting differences. Although in practice this approach works well for real-world datasets, it is theoretically prone to fail on *fairness gerrymandering* situations [93]. These situations happen when the intersection of two *unfavoured* PA groups becomes *favoured*, i.e. the PR of the intersection is greater than the dataset's. Although it is hard to present an example of this phenomenon, the problem has been addressed and shown to be computationally hard by several authors [93, 76, 171, 1].

3.6 Fairness and Privacy

This section presents work related to the FAIR-MDAV algorithm, briefly described in Section 1.5 and presented in detail in Chapter 6. Certain fairness goals, e.g. EO, share a common target with privacy: to represent data in such a way that it is informative of the task output, but uninformative of the sensitive information [131]. Privacy strives to protect information from disclosure, while fairness seeks to prevent certain classification behaviours related to the PA [38]. Ekstrand et al. [47] argue in favour of integrating fairness research to socio-technical systems that provide privacy protection. A k-anonymisation method was used in Hajian et al. [69] to protect frequent patterns with fairness. However, most of the work connecting privacy and fairness has to do with *differential privacy*, a technique based on introducing a controlled amount—determined by a parameter ε—of noise to a dataset in order to prevent determining an individual instance's presence in the dataset [45].

Differential privacy may have disparate impact for the PA subgroups in both performance and fairness. Bagdasaryan et al. [8], Pujol et al. [128] show that the reduction in accuracy incurred by deep differential private models disproportionately impacts *underrepresented* PA groups, with differential privacy amplifying the model's bias towards the most popular groups. Cummings et al. [35] show that if DP is satisfied by an ε-differentially private algorithm, this is caused by trivial-accuracy classifiers, i.e. a classifier that predicts every instance with the same label—positive or negative, whichever label occurs more frequently. However, ε-differentially private LR models with near-DP fairness differences are presented by Xu et al. [164].

Foulds et al. [54] define a multi-attribute fairness metric, *differential fairness*, inspired by the use of ε in differential privacy, and extend the notion of the 80% rule [14], defined in Subsection 2.6.1: in order to satisfy ε-differential fairness, the quotient of any two PA subgroups' PR should be greater than $e^{-\varepsilon}$ and smaller than e^{ε}. Jagielski et al. [82] introduce *private EO*, a post-processing method which is differentially private with respect to the PA and achieves EO fairness, at the cost of low accuracy. Finally, Dwork et al. [46] discuss conditions upon which a particular fairness definition implies privacy, and how differential privacy may be related to such fairness definition.

3.7 Fairness in Other ML Domains

Although this thesis' research focuses on fair classification, other domains of ML have their own fairness paradigms and definitions. Two specific domains are briefly mentioned

here: fair regression problems, which aim at predicting continuous values such as fairly assigning a line of credit to bank clients, and fair clustering problems, i.e. dealing with making each cluster demographically representative of the population.

Counterfactual Fairness A specific type of individual fairness that has been researched extensively is *counterfactual fairness* [101, 27, 59, 163], which is based on expert-created causal models that explain the relationship between the variables of a dataset. A classifier is said to satisfy counterfactual fairness if modifying the PAs with every possible set of values does not affect the prediction probabilities for every data instance. The main problem with this approach is that given a dataset, there are multiple possible causal models for it [125], and different models may present different fairness behaviours.

Fair Regression Fair regression, where the quantity to predict is a continuous variable, has not been studied as much as fair classification. In the regression case, two fairness definitions are used: *statistical parity*, the continuous analogue to DP, which requires the prediction to be statistically independent of the PA, and *bounded group loss*, which requires that the prediction error with respect to to each PA remains below a pre-determined level [2]. While the regression problem was not analysed, given the similarity of the fairness definitions in both domains, in the future it would be interesting to analyse whether this thesis' approaches to fairness correction could be modified to work in the regression case.

Fair Clustering The main concern regarding fair clustering is to give each PA group approximately the same proportion in every generated cluster [28]. This is achieved by decomposing the data into *fairlets*, small data subsets that have similar PA group proportions as the whole data, and then clustering these fairlets into the most efficient configuration possible, according to the chosen clustering algorithm, e.g. k-means or k-medians. Finding the fairlets of a dataset, though, is computationally expensive. While the algorithm provided in the original paper can find them in quadratic time, an improved algorithm by Backurs et al. [7] can find them in nearly linear time. The solution to fairness and privacy presented in Chapter 6 borrows concepts from fair clustering.

Chapter 4

Genetic Pipeline Optimisation

Contents

Summary

Improving the fairness of classifiers by manipulating the preprocessing stages of the learning pipelines is an active area of research, closely related to AutoML. Yet most approaches optimise for one specific fairness metric, and neglect to account for the contrasting requirement of preserving the model's predictive performance (ACC). In contrast, FAIRPIPES is proposed: a genetic optimisation algorithm which optimises for user-defined combinations of fairness and ACC and for multiple definitions of fairness, providing flexibility in the fairness/accuracy trade-off. FAIRPIPES heuristically searches through a large space of pipeline configurations, achieving near-optimality efficiently and presenting the user with an estimate of the solutions' Pareto front. Given one binary protected attribute and a binary classification problem, FAIRPIPES evolves an initial population of seed pipelines through multiple crossover and mutation iterations. The approach was evaluated experimentally using three fairness metrics: demographic parity difference (DPD), *equality of opportunity difference (EOpD)* and equalised odds difference (EOD), and five types of preprocessors: encoders, imputers, scalers, resamplers and feature selectors, each provided with multiple specific methods, resulting in a large search space of 3,240 pipelines. When tested on three well-established benchmark datasets: *Income*, *COMPAS*, *German* and *Titanic*, FAIRPIPES achieves near-optimality across a range of combinations for the fairness/accuracy optimisation objectives, while exploring about 6% of the search space. The optimal pipelines were observed to differ for different datasets, suggesting that no "universal best" pipeline exists and confirming that FAIRPIPES fills a niche in the *fairness-aware AutoML* space.

4.1 Introduction

The focus of this chapter is on the data preprocessing steps that are deployed to transform the raw input data into its final form as a training set, and on their effect on the fairness and predictive performance of the resulting model. Having two objectives which may be in opposition with each other indicates that finding the adequate pipeline should be treated as a multi-objective optimisation problem. This means that the optimal solution will not be unique, but a range of such solutions will exist, each of these presenting a different fairness/performance trade-off, as discussed in Section 3.2; the choice among these solutions will then be decided by the user's preferences or by external factors, such as legal regulations.

As a motivating example, consider a company having presence in three different countries, each of them having different regulations with respect to anti-discrimatory hiring

policies. Country A requires that the hiring proportion for males and females is 1, i.e. the same number of males and females are hired. Country B, on the other hand, requires the difference between male and female hiring numbers to be less than 10%. Finally, country C has no anti-discriminatory hiring regulations at all. If the company wants to use a classifier to select the applications to hire which has the best possible accuracy while still complying with each country's regulations, the accuracy will be different across countries. Figure 4.1 presents such a case, where three different optimal classifiers—resulting from different preprocessing pipelines—can be used to satisfy the countries' different fairness constraints. Having access to this set of optimal solutions will let the company know the expected predictive performance of the classifiers in the three countries, as well as on any other place where the company decides to establish itself.

Several catalogues and classifications have been proposed for data preprocessing operators, e.g. by ML Bazaar [145], Orange [40] and others [58]. A summary of the most common preprocessing steps is given in Table 3.1, where the operators are grouped into five categories: encoders, imputers, scalers, samplers, and feature selectors; the top row lists the strategies considered in this chapter. These are compared with the operators used in related work, as discussed in Section 3.3. Some of these steps are required by the classification framework, e.g. encoding categorical variables and imputing missing data, while others may optionally be deployed to improve model performance, e.g. class balancing, scaling and feature selection. These steps are generally selected and combined into pipelines based on best practice considerations, with model performance as the main objective [129]. While the effect of preprocessing on classification performance has been analysed for individual operators [33, 155, 62], the effect of such preprocessing on the fairness of the resulting classifier is studied here.

4.1.1 Fairness Differences

For this work, fairness definitions were rewritten in their "differences" form, as follows:

$$\begin{aligned}
\mathrm{DPD}(\hat{Y}) &\coloneqq |\mathbb{P}(\hat{Y}=1 \mid \mathrm{PA}=F) - \mathbb{P}(\hat{Y}=1 \mid \mathrm{PA}=U)|, \\
\mathrm{EOpD}(\hat{Y}) &\coloneqq |\mathbb{P}(\hat{Y}=1 \mid \mathrm{PA}=F, Y=1) - \mathbb{P}(\hat{Y}=1 \mid \mathrm{PA}=U, Y=1)|, \\
\mathrm{EOD}(\hat{Y}) &\coloneqq \sum_{y\in\{0,1\}} |\mathbb{P}(\hat{Y}=1 \mid \mathrm{PA}=F, Y=y) - \mathbb{P}(\hat{Y}=1 \mid \mathrm{PA}=U, Y=y)|.
\end{aligned}$$

For all three differences, small values indicate a "fairer" model.

Table 4.1 Preprocessor options for the preliminary experiment.

Preprocessor	Options
Encoder	One-Hot, Ordinal, Target, LOO, WoE, Count
Imputer	Mean, Median, Most Frequent
Sampler	None, Random Undersampling, Random Oversampling
Scaler	None, MaxAbs, MinMax, Normaliser, Quantile
Feature Selector	None, K-Best

4.1.2 Preprocessing Affects Fairness

Pipelines that are not fairness-aware by design risk aggravating the fairness issues that are naturally present in the input data. If this is true, it motivates the need to address the problem of automatically configuring pipelines that are optimal with respect to a user-defined trade-off between fairness and ACC, for one of the fairness definitions given above.

The hypothesis that *fairness levels are affected by the choice of data preprocessing steps, as well as by their relative ordering in the pipelines* was experimentally tested by evaluating the effect of 20 different preprocessors (that is, single-operator pipelines) on ACC and on each of the three fairness definitions given above. Each experiment involved learning a binary classifier—LR was used due to its fast training times, but the method is classifier agnostic—from the well-known *Income* dataset, after preprocessing the raw data using a single preprocessor chosen from the options presented in Table 4.1. The *scikit-learn* [126] implementations were used for imputing, scaling and feature selection, while *Category Encoders* [113] was used for encoding, and *imbalanced-learn* [109] for sampling. Model ACC was estimated using 4-fold CV train/test splits, repeated 128 times from unique random seeds for robustness.

Kruskal-Wallis H tests were performed to determine whether specific preprocessing operator choices affect the analysed metrics. The tests, reported in Table 4.2, show that for most preprocessor family/metric pairs the null hypothesis that *the choice of a specific operator in a preprocessor family does not affect the resulting measure* may be rejected with 95% confidence, with the notable exception of imputation, for which the resulting p-values were not low enough, but still close to the 0.05 threshold for DPD and EOD. These results confirm that the choice of (most) specific operators affects not only the ACC, but also the fairness of the model trained on data that has been preprocessed using that operator.

Table 4.2 Kruskal-Wallis 95% p-values for operator-choice affecting each metric on *Income*; significant values are highlighted.

Task	DPD	EOpD	EOD	ACC
Encoding	**< .001**	**< .001**	**< .001**	**< .001**
Imputation	.081	.151	.055	.939
Sampling	**< .001**	.055	**< .001**	**< .001**
Feature Scaling	**< .001**	**< .001**	**< .001**	**< .001**
Feature Selection	.051	**.004**	**.001**	**< .001**

4.1.3 Problem Formulation

While an analysis similar to the one just presented has been seen before in the literature, cf. [165], FAIRPIPES' goal is to automatically generate fairness-aware pipelines, where the data scientist has control over the trade-off between fairness and ACC. This is formulated as a multi-objective optimisation problem, as follows: given a universe of configurable data processing operators (as in Table 4.1) and a target performance-fairness objective (for some choice of fairness definition), find an optimal sequence of configured operators, i.e. an optimal *pipeline*, with respect to the target.

For any decision problem involving two or more optimisation objectives, a point in the solution space (i.e., a specific configured pipeline) is said to be *Pareto-efficient* (or Pareto-optimal) if none of the individual objectives can be improved without worsening at least one of the other objectives. The set of all Pareto-efficient solutions is called the *Pareto front* [166]. In the case of fairness and performance pipeline optimisation, the aim is to compute the Pareto front consisting of all the pipelines such that both fairness and performance cannot improve at the same time.

A naive approach to addressing the problem is to consider each possible pipeline as an ordered combination of operators, learn a classifier for each of those, and calculate both its ACC and its fairness (note that this is a vector of values, one for each of the fairness metrics). This is a combinatorial problem, however. For example, there are 3,240 such pipelines in the experimental testbed, resulting from five operator families with varying number of options: six encoders, three imputers, five scalers, three resamplers, two feature selectors and six possible orderings. To address this complexity, a heuristic approach was taken. In Section 4.2 FAIRPIPES, a genetic algorithm that aims to optimise for both ACC and fairness, while also allowing for user-specified *policies* where these objectives can optionally be combined, is proposed. Given a choice of fairness metric, FAIRPIPES approximates the set of fairness/accuracy Pareto-efficient pipelines for that metric.

Contributions This chapter's contributions can be summarised as follows.

- Empirical evidence is presented on the effect of different data preprocessing operators over the fairness of the resulting classifier (Section 4.1.2).
- FAIRPIPES, a genetic algorithm producing fairness-aware preprocessing pipelines that are optimised for any combination of fairness and ACC, for three different exemplar definitions of fairness, is introduced. FAIRPIPES presents the data scientist with an estimate of the pipeline space's Pareto front, providing them with performance–fairness trade-offs.
- Preprocessing pipelines are encoded as individuals by turning the preprocessor options and the order of execution into a set of *genes*, which may mutate or be inherited by an individual's descendants across generations.
- FAIRPIPES is based on the standard genetic algorithm framework and, therefore, is applicable to any existing fairness measure and others that may become prominent in the future.
- Since FAIRPIPES relies on a given set of preprocessing operators, it is less invasive than dedicated preprocessing-fairness-correction methods, which consist of modifying the training data in a more invasive fashion, e.g. by resampling or relabelling the data. Therefore, FAIRPIPES may be used in situations where regulations prevent the modification of training data for automated-decision systems.
- An extensive experimental evaluation of the approach is presented, using four benchmark datasets: *Income*, *COMPAS*, *German* and *Titanic*, and a universe of 3,240 pipeline configurations. Using this testbed, it is shown that pipelines that are measurably close to the Pareto front for the chosen multi-objectives are discovered by exploring about 6% of the search space.

The experimental results, presented in Section 4.3, show that (i) fairness and performance stand in contrast with each other, as expected [90], and (ii) FAIRPIPES converges on pipelines that optimise for different objectives. In the experimental setting, evaluating the performance of 200 out of 3,240 possible pipelines—roughly 6% of them—led to estimated Pareto fronts with the average instance in the estimate less than 0.04 DPD/ACC units away from a true Pareto instance.

This is illustrated in the scatterplot of Figure 4.1, where pipelines are shown in the Fairness (DPD)/ACC space. Please note that $1 - \text{ACC}$ is used for consistency, so that

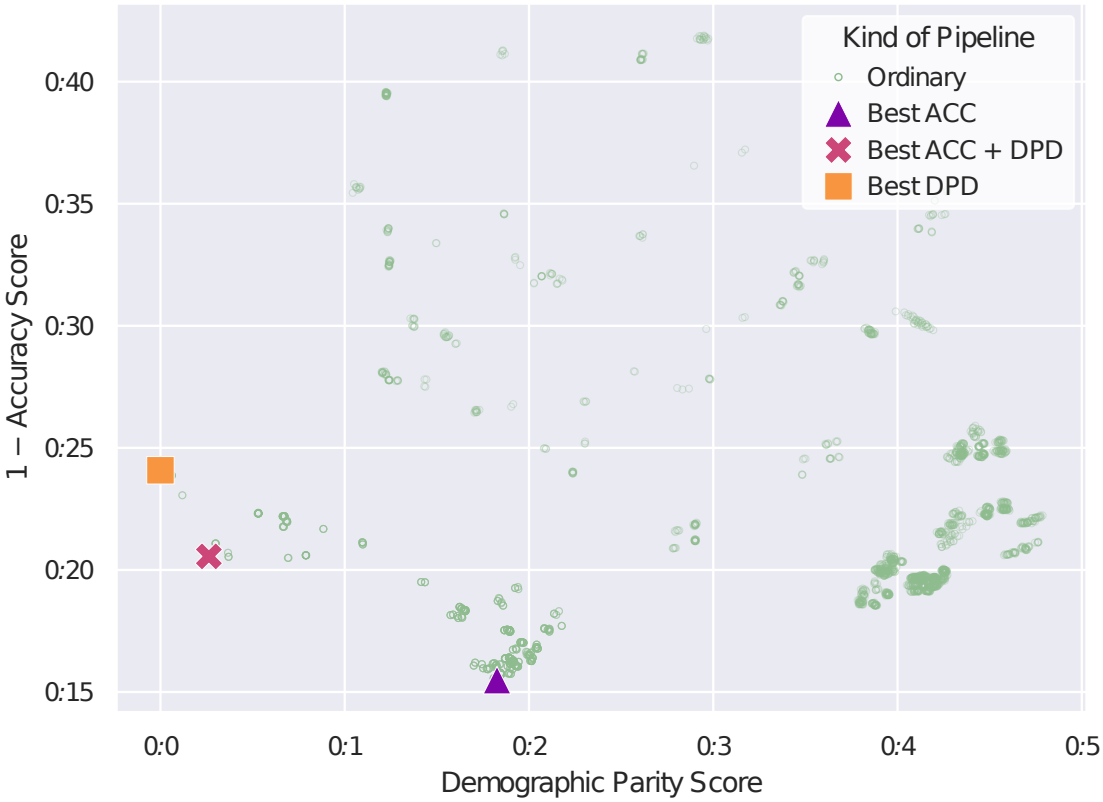


Figure 4.1 The three best pipelines optimised for ACC, DPD fairness, and the linear combination ACC + DPD all lie on the Pareto front, on the bottom left. Other non-optimal (*Ordinary*) pipelines are shown for comparison.

lower values are better for both fairness and ACC in this and every othe plot in the chapter. The "Ordinary" pipeline points in the plot are obtained by evaluating fairness and ACC on a selection of pipelines in the search space. As may be seen, only a few of those are on or close to the Pareto front in the bottom left corner. The three important points denoted Best ACC, Best DPD, and Best ACC + DPD represent the optimal Pareto front pipelines with respect to ACC only, DPD only, and for the sum ACC + DPD.

4.2 FAIRPIPES

FAIRPIPES performs a genetic-algorithm search [162] over the space of all preprocessing pipelines. In our experimental setting, pipelines are characterised by six *genes*, the first five representing a choice for each of the preprocessor options presented in Table 4.1, with the sixth one representing the order in which the operators are applied over the data; this preprocessor set was selected as a representative sample of they typical data preprocessing pipeline, but by no means is FAIRPIPES restricted to these, as it may easily be extended with any preprocessor that adheres to `scikit-learn`'s *fit-transform* paradigm.

FAIRPIPES has four tunable parameters:

- Generations (*n_gen*): number of FAIRPIPES iterations.
- Population size (*n*): number of pipelines per generation.
- Crossover rate (*c*): proportion of crossed-over pipelines in the next generation.
- Mutation rate (*m*): probability of a *gene* mutation.

FAIRPIPES can optimise for two families of objectives, or *policies*:

Change of Objective Taking advantage of the iterative nature of the genetic algorithm, users are allowed to control the *trajectory* leading from an initial set of seed pipelines, to a close approximation of the Pareto front. For instance, they may decide to optimise only for ACC for the first, say, ten generations, and then switch objective to DPD (or the other way around, or indeed one can specify different objectives for different phases of the iteration process). This makes it possible to reach specific areas in the Pareto front. For instance, a "ACC-then-DPD" policy with a 50% switch-over has experimentally shown to produce the effect of achieving high ACC and then converging towards a solution which has some level of fairness; in genetic-algorithm terminology, this is a particular case of a dynamic optimisation problem [15].

Linear Combination One can define linear combinations of fairness and performance metrics, e.g. $3 \times \text{DPD} + 7 \times \text{ACC}$, to convert the fairness/accuracy multi-objective problem into a single-objective problem. Note that *pure* objective metrics are a particular case of these linear combinations, e.g. $\text{DPD} = 1 \times \text{DPD} + 0 \times \text{ACC}$.

The FAIRPIPES algorithm is now presented using the example in Figure 4.2, with reference to the corresponding methods in Algorithms 4.1, 4.2 and 4.3.

Step (a) — Initialisation: To initialise the process, FAIRPIPES generates n random pipelines by choosing one option per gene for each pipeline (`GenPipes` in Algorithm 4.1).

Step (b) consists of two parts:

Evaluation: n copies of the raw dataset are separately processed through each of these pipelines, with the resulting processed datasets train/test split, binary classifiers, e.g. LR, are learnt from each of the training sets and the objective metrics are evaluated on the corresponding test sets (`GetMetrics` in Algorithm 4.2). This evaluation will be the main criterion that the algorithm aims to optimise.

Selection: The pipelines are ranked and sorted with respect to the objective. The best-ranking pipeline becomes the *elite*, i.e. it will survive for the following generation unmodified. The *elite* is kept in order to guarantee that the next generation will be at least as good as the current one with respect to its *best* individual. `Rank` in Algorithm 4.1 then assigns the i-th pipeline a probability of becoming "parent" of a pair of next generation "children" pipelines

$$\mathbb{P}_{\text{parent}}(i) = \frac{n+1-i}{\sum_{k=1}^{n} k} \quad \text{for } i \in \{1, \dots n\},$$

and a probability for the *non-elites* of "surviving" for the next generation of

$$\mathbb{P}_{\text{survive}}(i) = \frac{n+1-i}{\sum_{k=1}^{n-1} k} \quad \text{for } i \in \{2, \dots n\}.$$

This probabilistic approach was taken to add a randomness element to parent-selection, allowing the method to explore pipelines which might otherwise never be noticed.

Algorithm 4.3 consists of steps (c), (d), (e), (f) and (g).

Steps (c) and (d) — Crossover: are repeated $\lfloor c \cdot n \rfloor / 2$ times. Each time two parents are chosen without replacement with probability $\mathbb{P}_{\text{parent}}()$. One of the six genes, randomly selected, is swapped ("crossed over") between the two parents, and the resulting pipelines are appended to the *next_gen* list. The main reason for swapping over just one gene is to reduce the variability between parents and children, given that there are only six genes to modify. In standard genetic-algorithm terminology, this is a *two-point crossover* with consecutive crossover points.

Step (e) — Selection: $n - \lfloor c \cdot n \rfloor - 1$ different *non-elite* pipelines are chosen with probability $\mathbb{P}_{\text{survive}}()$ and appended to *next_gen*. This second part of the *selection* process again makes use of probabilities to allow for a small chance of additional exploration, at the expense of exploitation [11].

Step (f) — Mutation: Each *gene* of every *next_gen* pipeline may *mutate* with probability *m* into a different random option of the same kind, e.g. an encoder may mutate into another encoder, but not into an imputer. This allows for a small probability of more than one gene in any given pipeline.

Step (g) — Selection: The *elite* pipeline is appended to *next_gen* unmodified, completing the next generation. The *elite* is added at the end to prevent it from mutating.

This process is repeated from step (b) *n_gen* times, using the previous generation's *next_gen* instead of a random pipeline list to continue after the first generation. Throughout the experiments, $n\_gen = 20$ is set, as the number of pipelines to evaluate this way represents a substantial gain in computing time compared to an exhaustive search, while still providing good Pareto-optimal estimates.

4.3 Experimental Evaluation

The evaluation of the FAIRPIPES algorithm aims to show that close approximation to the Pareto front is achieved across a range of multi-objective optimisation policies, while exploring only a fraction of the entire search space.

As a baseline for the computational cost of using FAIRPIPES, running an exhaustive search on *Income*, i.e. evaluating all 3,240 pipelines over the dataset takes an average of 25 minutes on a *Microsoft Azure d64as_v4* VM with 64 vCPUs and 256 GB of RAM. In comparison, an average FAIRPIPES run evaluates 200 pipelines (20 ten-pipeline generations)—roughly 6% of the search space—in less than 1.5 minutes under the same

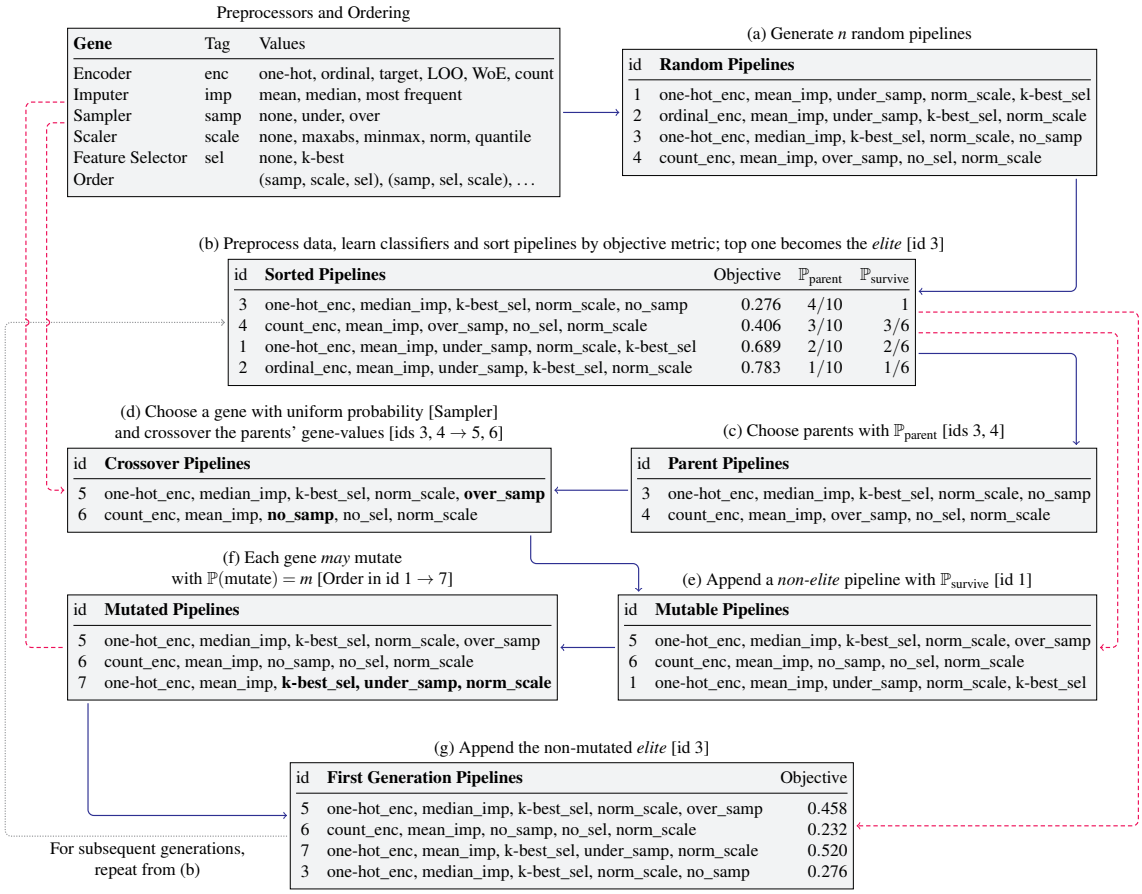


Figure 4.2 A FAIRPIPES run over *Income*, with population size $n = 4$, crossover rate $c = 0.5$, mutation rate $m = 0.4$ and objective $\text{DPD} + (1 - \text{ACC})$.

Algorithm 4.1: The FAIRPIPES algorithm.

input : D: dataset to process, with binary PA-and-label, pp_options: dict of preprocessors and task order,
n_gens: number of generations to run,
pop_size: number of pipelines per generation,
clf: classifier,
policy: optimisation strategy to follow,
co_rate: crossover rate,
mut_rate: mutation rate

output : pareto_front: Estimated front for the pp_options space

```
/* Generate pipe_pop, a pipeline list of size pop_size.  The pipelines are
   built by randomly choosing an element of each of pp_options:  encoder,
   imputer, feat_selector, sampler, scaler and permutation                  */
pipe_pop ← GenPipes(pp_options, pop_size);
all_metrics ← empty_df;          // Empty data frame to store pipeline metrics
for i ← 1 to n_gens do
    processed_dsets ← {pipeline(D) | pipeline ∈ pipe_pop};
    metrics_df ← GetMetrics(processed_dsets, clf);                  // Algorithm 4.2
    ranked ← Rank(pipe_pop, metrics_df, policy);
    pipe_pop ← GetNextGen(ranked, co_rate, mut_rate);               // Algorithm 4.3
    all_metrics ← Append(all_metrics, metrics_df);
end
pareto_front ← GetPareto(all_metrics);        // Locate non-dominated pipelines.
```

Algorithm 4.2: `GetMetrics` method.

input : processed_dsets: list of preprocessed datasets,
clf: classifier,
k: number of cross-validation folds

output : metrics_df: data frame of fairness and ACC metrics

```
metrics_df ← empty_df;
foreach D in processed_dsets do
    trains, tests ← KFoldSplit(D, k);
    metrics_list ← empty_list;
    for i ← 1 to k do
        clf ← Fit(clf, trains[i]);
        preds ← Predict(clf, tests[i]);
        metrics_fold ← GetFairnessPerformance(preds);
        metrics_list ← Append(metrics_list, metrics_fold);
    end
    metrics_average ← Average(metrics_list);
    metrics_df ← Append(metrics_df, metrics_average);
end
```

Algorithm 4.3: `GetNextGen` method.

```
input : ranked: ordered list of pipelines,
        co_rate: proportion of crossovers in next_gen,
        mut_rate: probability of a gene mutation
output : next_gen: pipeline list of length |ranked|
elite ← ranked[1];                                   // the best-ranked pipeline
next_gen ← empty_list;                               // stores next generation
n_child ← Round_to_Integer(co_rate ∗ |ranked|);      // round to nearest integer
while |next_gen| < n_child do
    {p1, p2} ← Parents(ranked); // select p1, p2 with rank-dependant probability
    {c1, c2} ← Crossover(p1, p2);        // select gene and swap values for p1, p2
    /* prevents duplicate pipelines                                          */
    if c1 and c2 not in next_gen then
        next_gen ← Append(next_gen, c1);
        next_gen ← Append(next_gen, c2);
    end
end
/* −1 kept for elite space                                                   */
while |next_gen| < |ranked| − 1 do
    s ← Survive(ranked \ {elite});    // select s with rank-dependant probability
    /* prevents duplicate pipelines                                          */
    if s not in next_gen then
        next_gen ← Append(next_gen, s);
    end
end
foreach pipe in next_gen do
    foreach gene in pipe do
        gene ← Mutate(gene, mut_rate); // modify gene with probability mut_rate
    end
end
next_gen ← Append(next_gen, elite);          // elite is kept for next generation
```

Table 4.3 Size and average FAIRPIPES run time for the analysed datasets.

Dataset	Attributes	Instances	Avg FAIRPIPES Run (*s*)	Per 100 Datum (*s*)
Income	14	32561	2472.12 ± 2462.97	0.5424
COMPAS	27	11038	539.93 ± 246.94	0.1812
German	22	1000	93.07 ± 23.43	0.4230
Titanic	10	891	44.48 ± 12.29	0.4992

configuration. A comparison of the average FAIRPIPES run time using a single Azure vCPU (no parallelisation) with 220 replicates per dataset over the four analysed datasets with default parameter values is presented in Table 4.3. As may be seen, the run times are dependant on the dataset's number of attributes and instances, but when these times are normalised per 100 datum, the run times are similar across datasets.

A more detailed analysis of the run time for all four datasets is presented in Figure 4.3, where the average values for 20 replicate runs per dataset and DPD importance (the quotient between the DPD and ACC coefficients in the objective linear combination). On *Income* and *Titanic*, smaller DPD importances caused longer FAIRPIPES run times, while for *COMPAS* and *German* the coefficients for DPD and ACC did not majorly affect the FAIRPIPES' average run times.

To analyse the effectiveness of FAIRPIPES, it is first shown to be effective at optimising either DPD or ACC, separately. Afterwards, the way in which DPD/ACC trade-offs are achieved incrementally through the generations is presented, using any of the objective policies described earlier.

All experiments are conducted over the space of pipelines obtained from all possible combinations of the preprocessors listed in Table 4.1, consisting of 3,240 data points. Although the most typically used preprocessors have been included, FAIRPIPES can be further extended to include additional tasks, as long as they comply with the fit/transform interface used by *sci-kit learn*. The pipelines always start by selecting an encoder followed by an imputer, as these steps are required for the rest of the preprocessing operators to work, and are completed by some ordering of the sampler, scaler and feature selector families.

This choice of experiment scale makes it tractable to compute the actual position of each data point relative to the real Pareto front (see here below), while providing sufficient evidence for the effectiveness and efficiency of the method. Please note that the space increases exponentially as the number of available operators is increased.

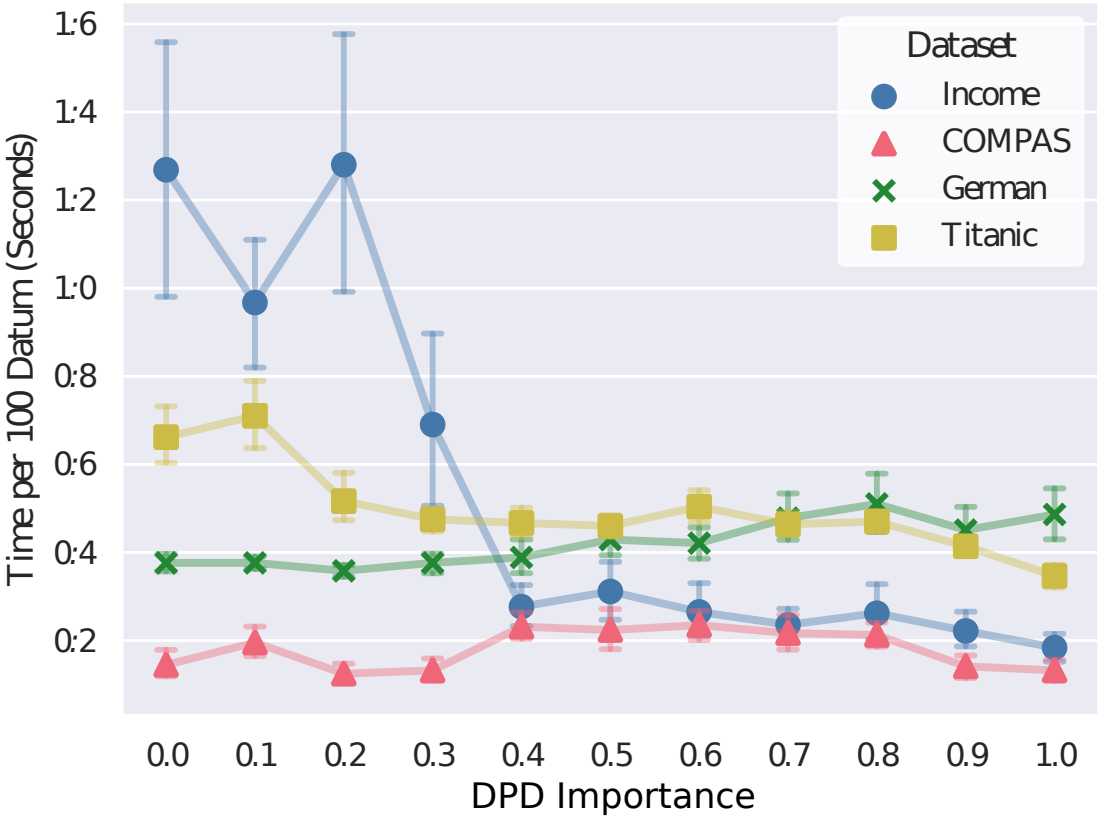


Figure 4.3 Average running time per 100 instances for all four benchmark datasets.

4.3.1 Baseline Mapping of the Search Space

First, the ACC and the fairness vector for the entire collection of pipelines in the search space was computed. This gives us the true Pareto front as a baseline to quantify the fidelity of the approximate solutions, as well as a computational effort baseline for exhaustive search. This mapping exercise was repeated for each of four benchmark datasets, *Income*, *COMPAS*, *German* and *Titanic*, described in Subsection 2.6.3. LR was used throughout as the reference binary classifier, owing to their fast training and ease of interpretability. Each training instance included 4-fold cross validation with a fixed random seed. Note however that FAIRPIPES is agnostic to the choice of classifier, as it is a preprocessing method.

The default crossover and mutation rates were selected based on recommended values in the literature [74] and were further fine-tuned by running FAIRPIPES 128 replicates, optimising for the objective value DPD + ACC. Figures 4.4 and 4.5 show the distance to the best possible objective value through 20 generations for the four benchmark datasets. To estimate the best defaults for the crossover and mutation rates, crossover was fixed while estimating mutation and vice-versa.

As may be seen in Figure 4.4, higher crossover rates yield a faster convergence towards the optimum objective value. The crossover rate values 0.6 and 0.8 had almost identical performances, and 0.6 was chosen as FAIRPIPES's default crossover rate.

When tuning the default mutation rate, higher mutation rates yielded a faster convergence to the optimum objective value, as may be seen in Figure 4.5, but the gains become negligible for values higher than the chosen default mutation rate of 0.4.

Fairness and ACC metrics were collected for each pipeline in the space and for each dataset, replicating each training session 64 times using different random seeds for robustness. The default parameters for FAIRPIPES were used in all the experiments: 0.6 crossover rate, 0.4 mutation rate, populations of 10 individuals, 20 generations per run, and 1-elitism, i.e. the best pipeline is kept unmodified from one generation to the next.

4.3.2 Single-Objective Optimisation

Setting each of the available target metrics as objectives, i.e., (DPD, EOpD and EOD) as well as to ACC, their values (averaged over all replicates as outlined above) were tracked over 20 generations of genetic evolution using FAIRPIPES. The plots in Figure 4.6, for each of the four datasets, show how each of the curves reaches a plateau, indicating that the empirical selection of 20 ten-pipeline generations is adequate to achieve stable results. To recap, in each plot lower metric values, including ACC (reported as $1 - ACC$) are better. In each plot, FAIRPIPES optimises for the metric on the y-axis, which are expected to be

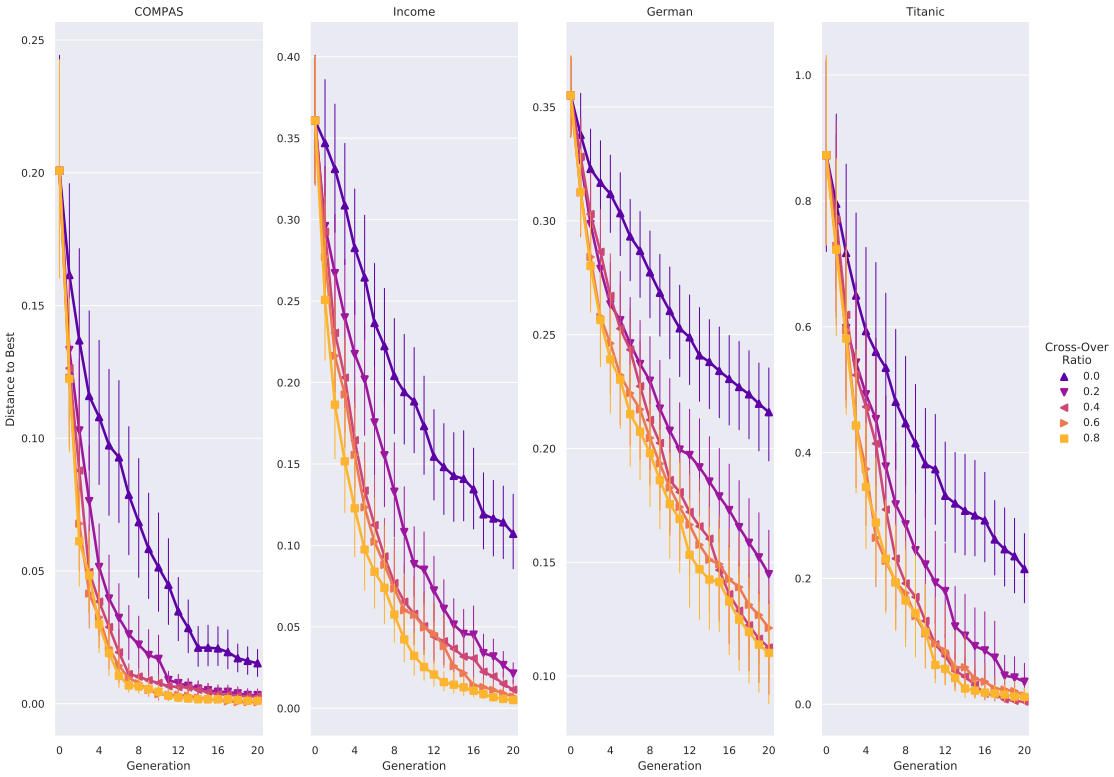


Figure 4.4 Average distance to best objective value per generation for different crossover rates across the four benchmark datasets. The results were averaged from 64 replicates, optimising for the objective value DPD + ACC.

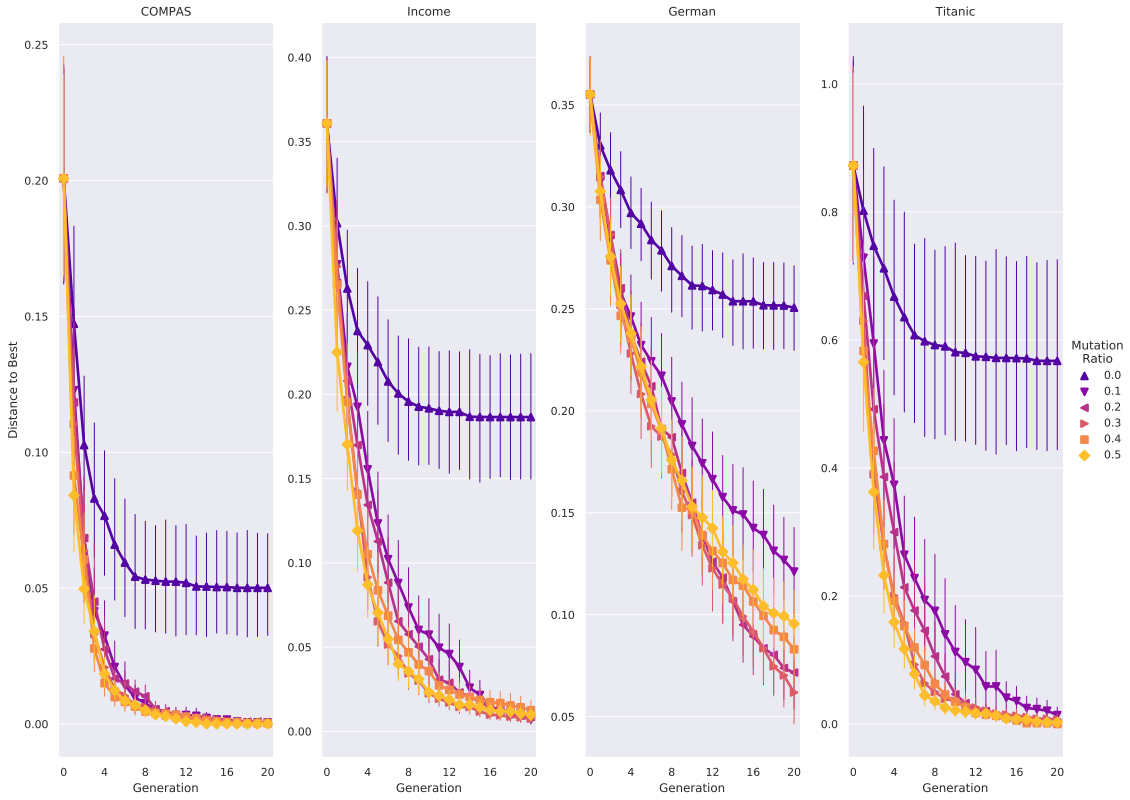


Figure 4.5 Average distance to best objective value per generation for different mutation rates across the four benchmark datasets. The results were averaged from 64 replicates, optimising for the objective value DPD + ACC.

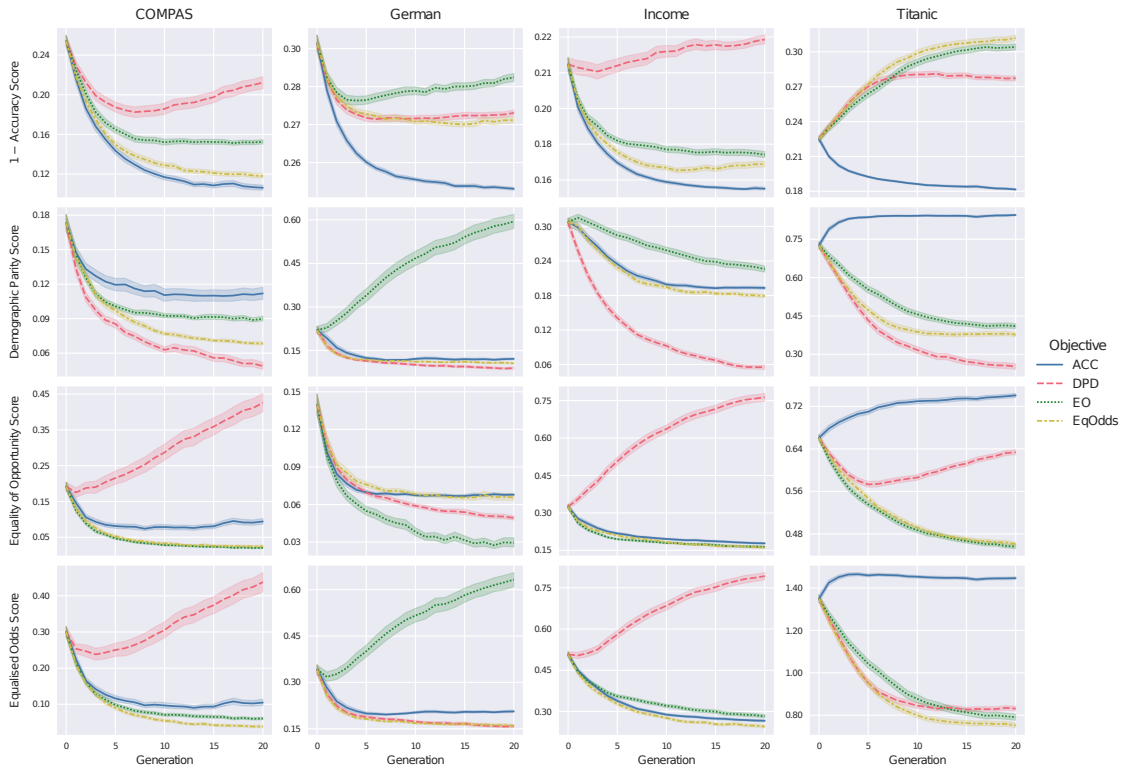


Figure 4.6 Evolution of all metrics as single objectives.

the lowest of the curves, while the other metrics are plotted for reference, and are not optimised for.

Indeed, some of the metrics run in the opposite direction, e.g. DPD on *COMPAS* and *Income*, EOpD on *German*, and ACC on *Titanic*. This behaviour seems to be dataset-specific, indicating that it would be difficult to identify general patterns for these metrics, without in fact running FAIRPIPES every time.

4.3.3 Multi-Objective Policies Optimisation

According to Justesen [88], "multi-objective problems are problems with two or more, usually conflicting, objectives. The main difference from single-objective optimization is that a multi-objective problem does not have one single optimal solution, but instead has a set of optimal solutions, where each represents a trade-off between objectives".

The two types of policy presented in Section 4.2 were considered:

1. Optimise for DPD for the first i generations and then change to ACC for the remaining $20-i$ generations, with i in $\{0,4,8,12,16,20\}$, producing six results for each benchmark dataset.

2. Optimise for a linear combination

$$k_{\text{DPD}} + k_{\text{ACC}} \text{ with } (k_{\text{DPD}}, k_{\text{ACC}}) \in \{(0,20),(4,16),\ldots,(20,0)\}.$$

In both cases, $(0,20)$ and $(20,0)$ purely optimise for ACC and DPD, respectively.

While both policy approaches are not true multi-objective optimisation (where the result is an approximation of the solution-space's Pareto front), the *explored pipelines*' Pareto front resulting from both policies is close to the ground-truth Pareto, and the policies are particularly good at finding the best pipeline with respect to the *user-selected* fairness/accuracy trade-off, as shown in Section 4.4. An alternative to both policy-approaches would have been to use an actual multi-optimisation algorithm, such as *NSGA-2* [39].

Figure 4.7 shows the evolution of average DPD and ACC through the generations, with the *change of objective* policy plotted on the left and the *linear combination* policy on the right, using *Income*. On the left, the changes of policy points are obvious, and they generally lead to very different endpoints for both ACC and DPD. These policies seem suitable for controlling the desired DPD/ACC trade-off.

Comparing each of the trajectories on the right with the corresponding ones on the left, it may be observed that the endpoints are fairly similar, however the curves on the right avoid the "spikes" that are apparent on the left. These spikes occur at the generation on which the policies change their objective, and before the spikes different objectives are indistinguishable from each other. Given this disadvantage of the change-of-objective policies and the similarity in the end-results between both approaches, the rest of the experiments focus exclusively on linear-combination policies, measuring the evolution of DPD/ACC across generations for different policies in Section 4.3.3 and the fidelity of the resulting estimates in Section 4.4.1.

DPD/ACC Evolution

Figure 4.8 shows a different perspective on how FAIRPIPES converges on its solutions, namely by presenting, for each of the benchmark datasets, how the solutions approach optimality (bottom left corner in the DPD/ACC space) for different linear combinations of the objective. Gradually varying the DPD and ACC coefficients (see legend) results in ordered and predictable paths: the higher DPD (correspondingly, the lower the ACC) coefficients result in fairer but less accurate averages across every generation on all datasets. Interestingly, for all datasets except *Titanic*, the initial random pipelines lie on the top right of the space, indicating poor fairness as well as poor ACC. However, the plots clearly show

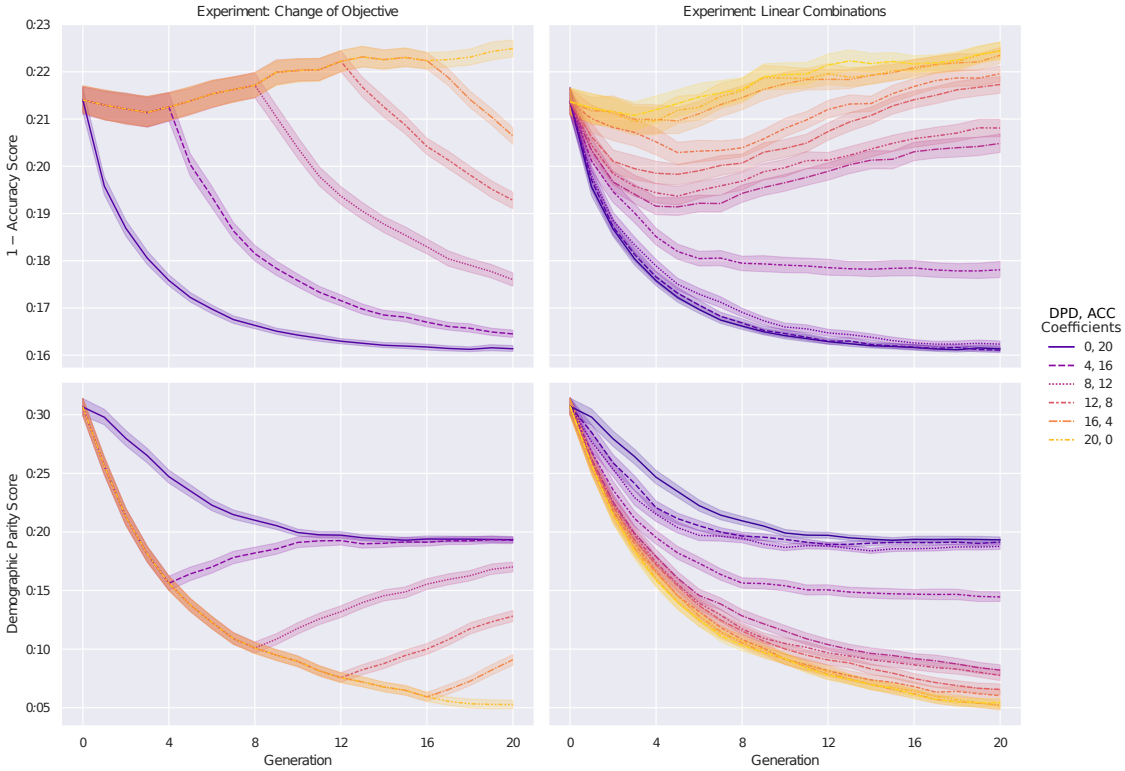


Figure 4.7 Comparison of metrics evolution on the *change of objective* and the *linear combinations* policies.

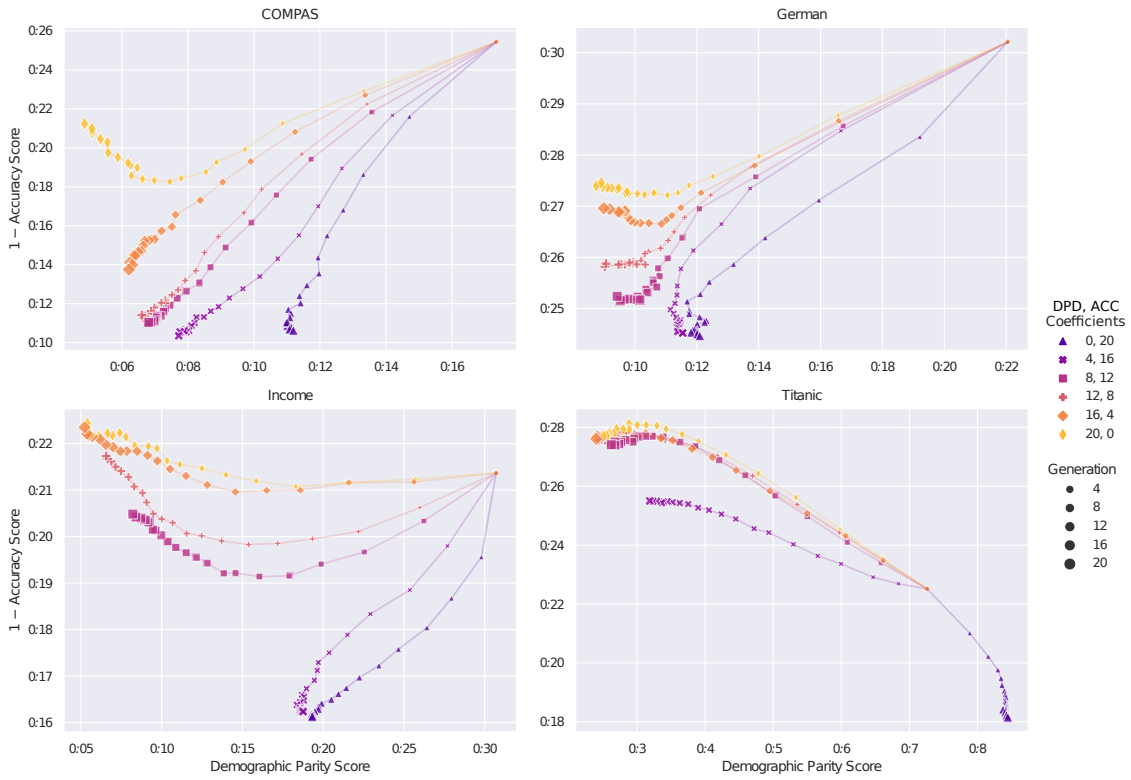


Figure 4.8 Average DPD and ACC per generation for different objective coefficients.

how FAIRPIPES achieves a combination of both, with the objectives not initially in contrast with one another. The extreme case of *Titanic* is peculiar, showing that DPD and ACC are in fact opposing objectives, i.e. the trajectories end in the top left/bottom right corners. This can be explained by considering the actual meaning and history of the dataset, as when sex is used as PA and survival as outcome, DPD and ACC are mutually exclusive: the "fairer" a model gets (both sexes have a similar death-rate), the less accurate it will become, as in reality most men died and most women lived.

Importance of processor ordering

Even though no specific preprocessing order was found to be best, once a specific set of operators is selected their order seems to matter. This is shown in Figure 4.9, where the pipelines belonging to at least one Pareto-estimate from the 128 replicates of the linear-combinations experiment are grouped by their preprocessing tuples (encoder, imputer, scaler, sampler and selector), regardless of the operators order. The number of different order-permutations present in the estimates for each of these unordered tuples is then counted. As may be seen, for each dataset, roughly 70% of the unordered tuples have one

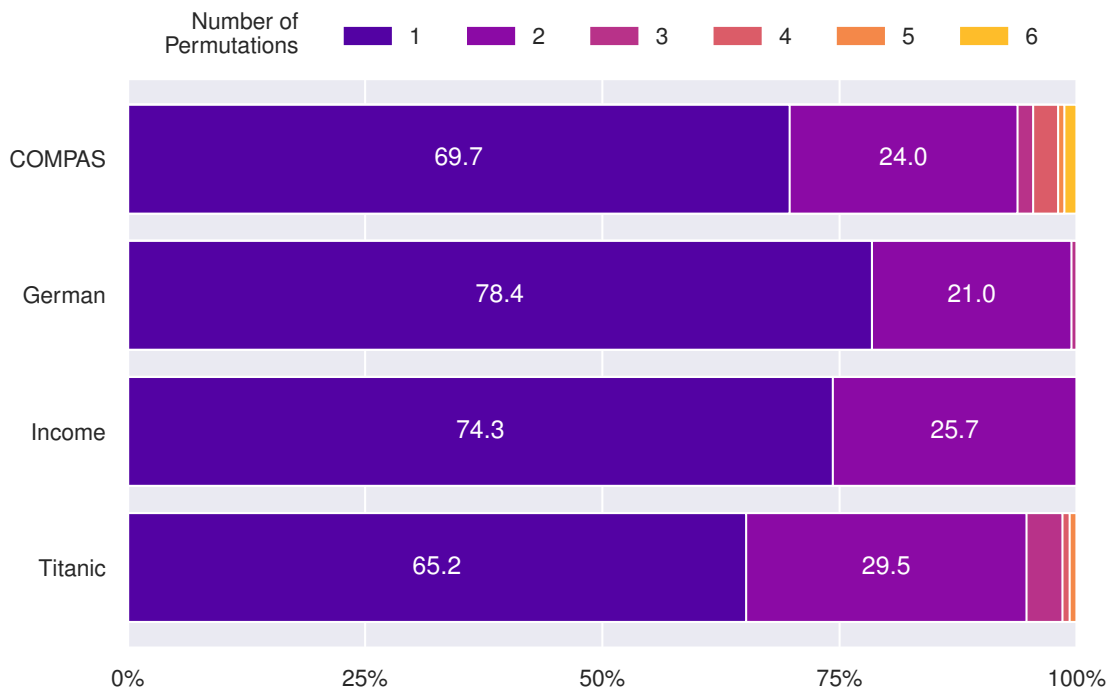


Figure 4.9 Proportions of the number of orderings present in any linear-combination experiment replicate for every Pareto-efficient unordered tuple.

ordering present in the estimated Pareto fronts, with most of the remaining tuples having two orderings. Specifically, for *German* and *Income* no tuple had more than two orderings present in their Pareto estimates, and only a small fraction of the *COMPAS* and *Titanic* tuples had more than two orderings present. This indicates that for *good* pipelines the task order is important, as otherwise more permutations would be expected.

No universally good solutions

Looking at Figure 4.10, it may be observed that the best pipelines are dataset-specific, i.e. given a dataset there will be better-performing preprocessor combinations, yet these are not the same across datasets, even when just a single preprocessor is considered.

The plot focuses on the encoders in the analysed pipelines. In general, different encoders achieve different DPD/ACC trade-offs. What is interesting, however, is that this is not uniform across the datasets. On *German*, for example, instances of *Target* produce the best DPD, while instances of WoE produce the best ACC, albeit with a poor DPD. In contrast, on *Titanic* it is *Count* instances that produce the best DPD, while instances of *One-Hot* produce the best ACC. Similarly, while some *LOO* produces relatively good DPD

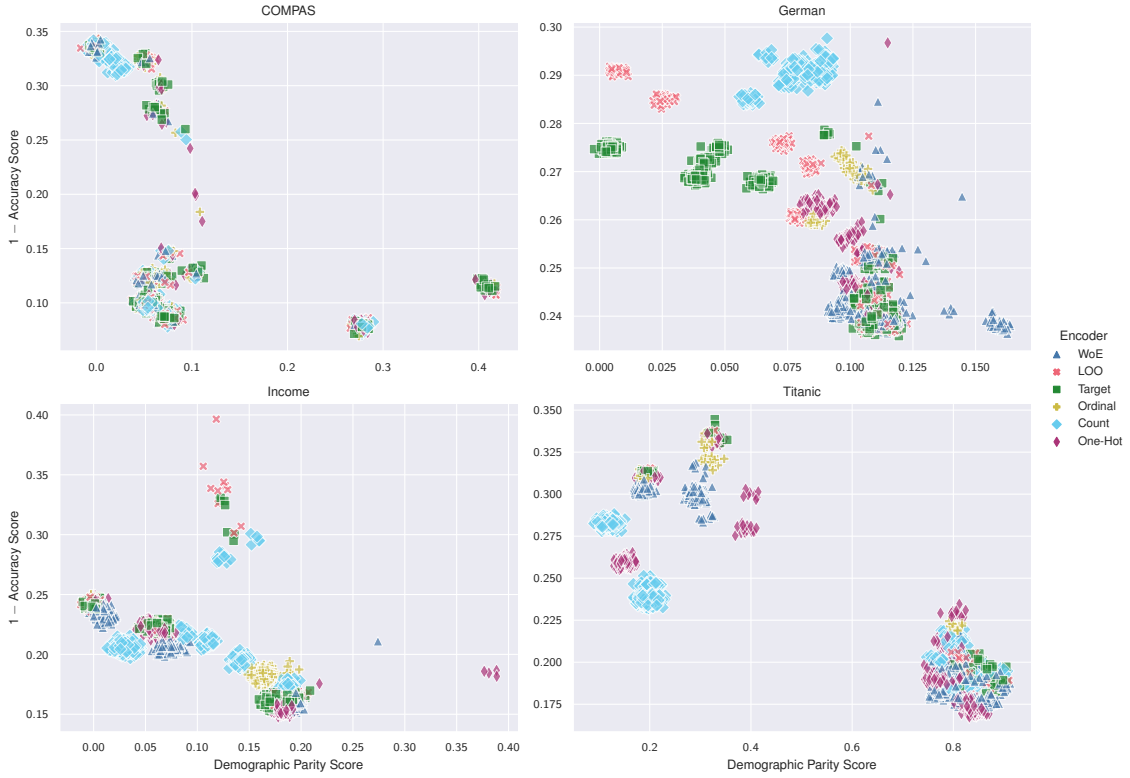


Figure 4.10 DPD/ACC trade-off for pipelines appearing in estimated Pareto fronts for different encoders. The patterns are not consistent across datasets.

values on *German*, this is not true on any of the other datasets. These observations are still true for the other preprocessor families (not shown due to space constraints).

It may be concluded that, while no single pipeline can be found that is uniformly optimal across datasets, FAIRPIPES is able to converge to near-optimal solutions for any of the benchmark datasets, without any prior knowledge of them.

4.4 Performance Evaluation

Finally, the convergence performance of FAIRPIPES was examined and compared against random pipeline searches consisting of the same number of explored pipelines. Two performance metrics were used: AHD, a global similarity measure which, in this chapter's setting, measures the similarity of the estimated and the true Pareto fronts, and B2B, a local similarity which compares the best pipeline found by FAIRPIPES against the best pipeline overall with respect to the specified linear-combination objective.

4.4.1 Pareto Front Estimation

Having established that solution trajectories end with trade-offs that attempt to optimise simultaneously fairness and ACC, next the quality of the solutions in terms of distance from the actual Pareto fronts was quantified, computed as described in Section 4.3.1 above. To measure the similarity of FAIRPIPES's estimated Pareto front with the true Pareto front, *averaged Hausdorff distance (AHD)*, a common performance measure in evolutionary multi-objective optimisation [141, 6] was used. AHD is a useful metric to estimate the quality of the Pareto estimate's coverage, i.e. it is a *global* metric; it is defined as

$$\mathrm{AHD}(X,Y) := \frac{1}{2}\left(\frac{1}{|X|}\sum_{x\in X}\min_{y\in Y} d(x,y) + \frac{1}{|Y|}\sum_{y\in Y}\min_{x\in X} d(x,y)\right),$$

where $d(x,y)$ is the Euclidean distance between x and y. For this chapter's comparisons, the estimated Pareto fronts are given the role of X and the true Pareto fronts are given the role of Y.

Figure 4.11 shows the mean AHD across replicates between the solutions generated by FAIRPIPES through the generations for the *Income* dataset, and the Pareto front, again across a range of linear combinations of the objectives. For each dataset, the estimates at the end of the generation process lie within 0.04 units of the true Pareto front, with estimates for *German* and *Income* achieving even better results. Interestingly, assigning a

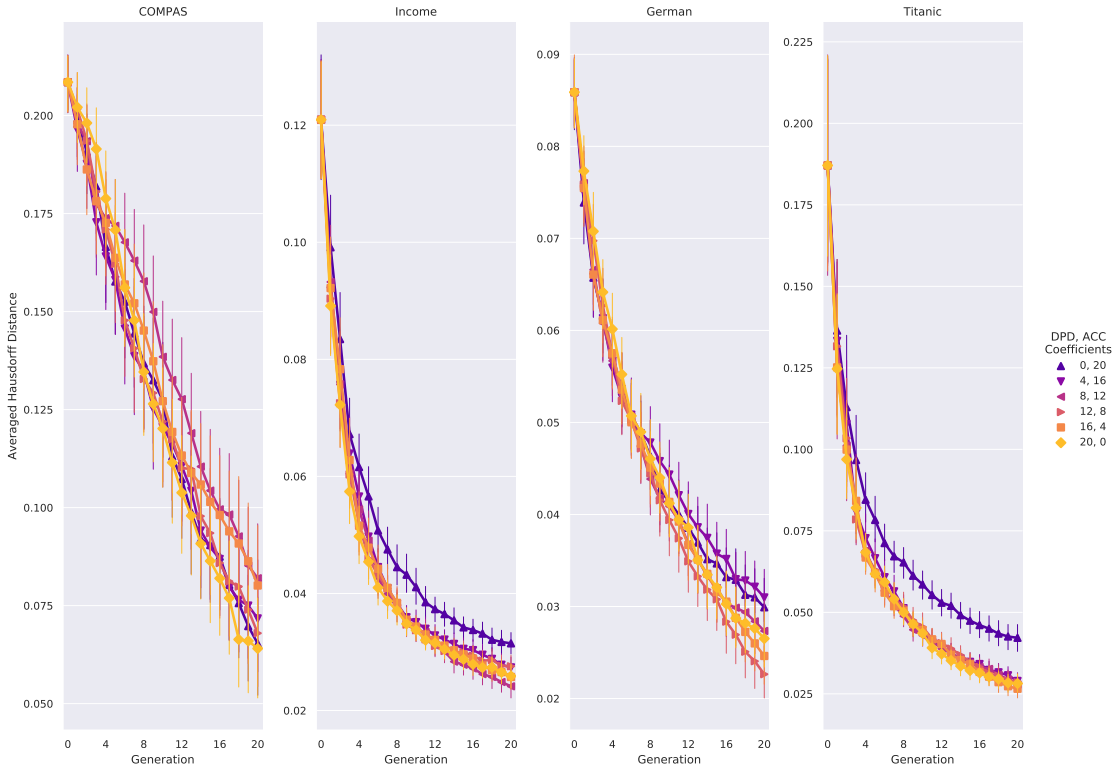


Figure 4.11 Averaged Hausdorff distance from estimated to true Pareto front across generations for different DPD/ACC linear combinations in *Income*.

positive coefficient to both ACC and DPD causes the average instance in the estimates to be closer to the Pareto front than estimates resulting from a pure-ACC policy.

4.4.2 Distance to Best Estimation

To evaluate the *local* estimation performance of FAIRPIPES, it is necessary to measure how close the best found pipeline is to the best possible pipeline given a user-selected objective value. This measure, denoted *best to best distance (B2B)*, was evaluated for different DPD and ACC linear combinations as follows:

$$\mathrm{B2B}(X,Y) := \min_{x\in X}\left[k_{\mathrm{DPD}}\mathrm{DPD}(x)+k_{\mathrm{ACC}}\mathrm{ACC}(x)\right]-\min_{y\in Y}\left[k_{\mathrm{DPD}}\mathrm{DPD}(y)+k_{\mathrm{ACC}}\mathrm{ACC}(y)\right],$$

where the estimated Pareto fronts are given the role of X and the true Pareto fronts are given the role of Y. These were measured and averaged over 128 replicates, and the resulting values are shown across 20 generations in Figure 4.12. For three of the benchmark datasets (*COMPAS*, *Income* and *Titanic*), the optimal pipeline was found within 20 generations.

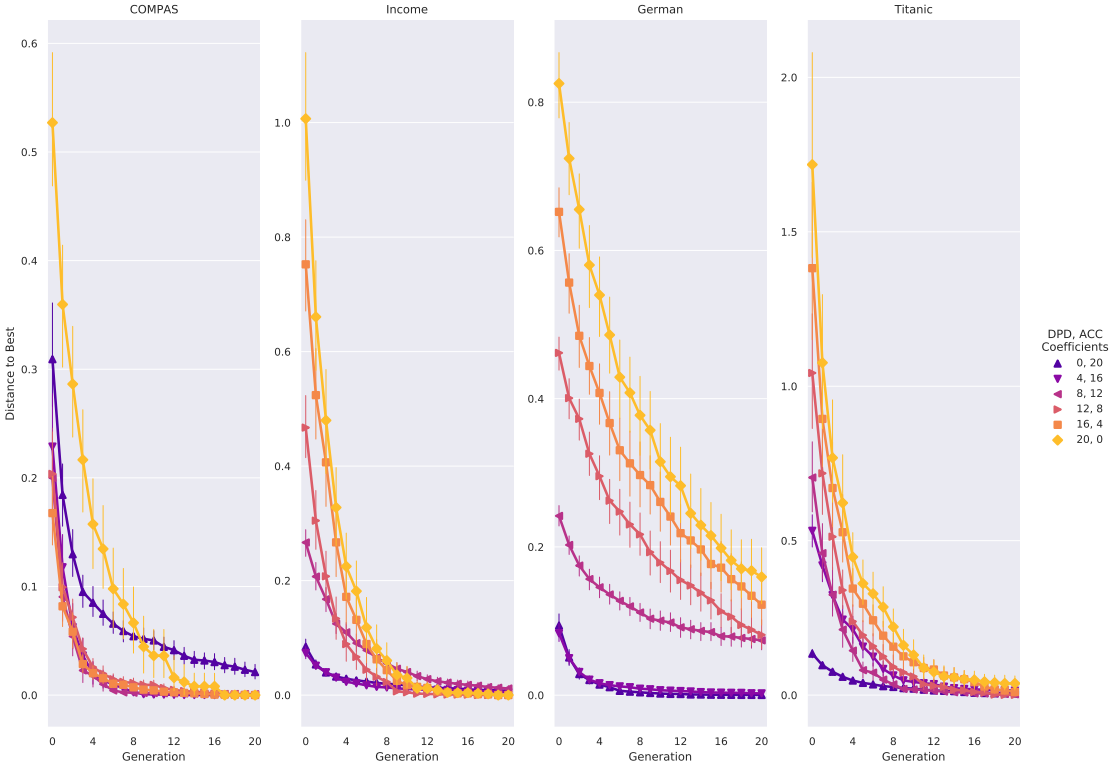


Figure 4.12 Average B2B distance from 128 replicates across 20 generations for different DPD/ACC coefficients for every benchmark dataset.

Table 4.4 Performance comparison between FAIRPIPES after 20 generations and a random pipeline sample without replacement of the same size (210 pipelines). For both metrics, lower values are better.

		FAIRPIPES		Random Sample		Two-Sample t-Test	
Metric	Dataset	Mean	SD	Mean	SD	t-value	p-value
AHD	COMPAS	**0.0049**	0.0050	0.0050	0.0048	−0.1666	.867
	German	**0.0053**	0.0032	0.0059	0.0019	−2.3357	**.019**
	Income	**0.0030**	0.0019	0.0032	0.0015	−0.8856	.376
	Titanic	0.0048	0.0038	**0.0042**	0.0023	1.8093	.071
B2B	COMPAS	**0.0021**	0.0132	0.0032	0.0154	−1.9686	**.049**
	German	**0.0699**	0.1319	0.1070	0.1513	−6.9388	**< .001**
	Income	**0.0063**	0.0204	0.0099	0.0210	−4.6448	**< .001**
	Titanic	**0.0121**	0.0596	0.0270	0.0977	−4.8908	**< .001**

4.4.3 Comparison with Random Sampling

To the best of the author's knowledge, there are no other existing pipeline-optimisation solutions that include fairness as an objective, either for single or for multi-objective optimisation. Therefore, as a baseline to compare FAIRPIPES's performance, the two defined similarity metrics were measured on random pipeline selections with a size equivalent to 20 FAIRPIPES generations, i.e. 210 pipelines. Eleven representative DPD/ACC linear combinations were used as objective values: $(0,10),(1,9),\ldots,(10,0)$, and the resulting metrics were averaged out for AHD and B2B. Each of these measurements was replicated and averaged 128 times for every benchmark dataset, and a two-sample t-test was performed using *SciPy*'s `ttest_ind` [159] to test for the null hypothesis that two independent samples—the FAIRPIPES and the random sample metric values—have the same expected value.

As may be seen in Table 4.4, FAIRPIPES outperformed random sampling in most cases for AHD, albeit not significantly. According to the performed t-test, the difference in performances was significant for every benchmark dataset in the case of B2B, where the average distance with FAIRPIPES was between 34% and 55% smaller than with random sampling. This indicates that FAIRPIPES did not only estimate the DPD / ACC Pareto front adequately, but estimated the optimal pipeline for the specified objective much better than random search did. The computing-time difference between running FAIRPIPES and performing the equivalent random search is negligible, as FAIRPIPES's genetic selection mechanism takes virtually no time to be computed.

4.5 Conclusion

This chapter presented FAIRPIPES, a genetic-algorithm approach for the discovery of data preprocessing pipelines that are near-Pareto-optimal with respect to both the fairness and performance of binary classifiers learnt from the data.

FAIRPIPES can optimise user-defined objective metrics defined through both linear combinations of fairness and ACC (for a variety of metrics), as well as through policies where the objective changes across the progress of the generations, presenting its users with estimates of the pipeline space's Pareto front, allowing them to select an adequate fairness/performance trade-off. Besides an adequate estimation of the Pareto front, FAIRPIPES significantly improved the estimation of the best pipeline for a given objective metric over an equivalent random pipeline search with an insignificant increase in computing time.

In further work, additional preprocessing operators may be introduced, as well as other types of classifiers, and higher-dimensional Pareto fronts may be explored, e.g. optimising for several fairness and performance metrics at once.

Chapter 5

Parametrised Data Sampling

This chapter is based on González-Zelaya et al. [65], presented at the 2021 *Extending Database Technology Conference (EDBT)*, which is a substantial extension of González-Zelaya et al. [63], presented at the *2019 KDD XAI Workshop*.

Contents

Summary

Improving machine learning models' fairness is an active research topic, with most approaches focusing on specific definitions of fairness. In contrast, PARDS is a parametrised data sampling method which can optimise the *fairness ratios* observed on a test set's predictions, in a way that is agnostic to both the specific fairness definitions and the chosen classifier. Given a training set with one binary protected attribute and a binary label, PARDS' approach involves correcting the positive rate for both the *favoured* and *unfavoured* groups through resampling of the training set. Experimental evidence showing that the amount of resampling can be optimised to achieve target fairness ratios for a specific training set and fairness definition is presented, while preserving most of the model's ACC. Conditions for the method to be viable are discussed, and then the method is extended to include multiple protected attributes. In the experiments three different sampling strategies were used, and results for three commonly used definitions of fairness, and three public benchmark datasets (*Income*, *COMPAS* and *German*) are presented.

5.1 Introduction

This chapter proposes PARDS, a fairness-definition and classifier agnostic resampling method, which may be easily implemented on top of existing ML solutions and can satisfy specific classifier requirements. PARDS is modulated through the continuous parameter d, which determines the amount of resampling introduced into the training data, and has three possible use cases: to find the optimal amount of correction for a specific fairness/classifier combination; to control a classifier's fairness/accuracy trade-off, and to align a classifier with a specific *world-view* [55], e.g. WYSIWYG, WAAE or AA, as defined in Section 2.6.1. Subsection 5.2.2 describes how parameter d can be mapped to align to these world-views.

Standard data preparation techniques may be used to correct the fairness behaviour of a classifier [138]. PARDS is based on data resampling, which is well understood and part of the typical data management pipeline [96]. Being a preprocessing operator, PARDS may easily be added after data cleaning into existing database solutions. Like other resampling techniques, PARDS can be computationally inexpensive and yield reduced classifier learning times, as shown in Subsection 5.4.2. PARDS offers the versatility of using both generic (random undersampling, random oversampling and *SMOTE* [24]) and fairness-specific (preferential sampling [89]) methods.

A common resampling problem is the loss of predictive ACC caused by such interventions [12]. In this chapter's setting, such loss can also be controlled through parameter

d, allowing for a decision in the amount of fairness/accuracy trade-off the user is willing to accept. Furthermore, the experiments in Section 5.4 show that even at high correction levels, the ACC loss for PARDS is relatively low.

Contributions PARDS, a parametrised fairness-correcting resampling method, is introduced. PARDS is fairness-definition and classifier agnostic, and achieves close to optimal fairness correction with a small loss in predictive performance. Extensive experiments to benchmark the effectiveness of the method using the *Income*, *COMPAS* and *German* datasets are presented, and the method's implementation is available as a collection of *Jupyter Notebooks*, linked and described in Appendix A.3.

This chapter's additional contributions are four-fold:

1. A mathematical formulation of resampling-based fairness correction is provided, through an analysis of conditions for its viability and effectiveness with respect to the linear separability of the training set, with theoretical results and experimental evidence on synthetic datasets.
2. The optimal fairness correction is estimated using Bayesian optimisation.
3. An initial investigation into extending the method to multiple protected attributes is made.
4. PARDS is benchmarked and compared with several existing fairness-correction methods.

5.1.1 Fairness Ratios

Slight variations of the fairness definitions presented in Chapter 3 now follow. A classifier's *demographic parity ratio (DPR)*, *equality of opportunity ratio (EOpR)* and *proxy fairness ratio (PFR)* are defined as:

Definition 5 (Fairness Ratios)**.**

$$\text{DPR} := \frac{\mathbb{P}(\hat{Y} = 1 \mid \text{PA} = U)}{\mathbb{P}(\hat{Y} = 1 \mid \text{PA} = F)},$$

$$\text{EOpR} := \frac{\mathbb{P}(\hat{Y} = 1 \mid \text{PA} = U, Y = 1)}{\mathbb{P}(\hat{Y} = 1 \mid \text{PA} = F, Y = 1)},$$

$$\text{PFR} := \frac{\mathbb{P}(\hat{Y} = 1 \mid do(\text{PA} = U))}{\mathbb{P}(\hat{Y} = 1 \mid do(\text{PA} = F))}.$$

For DPR and EOpR, we evaluate the ratio of the positive classification probabilities for U and F. PFR is computed by intervening on the test set T twice, assigning every individual in T the PA-values U and F, resulting in $T_{\text{PA}=U}$ and $T_{\text{PA}=F}$, respectively, and then evaluate the quotient of the intervened sets' classification *PRs*; in all cases, the ratios quantify how close the classifier comes to optimal fairness, with ratios closer to 1 indicating a "fairer" model.

5.2 PARDS

The initial focus is on datasets with both binary protected attributes and labels. The plots in this section result from applying PARDS to the *Income* dataset.

5.2.1 Correction Parameter

The *disparity correction* parameter $d \in [-1,1]$ may be used for three different objectives:

- To enforce a particular *world-view* [55], as defined above.
- To modulate a classifier's fairness/accuracy trade-off.
- To optimise a classifier with respect to a fairness definition.

This chapter's main objective will be the third one, to estimate the d-value optimising a classifier's predictions with respect to a fairness definition. The method is summarised as follows:

1. Define PR-correcting functions for F and U.
2. Select a *sampling strategy* to correct the training set.
3. Estimate the fairness-specific optimal d-value.

Details on each of these steps now follow.

5.2.2 Parametrising Correction

The first step is to define linear functions that will yield corrected PRs for both PA groups. These functions, which we will call $f^+(d)$ and $u^+(d)$, should satisfy the constraints: $f^+(1) = \text{PR}(F), f^+(-1) = \text{PR}(U)$ and $u^+(d) = f^+(-d)$.

The equations for these two linear functions are

$$f^+(d) = md + b, \quad u^+(d) = -md + b,$$

with coefficients

$$m = \frac{\mathrm{PR}(F) - \mathrm{PR}(U)}{2}, \quad b = \frac{\mathrm{PR}(F) + \mathrm{PR}(U)}{2}.$$

Having defined $f^+(d)$ and $u^+(d)$, it may be seen that parameter d maps to the three world-views introduced earlier, as follows:

- $d = 1$ is associated with the WYSIWYG world-view, with no effect on the training set.
- $d = 0$ is associated with WAAE, making the PRs for both the favoured and the unfavoured groups equal with the population PR.
- $d = -1$ is associated with AA, as it makes the PR of the favoured group equal with the unfavoured group's original PR and vice versa.

5.2.3 Sampling Strategies

In the second step, the resulting corrected ratios $f^+(d)$ and $u^+(d)$ are used to produce a resampled training set $\{\hat{U}, \hat{F}\}$ satisfying these ratios. The required amount of resampling for F and U will depend on d and the selected strategy.

PARDS can use one of four different sampling methods, modified to work on specific PA-label subgroups: random undersampling (*Under*), random oversampling (*Over*), *SMOTE* [24] and preferential sampling (*PS*) [89]. Depending on the sampling method, the following subgroups will be modified:

***Under*:** Undersample F^+ and U^-.

***Over*:** Oversample F^- and U^+.

***SMOTE*:** Oversample F^- and U^+.

***PS*:** Undersample F^+ and U^-, oversample F^- and U^+.

An intuition on these transformations may be visualised in Figure 5.1, where the area of the squares represents the relative sizes of the four subgroups F^+, F^-, U^+ and U^-.

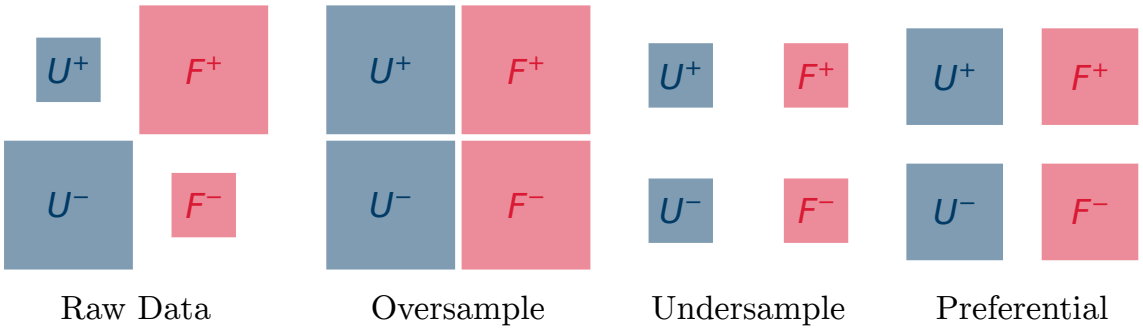


Figure 5.1 Intuition of the different sampling strategies to equalise F's and U's positive ratios. The area of the squares indicates the size of the respective subgroups.

Let $F*$ and $U*$ be the resampled favoured and unfavoured groups, respectively. $F*$ must satisfy

$$\frac{|F*^+|}{|F*^+| + |F*^-|} = f^+(d),$$

which may be rewritten as

$$\frac{|F*^+|}{|F*^-|} = \frac{f^+(d)}{1 - f^+(d)}. \tag{5.1}$$

The selected strategy will determine whether F^+ or F^- will be resampled to satisfy (5.1). Using *Under*, for example, $F*^+$ results from undersampling F^+, while $F*^- = F^-$. In contrast, using *Over* produces $F*^-$ from oversampling F^- while $F*^+ = F^+$. An analogous equation to (5.1) is used to resample U onto $U*$.

After the training-set has been resampled, a classifier learnt from the corrected training-set will display an improvement in fairness with respect to a classifier learnt from the original data. An example of the produced PR-correcting functions and their effect over *Income* is shown in Figure 5.2. Algorithm 5.1 specifies the full details of PARDS.

5.2.4 Finding the Optimal Amount of Sampling

Finally, the third step is to estimate the optimal correction for a specific fairness definition. As classifiers usually display non-linear—and sometimes unexpected—behaviours, it is not possible to deduce a closed-form solution to this optimisation problem. Hence, it becomes necessary to numerically approximate a solution.

A naïve approach is to compare the resulting fairness ratios on a training-set split for different values of d, and select the one producing the ratio closest to 1. As shown in Section 5.4.2, it is easy to find d-values close to the optimal by trial and error, yet this optimal d-value will usually be different for distinct fairness definitions.

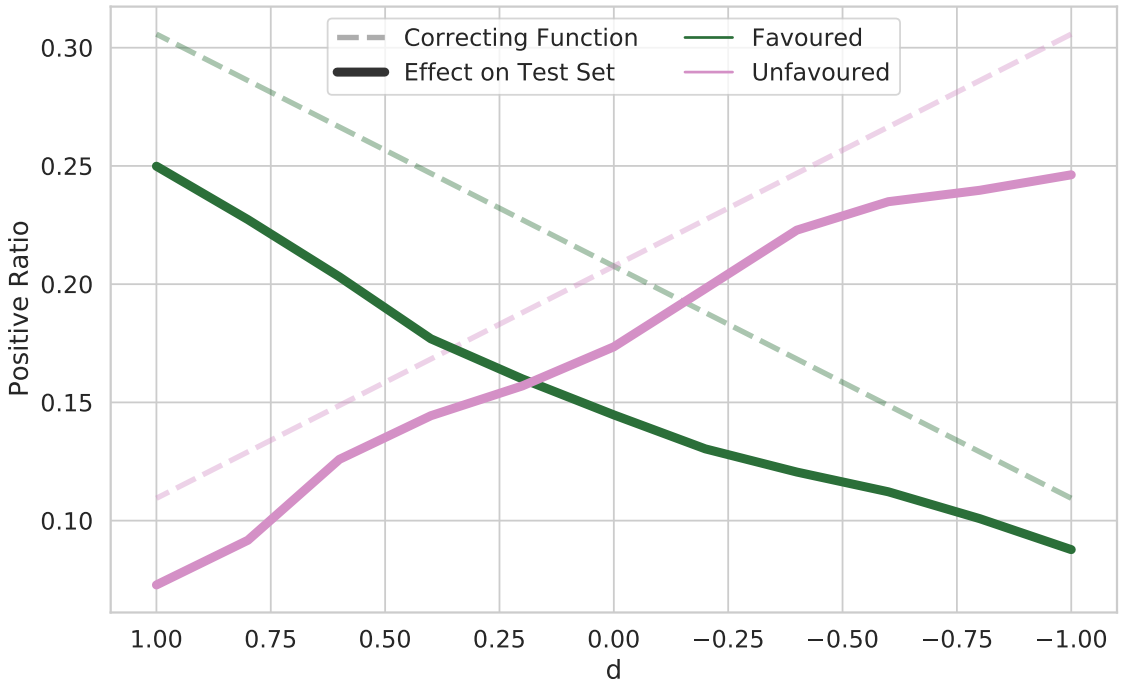


Figure 5.2 Correcting functions $f^+(d)$ and $u^+(d)$ applied to *Income* and their effect on the test set. The d-axis is reversed, going from 1 (no correction) to -1 (maximum correction). Note that the test-set PRs do not necessarily intersect at $d = 0$, as the attribute distributions will be different in the resampled training set and in the test set.

Algorithm 5.1: The PARDS algorithm.

Data:
T: a training set with binary PA and binary label Y,
$d \in \mathbb{R}$: the correction parameter,
$s \in \{$*Under, Over, SMOTE*$\}$: the sampling method.
Result:
Tcorr: a d-corrected training set.

```
   /* Get favoured F and unfavoured U datasets                    */
1  for i in {0, 1} do
2  |  T_i = T[PA == i] ;
3  |  TP_i = T[(Y == 1) and (PA == i)] ;
4  |  PR_i = size(TP_i) / size(T_i) ;
5  end
6  j = argmax_{i∈{0,1}}(PR_i) ;
7  F = T_j ;
8  U = T – F ;
   /* Get positive rates for U, F and T                           */
9  for p in {U, F, T} do
10 |  PR_p = size(p[Y == 1]) / size(p) ;
11 end
   /* Correcting polynomials                                      */
12 m = (PR_F – PR_U) / 2 ;
13 b = (PR_F + PR_U) / 2 ;
14 fpr = m * d + b ;
15 upr = –m * d + b ;
   /* Split F and U into Positive and Negative                    */
16 [fPos, fNeg] = [FY == 1, FY == 0] ;
17 [uPos, uNeg] = [UY == 1, UY == 0] ;
   /* Under case                                                  */
18 if s == under then
   |  /* Correct fPos and uNeg groups                            */
19 |  f_k = fpr / (1 – fpr) ;
20 |  u_k = (1 – upr) / upr ;
21 |  fPosSize = f_k * size(fNeg) ;
22 |  uNegSize = u_k * size(uPos) ;
23 |  fPos = undersample(fPos, size=fPosSize) ;
24 |  uNeg = undersample(uNeg, size=uNegSize) ;
   /* Over and SMOTE cases                                        */
25 else
   |  /* Correct fNeg and uPos groups                            */
26 |  f_k = (1 - fpr) / fpr ;
27 |  u_k = upr / (1 - upr) ;
28 |  fNegSize = f_k * size(fPos) ;
29 |  uPosSize = u_k * size(uNeg) ;
30 |  fNeg = oversample(fNeg, size=fNegSize) ;
31 |  uPos = oversample(uPos, size=uPosSize) ;
32 end
   /* Get all four groups back together                           */
33 Tcorr = concat(fPos, fNeg, uPos, uNeg) ;
```

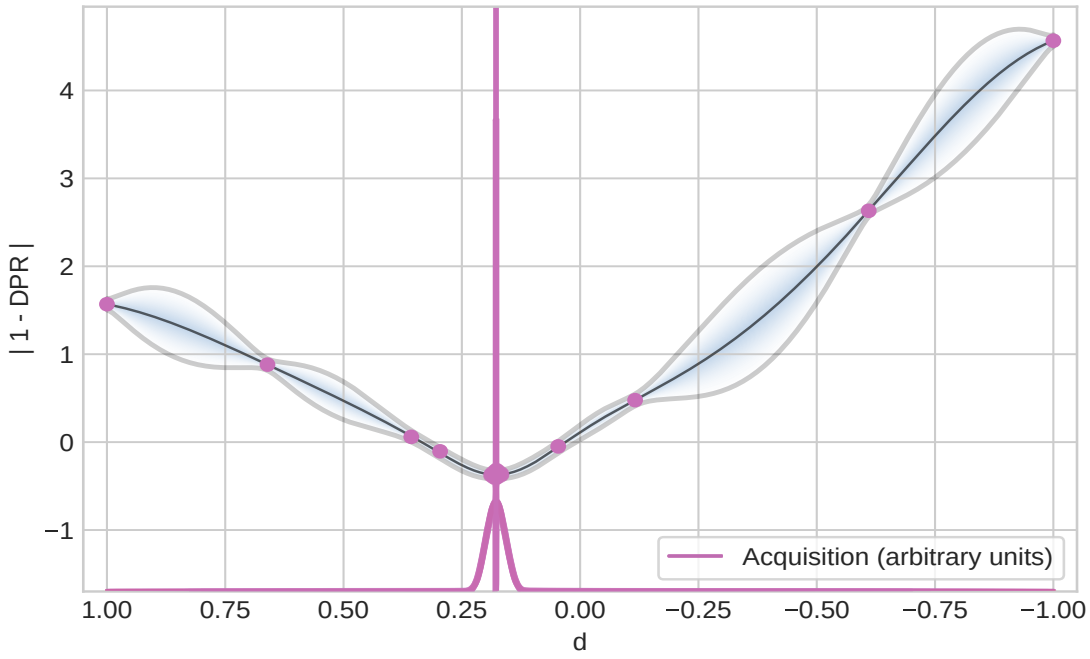


Figure 5.3 Plot of *GPyOpt*'s approximation of DPR as function of d for *Income*. The bottom red curve displays the resulting distribution for the optimal d-value.

A more systematic way to approximate the optimal value of d is to use Bayesian optimisation [61]. This technique estimates the objective function on candidate values obtained from previous function estimations. The main reasons for choosing Bayesian over other optimisation methods are that every fairness ratio evaluation will be different due to the randomness in the sampling process and that Bayesian optimisation is good when estimating the objective function is expensive, e.g. this chapter's setting, since large datasets are used, and the final results are averaged over many estimations.

A simple fairness optimiser was implemented using the *GPyOpt* [151] package, with a standard Gaussian process using d as the only parameter and the distances of the different fairness ratios to 1 as objective functions, e.g. estimate the d-value yielding the fairest DPR expectation, evaluated on a 90/10 split of the training set:

$$\arg\min_{d} |1 - \mathbb{E}(\mathrm{DPR}(d))| \quad \text{subject to} \quad -1 \leq d \leq 1.$$

An example run, used to approximate the optimal correction for DPR on *Income* training-sets may be seen in Figure 5.3.

5.2.5 Alternative Methods

PARDS' is not the only possible approach to fairness correction. Three other options are briefly discussed here.

Independent Subgroup Resampling

An alternative to using d would be to control the amount of resampling introduced into each of the four PA-label subgroups separately, either as an absolute number or as a proportion of the corresponding subgroup. This might improve the predictive performance of the corrected classifier, while still achieving optimal fairness. However, there are two drawbacks to this approach: 1. Additional parameters in the system imply a loss in interpretability about the introduced amount of applied correction, as the intuition described above would be lost; 2. Having to optimise four independent parameters instead of one necessarily implies an increased computational cost.

Data Relabelling

Instead of selectively resampling data, it is possible to selectively *relabel* data, i.e. systematically change the class label for selected datapoints [90]. This approach, although highly effective in correcting fairness, displays a marginally worse performance than resampling on both synthetic and real data. A comparison of relabelling and resampling, both using PARDS to determine the desired PRs may be seen in Figure 5.4. Besides this, there is an argument against using this method in the fact that it is highly invasive, since relabelling implies altering the ground truth, while resampling is based on real data to train from. However, relabelling may be an effective strategy when additional privacy constraints need to be enforced [135].

Weighted PAs

Another possibility for fairness correction would be to assign weights to datapoints based on their PA and label. For certain classifiers, this would indeed have an effect on fairness. However, not all classifiers can incorporate weights during learning [9].

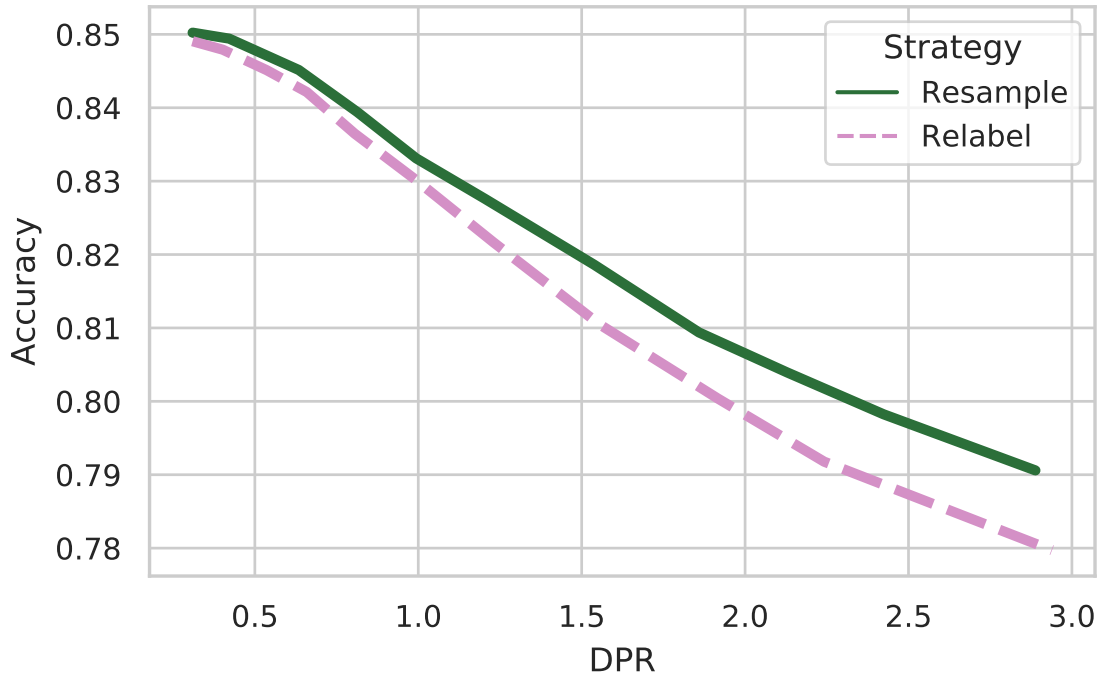


Figure 5.4 Comparison of resampling and relabelling on *Income*. The fairness/accuracy trade-off is consistently worse for the latter.

5.3 Theoretical Results

PARDS' fairness correction over linear classifiers works best on data with low linear separability. Its correcting effect softens as data has greater linear separability, and stops working altogether on linearly separable data, as will be shown for the single-predictive-feature case next.

5.3.1 Method Effectiveness

Consider a training set $D \subset X$ and test set $T \subset X$ with binary labels $Y(x) \in \{+,-\}$ for $x \in D$. The positive and negative instances of D will be referred to as

$$D^+ = \{x \in D \mid Y(x) = +\} \text{ and } D^- = \{x \in D \mid Y(x) = -\}, \text{ respectively.}$$

The resulting dataset of resampling D with PARDS is referred to as $D^* \subset X$. A classifier $\hat{Y}$'s *false negative, false positive, true negative* and *true positive* predictions with respect to D are denoted FN, FP, TN and TP, respectively. When necessary, subscripts are added for additional context.

Definition 6 (Linear Separability)**.** Let D be a dataset with label Y and $\mathscr{L}$ be a linear classifier. The *linear separability* of D with respect to $\mathscr{L}$ is defined as the maximum ACC attainable on D by some $\hat{Y} \in \mathscr{L}$, i.e.

$$s_{\mathscr{L}}(D) \coloneqq \max_{\hat{Y} \in \mathscr{L}} \frac{|TN_{\hat{Y}}| + |TP_{\hat{Y}}|}{|D|}.$$

If $s_{\mathscr{L}}(D) = 1$, D will be said to be *linearly separable*.

For the rest of this section, $\hat{Y} \in \mathscr{L}$ is assumed to be learnt from D and *optimal*, i.e. its ACC on D is maximum. An intuition for the way PARDS works in a univariate classification problem is presented, followed by the formalisation of this intuition.

The effects of PARDS' resampling on fairness in the univariate case are exemplified in Figure 5.5, where training elements $x \in D$ are represented by circles and shown with their *true* labels, while test element $t \in T$ is represented by a square and shown with its *predicted* label. In the univariate case, a linear classifier $\hat{Y}$'s *decision threshold* will be the value $b \in \mathbb{R}$ such that

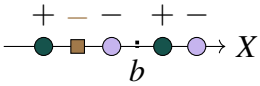
$$\hat{Y}(x) = \begin{cases} + & \text{if } x \geq b, \\ - & \text{otherwise.} \end{cases}$$

A useful observation is that for all $x \in FN$, there must be more elements of TN than elements of FN between x and b. This is formalised in Definition 7 and Lemma 1, and exemplified in Figure 5.6.

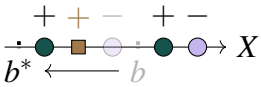
Definition 7 (Right Difference)**.** Let $x \in FN$. x's *right false negatives* $rFN(x)$, *right true negatives* $rTN(x)$ and *right difference* $\Delta r(x)$ are defined as:

$$rFN(x) \coloneqq \left\{ x' \in FN \mid x \leq x' < b \right\},$$
$$rTN(x) \coloneqq \left\{ x' \in TN \mid x \leq x' < b \right\},$$
$$\Delta r(x) \coloneqq |rTN(x)| - |rFN(x)|.$$

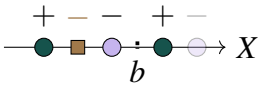
Lemma 1. *Let $\hat{Y} \in \mathscr{L}$. Then $\Delta r(x) > 0$ for all $x \in FN$.*



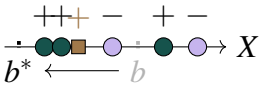
(a) In one dimension b is an optimal decision threshold.



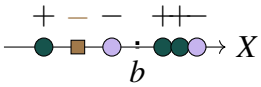
(b) Undersampling left of b.



(c) Undersampling right of b.



(d) Oversampling left of b.



(e) Oversampling right of b.

Figure 5.5 The effect of undersampling D^- or oversampling D^+ on the predictions of $t \in T$ (represented by the square). Labelled "−" in (a), resampling *left* of b, as shown in (b) and (d), causes t to be labelled "+"; resampling *right* of b, as shown in (c) and (e), has no effect on t's predicted label.

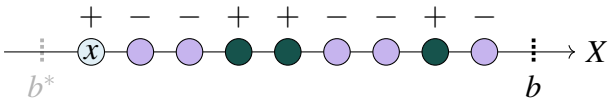


Figure 5.6 The right difference of $x \in FN$ must be positive, i.e. there must be more elements in $rTN(x)$ (shown in pink, labelled "−") than in $rFN(x)$ (shown in green, labelled "+"), as otherwise $\hat{Y}^*$, associated with $b^* \leq x$, would predict Y more accurately than $\hat{Y}$, associated with b, assumed to be optimal.

Proof. Let $x' \in FN$. If $rFN(x') \geq rTN(x')$, using x' as the decision threshold would produce fewer misclassifications over D than the original H, which was assumed to be optimal. □

Definition 8 (Induced Positive Ratio)**.** The positive ratio of T induced by D is defined as:

$$\mathrm{PR}_D(T) \coloneqq \frac{|\{t \in T \mid t \geq b\}|}{|T|}.$$

The following theorem shows the extent to which undersampling $D^- \subset D$ can modify the predicted label for $t \in T$ as long as $FN \neq \varnothing$ (and hence $s_{\mathscr{L}}(D) < 1$).

Theorem 1. *Let $\hat{Y} \in \mathscr{L}$. If $FN \neq \varnothing$, define*

$$n_D \coloneqq \frac{1}{|FN|} \sum_{x \in FN} \Delta r(x) \quad \textit{and} \quad b^* \coloneqq \frac{1}{|FN|} \sum_{x \in FN} x,$$

the average right difference and average value of $x \in FN$, respectively. Undersampling $\frac{|D^-|}{|TN|} n_D$ elements from D^- will result in a corrected training set D^* with an expected increase in $PR_{D^*}(T)$ of

$$PR_{D^*}(T) - PR_D(T) = \frac{|\{t \in T \mid b^* \leq t < b\}|}{|T|}. \tag{5.2}$$

Proof. Let $\hat{Y} \in \mathscr{L}$ learnt from D, and $x \in D^-$. Removing $x \in FP$ will not increase the number of instances misclassified by H, and $\hat{Y}$ will remain optimal with respect to $D \setminus \{x\}$. Thus, only undersampled $x \in TN$ may prevent $\hat{Y}$ from being optimal.

Since FN is not undersampled by ParDS, removing an average of n_D elements from TN will prevent Lemma 1's condition from holding for some $x^* \in FN$. As $TN \subset D^-$, in order to remove n_D elements from TN, an average of $\frac{|D^-|}{|TN|} n_D$ need to be removed from D^-.

Hence, $\hat{Y}^* \in \mathscr{L}$ with decision threshold b^* will exist. Any $t \in T \cap [b^*, b)$, labelled "$-$" by $\hat{Y}$, will be labelled "$+$" by $\hat{Y}^*$; this implies (5.2). □

Analogous results to Theorem 1 show that oversampling D^+ will *increase* $\mathrm{PR}_{D^*}(T)$, and undersampling D^+ or oversampling D^- will *decrease* $\mathrm{PR}_{D^*}(T)$, as long as $s_{\mathscr{L}}(D) < 1$;

the proofs are similar to Theorem 1's and hence omitted. These results show that undersampling or oversampling the adequate subgroups will improve the expected DPR by increasing $\mathrm{PR}_{D_U^*}(T_U)$ and decreasing $\mathrm{PR}_{D_F^*}(T_F)$, as long as D is not linearly separable.

5.3.2 PR Gain Estimation

The PR increase in Equation (5.2) may be estimated by the empirical distibution of D as follows, assuming that D and T were drawn from the same distribution:

$$\frac{|\{t \in T \mid b^* \leq t < b\}|}{|T|} \approx \frac{|\{x \in D \mid b^* \leq x < b\}|}{|D|}. \tag{5.3}$$

To exemplify (5.3), the PR increase per undersampled instance was estimated on training sets with different levels of separability. Figure 5.7 shows the theoretical estimations along with the actual increases in PR. For this experiment, ten synthetic training sets $\{D_1, \cdots, D_{10}\}$ with one continuous variable were generated by sampling 500 "$-$" class instances from $N(0,1)$ and 500 "$+$" instances from $\{N(z_{0.5-0.05i}, 1) \mid i \in [1..10]\}$, respectively. For each D_i, the decision boundary was $b_i = z_{0.5-0.05i}$, and $s_{\mathscr{L}}(D_i) = 0.1i$. Random undersamplings of every D_i^- were performed 100 times, averaging out both the estimated and actual PR gains. As may be seen, the expected increase in PR diminishes as the training sets have greater separability. In the extreme case of perfect separability, PARDS stops working altogether. However, introducing random noise lets correction become effective again, as explained in Subsection 5.4.1 and shown in Figure 5.9b.

5.3.3 Multiple Protected Attributes

PARDS has been generalised to multi-class PAs, as well as to multiple PA variables. In some cases, this could be addressed by binning several PA labels into just two categories, U and F. However, these arbitrary assignments would imply a loss of granularity in any subsequent fairness analysis. As an alternative, considering a *combined* PA has been chosen, which may be obtained for every datapoint $p \in$ train as follows:

1. Evaluate $\mathrm{PR}(D)$ for the training set D.
2. Define a set of PAs: $\{\mathrm{PA}_1, \mathrm{PA}_2, \ldots, \mathrm{PA}_k\}$.
3. Evaluate
$$\mathrm{PR}_i(p) = \mathrm{PR}\left(\mathrm{PA}_i(p)\right) - \mathrm{PR}(D)$$

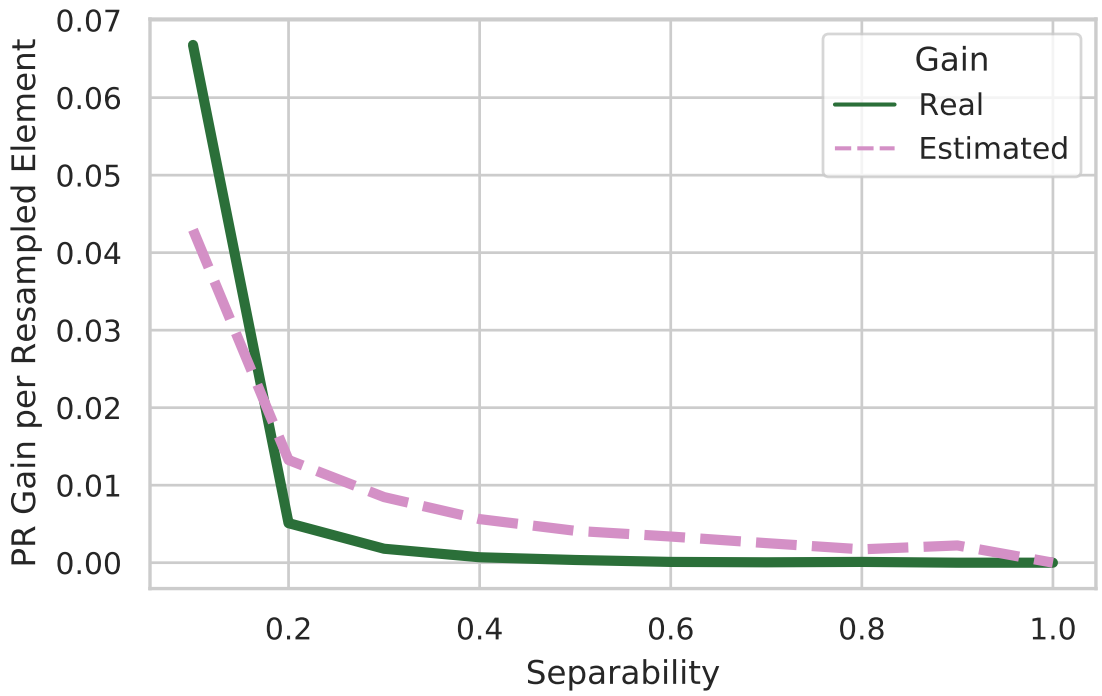


Figure 5.7 Expected vs actual gain in PR(T) per undersampled instance for training sets with different degrees of separability.

for $i \in \{1, 2, \ldots, k\}$.

4. Aggregate the partial PRs to obtain a combined value

$$\mathrm{PR}^*(p) = \sum_{i=1}^{k} \mathrm{PR}_i(p).$$

5. Define the combined PA of p as

$$\mathrm{PA}^*(p) = \begin{cases} F & \text{if } \mathrm{PR}^*(p) > 0, \\ U & \text{if } \mathrm{PR}^*(p) \leq 0. \end{cases}$$

This solution allows for a much more granular approach on determining a datapoint's relative "prosperity" with respect to every PA, as some PA attributes may prove to be more determining of disparate treatment than others, and the effects of several PAs may cancel each other out.

The performed experiments, carried out on *Income*, provide positive results, as described next. Figure 5.8 compares the effects—for unfavoured groups of different PAs (Gender, Nationality, Race and Age)—of applying disparity correction based on a single PA (Gender) to doing it based on a combined PA aggregating Gender, Age, Race and Country for *Income*. As can be seen, when correcting for Gender alone, the other unfavoured groups' PR remains relatively constant or gets worse. Likewise, the overall PR shows a drop on its PR as more correction is applied. When correcting for the combined PA, on the other hand, all of the unfavoured PRs improve at a similar rate, whilst the overall PR remains relatively constant across different correction levels. In short, this extension provides PARDS with the capability to correct for multiple biases simultaneously, at the individual level with similar optimal d-values across PAs. This method for combining several PAs into a single combined one, though, is not unique, and could further be improved by adding weights to the different PAs set as hyperparameters by experts.

5.4 Experimental Evaluation

This section reports the effectiveness of PARDS regarding separability in Subsection 5.4.1, comparing sampling strategies and fairness definitions in Subsection 5.4.2 and benchmarking PARDS with existing fairness-correcting methods, in Subsection 5.4.3.

5.4.1 Separability

To verify the effect of separability on PARDS' effectiveness, 11 simple datasets were created, consisting of one continuous feature f and one binary label l. These datasets were created using the *scikit-learn*'s [142] *make_classification* function, with varying levels of class separability s, ranging from 0 (completely mixed up) to 2 (over 95% probability of complete separation). What this function does is sample feature values from normal distributions centered at s and $-s$ for the two classes, respectively. A PA was then randomly added, ensuring a fixed $50/50$ proportion of F vs U datapoints, with PRs of 0.9 and 0.1 for F and U, respectively.

As may be seen in Figure 5.9a, the greater the separability that data has, the less effective PARDS becomes (represented by a near-flat DPR curve as a function of d), verifying the results of Section 5.3. However, adding random noise to a linearly separable dataset (effectively rendering it inseparable again) restores the effectiveness of PARDS. To test this, a linearly separable dataset with $s = 2$ was created, and noise gradually introduced through parameter n taking values from 0 to 1, the proportion of randomly-assigned labels.

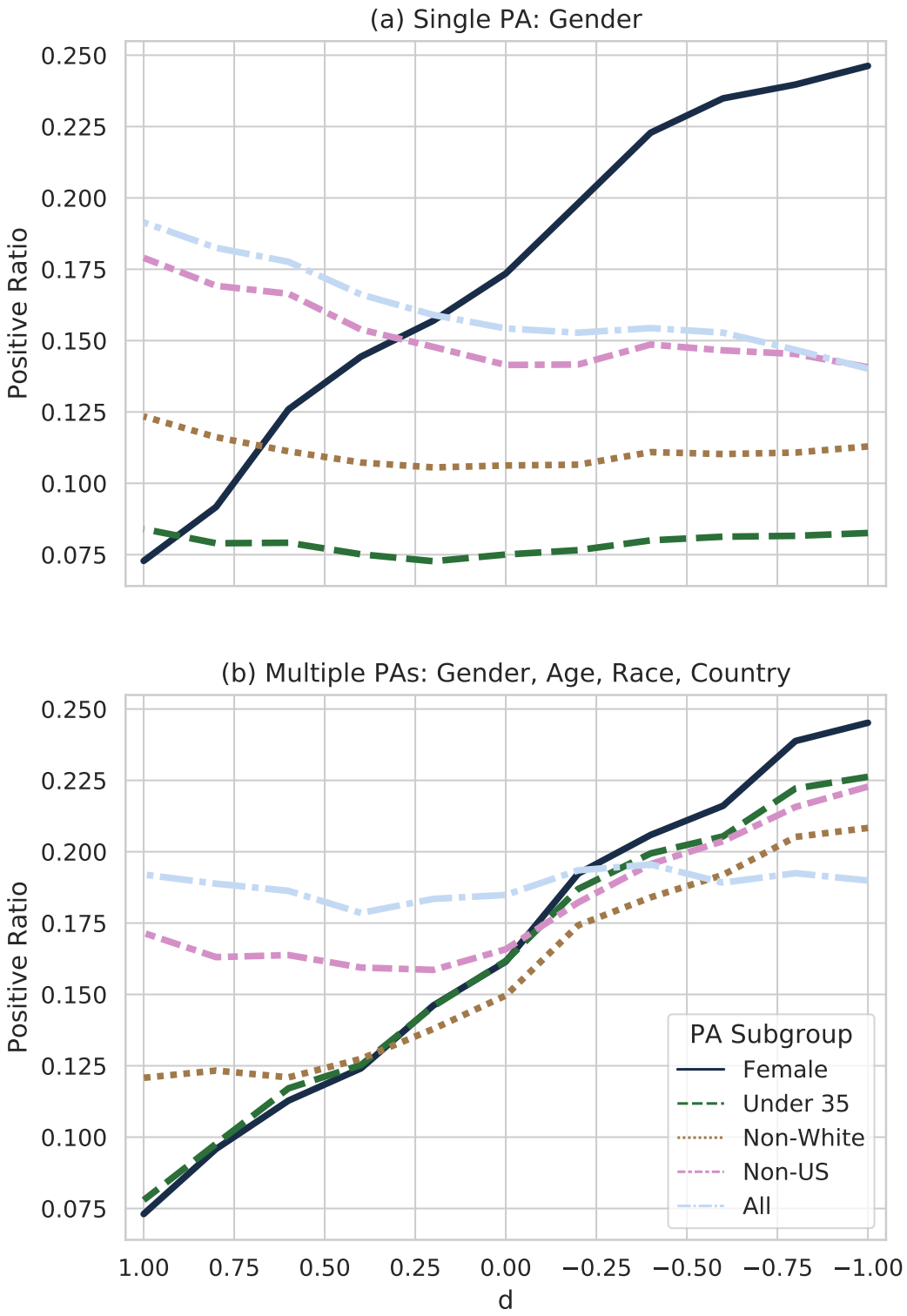


Figure 5.8 Positive ratios for the different PAs' unfavoured groups on *Income*, correcting for (a) one PA and (b) multiple PAs.

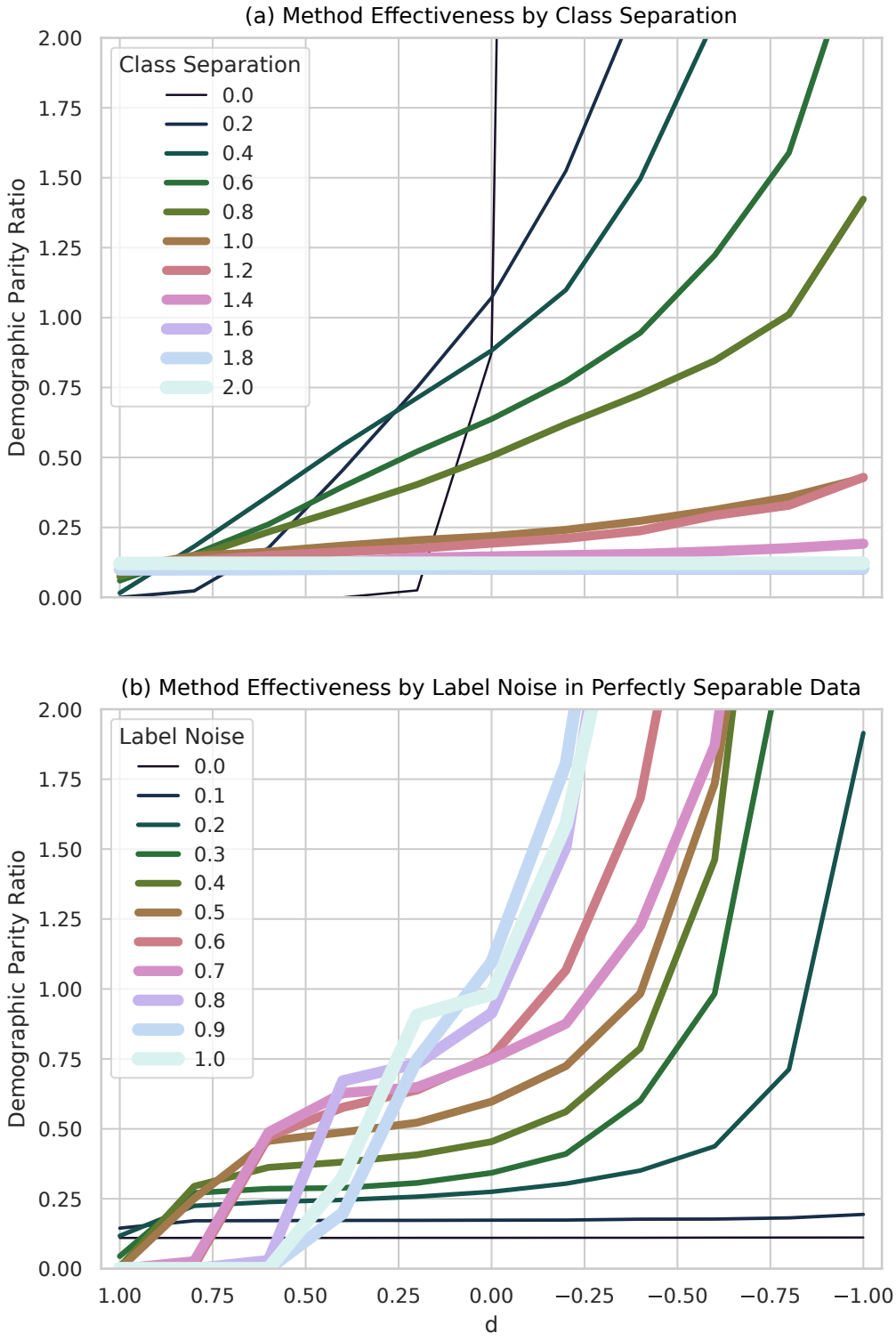


Figure 5.9 Correction effectiveness by separability. **(a)**: On close-to-linearly separable data, the method becomes highly inefficient, or even stops working at all. **(b)**: Introducing random noise into a separable dataset lets correction become effective again.

As shown in Figure 5.9b, this intervention can render fairness correction effective again, even with a small amount of added noise.

5.4.2 Method Validation

PARDS was tested on three benchmark datasets, described in Subsection 2.6.3: *Income*, *COMPAS* and *German*. For every dataset, the following experiment was performed 50 times, and then averaged the results for robustness:

1. Random train/test split the data with $90/10$ proportion.
2. For each sampling method get 11 training sets corresponding to $d \in \{1, 0.8, \ldots, -1\}$.
3. For each of these training sets, fit a classifier.
4. Get predictions for T, $T_{\text{PA}=U}$ and $T_{\text{PA}=F}$ for every model.
5. Compute metrics for ACC, DPR, EOpR and PFR, as well as the model coefficients.

An analysis of the resulting fairness metrics, and a comparison of these results with *PS* now follows.

Results

As expected, disparity correction has an effect on a classifier's PR. Figure 5.2 shows a particular instance of this, using *Under* as the correcting method on *Income*. As may be seen, at the optimal d-value—the point at which the curves cross over—both the F and U groups achieve the same PR, which is also the population PR.

Figure 5.10 shows the resulting metrics for ACC, DPR, EOpR and PFR for all three analysed datasets. It is worth noting that both the *German*—and to a lesser extent *COMPAS*—datasets have a relatively smaller number of instances, explaining why the trends for those datasets are not as smooth as *Income*'s. In all three datasets, ACC decreased monotonically with correction increases, as may be seen in the top row of Figure 5.10. However, the drop in ACC resulting from increased correction is not particularly severe, with less than a 4% ACC drop on all the sampling strategies for every dataset even at the highest correction level. Whether this trade-off is beneficial or not will ultimately be application-specific.

For the analysed fairness ratios, correction had a stronger effect when using *PS* at the same d-value. This is partly explained by *PS* resampling all four population subgroups

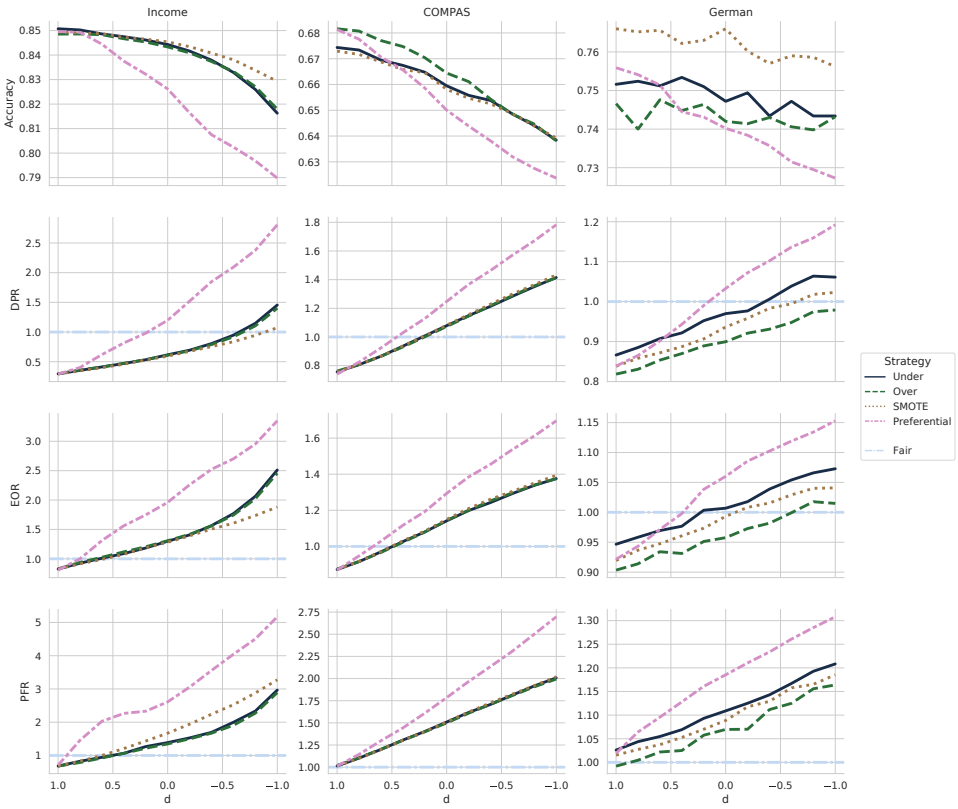


Figure 5.10 ACC and fairness ratios by correction level d for all three datasets.

at once, while the other three methods only resample two of said groups. This stronger effect of *PS* has both a negative consequence, though: the higher variability in fairness ratios due to the change in the value of d, i.e. optimising for a specific fairness ratio may magnify another ratio's unfairness. For every dataset, all the sampling methods achieved optimal DPR, EOpR and PFR, using different d-values. As may be observed in Figure 5.10, achieving DPR required greater correction than EOpR, which in turn required greater correction than PFR. For *Under*, *Over* and *SMOTE*, it took d-values close to -1 to achieve optimal DPR.

Table 5.1 shows diverse performance metrics for PARDS using the different sampling strategies to correct γ_{sr} on *Income*. The presented means and confidence intervals (CIs) result from 100 independent train/test splits, then using each sampling strategy with the optimal d-value for each estimated through Bayesian optimisation performed on the training set. As may be seen, there is a big difference in computing time across strategies, with *SMOTE* being over 10 times slower than *Under*. On the other hand, *SMOTE* produced the best scores for most performance metrics. Optimal fairness correction was achieved within the CIs for all methods, with roughly the same ACC-loss trade-off. Interestingly,

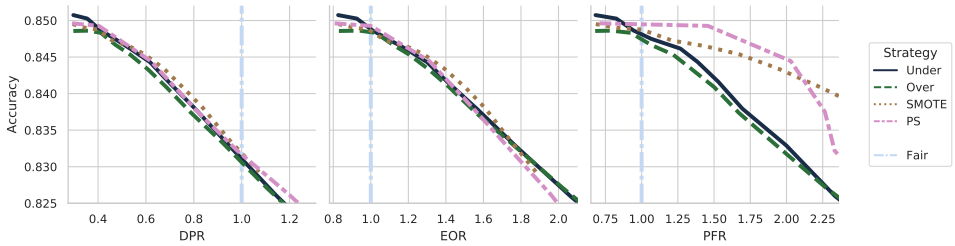


Figure 5.11 Fairness/accuracy trade-off for DPR, EOpR and PFR on *Income*.

running *Under* before training the classifier was 35% faster than just training the classifier over the full dataset. This would provide an additional advantage for *Under*-corrected training sets when learning models from large-scale datasets.

PA Coefficients

Scatter plots of PFR vs the PA coefficient of the 132 produced LR classifier-fits—obtained from fitting eleven d-values using four correcting strategies for three datasets—revealed an interesting relationship between these two quantities: the fit regression line for each dataset's scatter passes through point $(1,0)$, as may be seen in Figure 5.12. This happens because the coefficient for a feature being 0 means that the feature is completely irrelevant for the classifier (and its predictions). Hence the interventions $do\,(\text{PA} = U)$ and $do\,(\text{PA} = F)$ will have no effect on the resulting classifier. In other words, a LR classifier with PA coefficient 0 will satisfy PFR; on the other hand, such a relationship was not found for DPR and EOpR with respect to the PA coefficient.

5.4.3 Comparison with Other Methods

An intrinsic advantage of PARDS is that it can optimise a classifier with respect to different group fairness definitions. Three definitions: γ_{sr} [22], *discrimination (disc)* [90] and *EOD* [81] were used for the comparisons. They are defined as follows:

$$disc := \text{PR}(F) - \text{PR}(U), \tag{5.4}$$

$$\gamma_{sr} := \frac{\min\limits_{a \in \text{PA}} \mathbb{P}(\hat{Y} = 1 \mid \text{PA} = a)}{\max\limits_{a \in \text{PA}} \mathbb{P}(\hat{Y} = 1 \mid \text{PA} = a)}, \tag{5.5}$$

$$\text{EOD} := |\delta FPR| + |\delta FNR|, \tag{5.6}$$

Table 5.1 Fairness and performance metrics comparison using the four sampling strategies on *Income*, optimising LR models for γ_{sr} with 95% CIs. The best score for each metric is highlighted.

	No Correction	*Under*	*Over*	*SMOTE*	*PS*
Estimated Optimal d		–0.5770	–0.6083	–0.8246	0.1784
Time (s/iteration)	1.324 ± 0.018	**0.857 ± 0.014**	1.952 ± 0.030	10.974 ± 0.085	1.237 ± 0.016
Discrimination	0.178 ± 0.003	–0.001 ± 0.002	**0.0 ± 0.003**	0.006 ± 0.003	–0.003 ± 0.003
γ_{sr}	0.295 ± 0.007	1.008 ± 0.018	**1.005 ± 0.019**	0.966 ± 0.018	1.023 ± 0.018
ACC	0.85 ± 0.001	0.832 ± 0.001	0.831 ± 0.001	**0.833 ± 0.001**	0.831 ± 0.001
Balanced Accuracy	0.761 ± 0.002	0.695 ± 0.002	0.692 ± 0.002	**0.717 ± 0.002**	0.712 ± 0.002
Precision	0.737 ± 0.003	**0.77 ± 0.004**	**0.77 ± 0.004**	0.727 ± 0.004	0.726 ± 0.004
Recall	0.59 ± 0.003	0.43 ± 0.004	0.425 ± 0.003	**0.493 ± 0.003**	0.482 ± 0.003
F-Score	0.655 ± 0.003	0.552 ± 0.003	0.547 ± 0.003	**0.587 ± 0.003**	0.58 ± 0.003
ROC AUC	0.761 ± 0.002	0.695 ± 0.002	0.692 ± 0.002	**0.717 ± 0.002**	0.712 ± 0.002

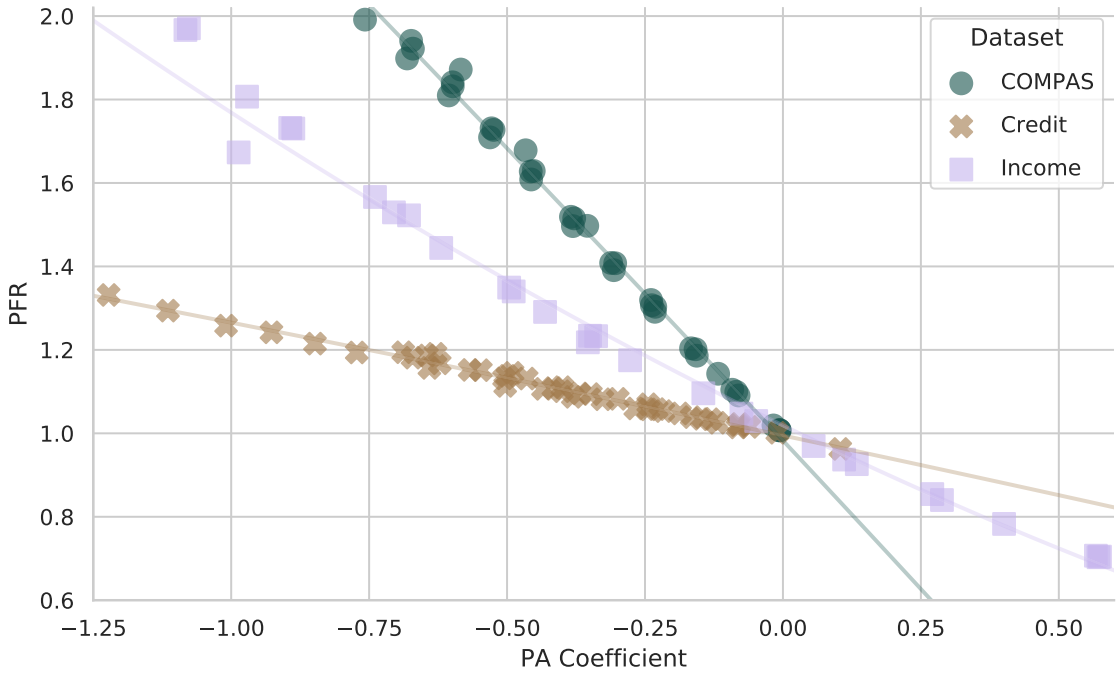


Figure 5.12 PFR by PA coefficient scatter plot for all three datasets, with quadratic regression curves.

where δFPR and δFNR represent the difference of U's and F's false positive and false negative rates, respectively.

Tables 5.2 and 5.3 compare PARDS with a variety of *preprocessing* [25, 81, 89, 97, 20, 138], *inprocessing* [1, 22, 168, 167, 170] and *postprocessing* [72] fairness-correcting methods.

Since four different classifiers were used on the papers to compare with—AB, DT, *gaussian naive bayes (GNB)* and LR—PARDS' results are presented using all three of them. Classifiers were optimised to compare with the state-of-the-art methods, hence three of the presented methods are optimised for DPR and two are optimised for EOD, as defined in Equation (5.6).

Metrics were evaluated using the same classifiers as the ones used in the papers we compare with. The objective functions to optimise were $|1 - \text{DPR}|$ and $|1 - \text{EOD}|$ for DPR and EOD, respectively, with 0 being the best value the objective function may take in both cases.

For every tested d-value the resulting DPR of 50 random 90/10 train/test splits were averaged, finding optimal d-values of {0.8338, -0.1803, -1.1528, -0.6083} for AB, DT, GNB and LR, respectively. All of the classifiers were trained using the default scikit-learn hyper-parameter values; using these parameter values, PARDS was run 10 times, averaging

Table 5.2 Fairness metrics and ACC comparison of the DPR-optimised PARDS with related fairness-correcting methods. The best result for each metric is highlighted.

Algorithm	Classifier	*disc*	γ_{sr}	ACC
No Correction	LR	0.18	0.30	0.85
PARDS (*Over*)	LR	**0.00**	**1.00**	0.83
PARDS (*PS*)	GNB	**0.00**	0.98	0.82
PARDS (*PS*)	DT	0.01	0.97	**0.84**
Kamiran and Calders [89]	DT	0.03	—	**0.84**
Zemel et al. [170]	LR	0.20	—	0.68
Calmon et al. [20]	LR	0.03	—	0.79
Zafar et al. [168]	GNB	—	0.87	0.77
Hardt et al. [72]	GNB	—	0.85	0.81
Zafar et al. [167]	GNB	—	0.42	**0.84**
Agarwal et al. [1]	GNB	—	0.72	0.79
Celis et al. [22]	GNB	—	0.95	0.77

Table 5.3 EOD, ACC and balanced accuracy comparison of the EOD-optimised PARDS with related fairness-correcting methods. The best result for each metric is highlighted.

Algorithm	Classifier	EOD	ACC
No Correction	AB	0.18	0.86
PARDS (*PS*)	AB	0.08	**0.86**
PARDS (*PS*)	LR	0.09	0.83
Krasanakis et al. [97]	LR	**0.05**	0.82
Iosifidis and Ntoutsi [81]	AB	0.08	0.83
Chawla et al. [25]	AB	0.47	0.81

the resulting metrics. The fairness and ACC metrics for the compared methods refer to the *best* reported values in [89, 170, 22, 81, 138]. Likewise, for methods evaluated on more than one classifier, the best one is presented.

The PARDS EOD-optimised AB classifier produced the best overall ACC (86%), while showing an EOD value within 3% of the best performing method [97]. Regarding the PARDS DPR optimised classifiers, although LR performed the best overall, both PARDS' DT and GNB performed better than the other methods' DTs and GNBs, respectively. Interestingly, PARDS' LR produced the fairest classifiers with respect to definitions (5.5) and (5.4), even though they were actually optimised for DPR. While the ACC of DPR-optimised PARDS LR was not the best (83%), it came within 1% of the best performing classifiers (PARDS' DT, Kamiran and Calders [89] and Zafar et al. [167], with an ACC of 84%).

5.5 Conclusion

This chapter presented PARDS, a parametrised fairness optimisation method agnostic to both fairness definitions and classifiers. Correcting through training set resampling, it has been shown that PARDS produces fairness-optimal predictions with a small loss in predictive power. When compared with the existing methods, in most cases PARDS produces the best fairness performance.

In future work further improvements could me made on the data resampling methods, in order to optimise for different fairness definitions at once. Although PARDS shows a relatively low impact on prediction performance and its main objective is to estimate the optimal amount of correction with respect to fairness, considering predictive performance as well would be interesting, either in the form of a restriction—e.g. a maximum loss in ACC or a minimum level of fairness—or by setting an acceptable trade-off rate between both metrics.

Chapter 6

Fairness and Privacy

This chapter is based on González-Zelaya et al. [64], submitted to the *Special Issue on Bias and Fairness in AI* of the Data Mining and Knowledge Discovery (DAMI) journal, which is an extension of Salas and González-Zelaya [135].

Contents

Summary

Protecting the privacy of personal data and enforcing fairness in automated decisions are fundamental requirements for responsible ML. Both may be achieved through data preprocessing, and share a common target: that data should remain useful for a task, while becoming uninformative of sensitive information; this chapter resides at the intersection of both areas. The intrinsic connection between privacy and fairness implies that modifications performed to guarantee one of these goals may have an effect on the other, e.g. hiding a sensitive attribute from a classifier may prevent a decision rule from using such attributes as part of its decision criteria. Experimental evidence is presented on the effect of differential privacy, t-closeness and k-anonymity over fairness. A second experiment shows how the two goals are compatible, and may be simultaneously achieved with a small loss in predictive performance by using the presented FAIR-MDAV algorithm. FAIR-MDAV's results are competitive with both state-of-the-art fairness correcting algorithms and hybrid privacy-fairness methods.

6.1 Introduction

Protecting the privacy of personal data and enforcing fairness in automated decisions are fundamental requirements for responsible ML. Both ideals share a common element: the need to hide or protect certain attributes of the available data. Who the data are being hidden from and the reason for hiding them represents the essential difference between these two disciplines. While privacy seeks to protect the sensitive attributes from being collected (or inferred) by a third-party, fairness strives to prevent the decision mechanism from learning potentially discriminatory biases with respect to these attributes. In the most elementary kind of fairness, a classifier is denied access to the attribute for which discrimination is to be prevented. However, this does not guarantee that the learnt decision rule will be discrimination-free, as the bias may not only lie in the removed attributes, but on other features correlated with them as well.

This chapter is related to different areas of responsible ML, such as fair clustering, fair classification and privacy-preserving data mining, as described in Section 6.2.

Contributions This chapter introduces FAIR-MDAV, a fairness correction method for classification tasks with t-closeness and k-anonymity guarantees. FAIR-MDAV is a modular system allowing for both privacy and fairness to be enhanced either separately or

all at once without any compromises, that may be adjusted through a protection parameter and a correction parameter.

Benchmark experiments are performed over three commonly used benchmarks for algorithmic fairness and privacy: *Income*, *COMPAS* and *German*, measuring several performance metrics as well as three fairness metrics. FAIR-MDAV is competitive in enforcing demographic parity, and outperforms state-of-the-art EOD methods at its fairness/accuracy trade-off.

While FAIR-MDAV is not an unsupervised learning method, it makes use of the MDAV clustering algorithm [42], and it also borrows the notion of *fairlet* from fair clustering [28].

6.2 Background and Definitions

The scope of this chapter lies between the research areas of algorithmic fairness and data privacy. A list of privacy definitions is presented to clarify the similarities and differences between both areas. This chapter will focus on the DPD and EOD fairness definitions. We use the *difference* version of these fairness definitions presented in Subsection 4.1.1; note again that a smaller fairness difference represents a fairer classifier.

The *fairlet* concept, borrowed from *fair clustering*, is defined as follows:

Definition 9 (Fairlet)**.** Given a dataset D with binary PA taking values F and U, an (m,n)-*fairlet* of D is defined as a subset of D with m instances such that PA $= U$ and n instances such that PA $= F$.

In the clustering context [28, 7], fairlets are used to obtain *fair clusters*, i.e. clusters in which the PA distribution is similar to the whole dataset. In FAIR-MDAV's case, the purpose of fairlets is to correct fairness locally and to anonymise the data.

Privacy

Removing all of the *identifiers* from a dataset, such as social security numbers or names and surnames does not necessarily protect individuals from re-identification, as they may still be associated with unique combinations of attribute values revealed by a data mining process.

Two of the main models for privacy-preserving data mining are k-anonymity (and its enhancements, e.g. t-closeness) and differential privacy. The differences and interactions between these two models are discussed in Salas and Domingo-Ferrer [134].

The confidential or *sensitive attributes* (SAs) are attributes that contain sensitive information about an individual, e.g. salary, medical condition or religious beliefs. *Quasi-identifiers* (QIs) are attributes such that a unique combination of their values may be used to single out an individual for their re-identification, e.g. gender, date of birth and postal code [148]. In this chapter's setting, the data features are split into SAs and QIs, with the SAs consisting of the PA and the label.

To prevent re-identification, k-anonymity is defined in Samarati [139], Sweeney [148] as follows.

Definition 10 (k-Anonymity)**.** A dataset is *k-anonymous* if each instance is indistinguishable from at least other $k-1$ instances within the dataset with respect to its QIs.

One way to produce a k-anonymous dataset D is through micro-aggregation, i.e. by combinining the QIs of instances from size-k-groups through an aggregation function, e.g. their mean or median values, resulting in a group of k instances with identical QIs. These groups will be referred to as D's *k-groups*.

Privacy by k-anonymity guarantees that the probability of associating an individual with their record is lower than $1/k$, i.e. an *adversary* knowing some of their characteristics will not be able to distinguish their record among a group of k similar records. The distribution of the SAs in a k-group, however, may be used to infer the SA-values of an individual, even if complete re-identification is not possible. For example, if all the SA-values in a k-group are the same, an adversary may use their knowledge of the QIs of an instance to associate it with a k-group and thus learn the SA-value from the k-group. To prevent this problem, an additional constraint known as t-closeness [110] may be introduced. In the k-anonymity context, t-closeness is defined as follows:

Definition 11 (t-Closeness)**.** Given a k-anonymous dataset D, a k-group belonging to D is said to have t-closeness if the similarity between the distributions of an SA in the k-group and in D is smaller than a threshold t. A dataset D is said to have t-closeness if all of its k-groups have t-closeness.

The idea behind t-closeness is that, by making the distribution of the SAs in the k-groups similar to the distribution of the SAs in the dataset, it will not be possible to infer an instance's SAs, even if the instance's k-group is known. In FAIR-MDAV's case, the distribution's similarity is measured by comparing the proportion of U and F PAs in the k-groups, as well as on D; this is consistent with the definition of fairlet (Definition 9).

Differential privacy [45], on the other hand, is a widely used post-processing approach for privacy protection, where noise is added to queries, e.g. a classifier's predictions, to hide an instance's SA values.

6.3 FAIR-MDAV

The FAIR-MDAV algorithm (Algorithm 6.1) consists of three methods:

1. Given a desired cluster size k, `MakeFairlets` (Algorithm 6.2) clusters the training set D into G, a collection of (m,n)-fairlets such that the fairlet's elements are close to each other, where m/n approximates the U/F proportion of D as closely as possible subject to $m+n=k$. Clusters are formed in the following way: Let $\bar{r}$ be D's "average record", i.e.
$$\bar{r} := \frac{1}{|D|}\sum_{r\in D} r.$$
`MakeFairlets` first locates e^*, the furthest element in D from $\bar{r}$, i.e.
$$e^* = \arg\max_{e\in D} d(e,\bar{r}),$$
where d is the Euclidean distance. Letting
$$D_i := \{e \in D \setminus \{e^*\} \mid \mathrm{PA}(e) = i\} \text{ for } i \in \{0,1\},$$
$F_{e^*}, U_{e^*} \subset D$ is defined for the $\mathrm{PA}(e^*) = 0$ case as follows[1]:
$$F_{e^*} := \arg\min_{A\in[D_1]^m} \sum_{a\in A} d(a,e^*), \tag{6.1}$$
$$U_{e^*} := \arg\min_{A\in[D_0]^{n-1}} \sum_{a\in A} d(a,e^*). \tag{6.2}$$
In the $\mathrm{PA}(e^*) = 1$ case, replace m with $m-1$ and $n-1$ with n in Equations 6.1 and 6.2, respectively. Then, $g := \{e^*\} \cup F_{e^*} \cup U_{e^*}$ will be an (m,n)-fairlet where F_{e^*} and U_{e^*} are the nearest *favoured* and *unfavoured* neighbours of e^*, respectively. Fairlet g is appended to G (D's fairlet partition), the elements of g are removed from D and

[1] Notation: $[A]^k = \{B \subset A \mid |B| = k\}$, i.e. the set of all k-element subsets of A.

`MakeFairlets` iterates over the remaining records in D until no more fairlets can be formed, discarding the remaining records.

2. `Micro-aggregate` (Algorithm 6.3) replaces the original records' feature values with the corresponding aggregated feature values of the fairlet they belong to, e.g. with the mean values. The only exceptions to this are the PA and the label, for which the original values are kept.

3. `CorrectFairness` (Algorithm 6.4) locally corrects the fairness of each fairlet by relabelling its records depending on their PA values so that $\text{PR}(U) \geq \tau \cdot \text{PR}(F)$, where τ modulates the amount of correction introduced to the data. Fairness correction may occur by relabelling negative *unfavoured* instances as positive (*positive correction*), or by relabelling positive *favoured* instances as negative (*negative correction*).

Algorithm 6.1: FAIR-MDAV algorithm.

```
input : D: dataset to process, with binary PA-and-label,
        nu: number of unfavoured records per fairlet,
        nf: number of favoured records per fairlet,
        ma: boolean whether entries are micro-aggregated or not,
        tau: float, the level of fairness correction,
        nc: boolean whether negative correction is performed.
G ← MakeFairlets(D, nu, nf);                                  // Algorithm 6.2
G_micro ← G;                          // No micro-aggregation if ma is False
if ma is True then
    G_micro ← Micro-aggregate(G);                             // Algorithm 6.3
end
D_corrected ← CorrectFairness(D, G_micro, tau, nc);           // Algorithm 6.4
return D_corrected
```

Simplified Example

Figure 6.1 exemplifies FAIR-MDAV being applied to a 7-entry minimal dataset consisting of one continuous feature (X) and binary PA and Y-label. In this particular example, FAIR-MDAV micro-aggregates the data into $(2,1)$-fairlets in the following way:

1. The average value of X is 8, and the record with the farthest X-value from 8 is `A`.

2. Since `A` has PA $= 1$, to complete a $(2,1)$-fairlet two records must be added: one with PA $= 0$ and one with PA $= 1$. The valid records closest to `A` are `B` and `D`. Hence, the first fairlet will be $\{\texttt{A},\texttt{B},\texttt{D}\}$, coloured orange in Figure 6.1.

Example Data

id	X	PA	Y
A	1	1	1
B	2	0	0
C	3	0	1
D	11	1	0
E	12	1	0
F	13	1	1
G	14	1	1

→

$(2,1)$-Fairlets

id	X	X_{ma}	PA	Y	PC	NC
A	1	4.67	1	1	1	0
B	2	4.67	0	0	1	0
C	3	9.33	0	1	1	1
D	11	4.67	1	0	0	0
E	12	9.33	1	0	0	0
F	13	9.33	1	1	1	1
G	*Dropped*					

Figure 6.1 Micro-aggregates generated by FAIR-MDAV, coloured in orange and purple, with micro-aggregated features (X_{ma}) and Y-labels corrected positively (PC) and negatively (NC).

3. After removing these three records, the process is repeated. The second fairlet becomes $\{\texttt{C},\texttt{E},\texttt{F}\}$, coloured purple.

4. Since three records are required to build a $(2,1)$-fairlet and only one element remains, it is dropped (G).

5. Once the fairlets are grouped, the PA may be positively (PC) or negatively (NC) corrected. For the orange fairlet, the uncorrected PRs are 0 for its *unfavoured* elements and 0.5 for its *favoured* elements. When applying PC, record B's Y-label changes from 0 to 1, increasing the unfavoured records' PR from 0 to 1. Conversely, when applying NC, records A and C are relabelled from 1 to 0, which changes the favoured records' PR from 1 to 0. On the other hand, the purple fairlet needs no correction, as the unfavoured PR is already greater than the favoured PR.

Assuming $\tau = 1$, the PC and NC columns display the positive and negative fairness correction relabellings, respectively, which are performed fairlet-wise. For PC, entries with PA $= 0$ and $Y = 0$ get relabelled as 1, as long as the proportion of PA $= 1$ with $Y = 1$ is higher than the proportion of PA $= 0$ with $Y = 1$. For NC, it is PA $= 1$ with $Y = 1$ entries that get relabelled to 0 as long as the same condition holds.

Algorithm 6.2: `MakeFairlets` method

input : D, nu, nf

$G \leftarrow \{\}$;
while $|\{x \in D \mid pa(x) = 0\}| \geq nu$ **and** $|\{x \in D \mid pa(x) = 1\}| \geq nf$ **do**
    x_mean $\leftarrow$ Mean($\{x \in D\}$);
    x_r $\leftarrow \arg\max_{x \in D}$(distance(x, x_mean));
    /* Get nu, nf nearest records to x_r with pa = 0, pa = 1 */
    g_u $\leftarrow$ GetNearestU(x_r, nu);
    g_f $\leftarrow$ GetNearestF(x_r, nf);
    g $\leftarrow$ Append(g_u, g_f);
    D $\leftarrow$ Drop(D, g);
    G $\leftarrow$ Append(G, g);
end
return G

Algorithm 6.3: `Micro-aggregate` method

input : G

/* Replace all the records in a fairlet by their mean value */
G_micro $\leftarrow \{\}$;
for g **in** G **do**
    g_mean $\leftarrow \{\}$;
    x_mean $\leftarrow$ Mean($\{x \in g\}$);
    foreach x **in** g **do**
        Append(g_mean, x_mean);
    end
    Append(G_micro, g_mean);
end
return G_micro

Algorithm 6.4: `CorrectFairness` method

```
input : G
/* Calculate the positive ratio for the favoured and unfavoured
   groups, replacing as many unfavoured negatives with favoured
   positives as needed, dependant on tau                        */
for g in G do
    fpr ← PositiveRatio(g, 1);
    upr ← PositiveRatio(g, 0);
    U_0 ← {x in g | pa(x) = 0 and y(x) = 0};
    F_1 ← {x in g | pa(x) = 1 and y(x) = 1};
    if nc is False then
        while upr < tau * fpr and |U_0| > 0 do
            y(x) = 1 for x in U_0
        end
    end
    else
        while upr < tau * fpr and |F_1| > 0 do
            y(x) = 0 for x in F_1
        end
    end
end
D_corrected ← Concatenate({g in G});
return D_corrected
```

Table 6.1 Experimental parameter values.

Parameter	Values	Description
u	$\lfloor 10i \cdot \frac{\lvert U \rvert}{\lvert D \rvert} + 0.5 \rfloor$ for $i \in 1 \dots 10$	(u, f)-fairlet size [$f = 10i - u$]
k	$10i$ for $i \in 1 \dots 10$	k-group size
ε	$10^{1-i/2}$ for $i \in 0 \dots 8$	Differential privacy
τ	$0.1i$ for $i \in 0 \dots 10$	Fairness correction level
ma	`True, False`	Micro-aggregation
nc	`True, False`	Negative correction

6.4 Experiments

Experiments were performed over three datasets commonly used as benchmarks in both the privacy and fairness literature: *Income*, *COMPAS* and *German*, described in Subsection 2.6.3. Given the instance-size difference between these datasets, patterns are clearer on *Income* and noisier on the *German* analyses.

A selection of fairness and performance metrics were evaluated exhaustively across the parameter values presented in Table 6.1. For each parameter combination, five-fold cross validation (CV) train-test splits were performed, applying FAIR-MDAV to the training sets and followed by training LR classifiers from the modified training sets. Fairness and performance metrics were evaluated over the corresponding test sets. The resulting metrics were averaged across CV splits using the same parameters. *scikit-learn*'s [142] LR implementation was used to learn the corresponding classifiers.

Fairness/Accuracy Trade-Off

The modularity of FAIR-MDAV makes the privacy and the fairness correction components independent of each other. It is nonetheless interesting to analyse the impact of choosing certain parameter values on the effectiveness of the correction. Figure 6.2 presents a scatter plot of EOD vs ACC—averaged across the CV folds on the test set—for every experiment performed.

Regarding fairness, better EOD scores were achieved with *negative correction* than with positive correction. Although micro-aggregation caused "fairer" classifiers to be learnt, this occurred at the cost of an increase in ACC loss. A detailed description of Figure 6.2 follows, which shows the effects of fairness correction across the parameter values presented in 6.4 for the *Income* dataset. Analogous plots to Figures 6.2 and 6.3

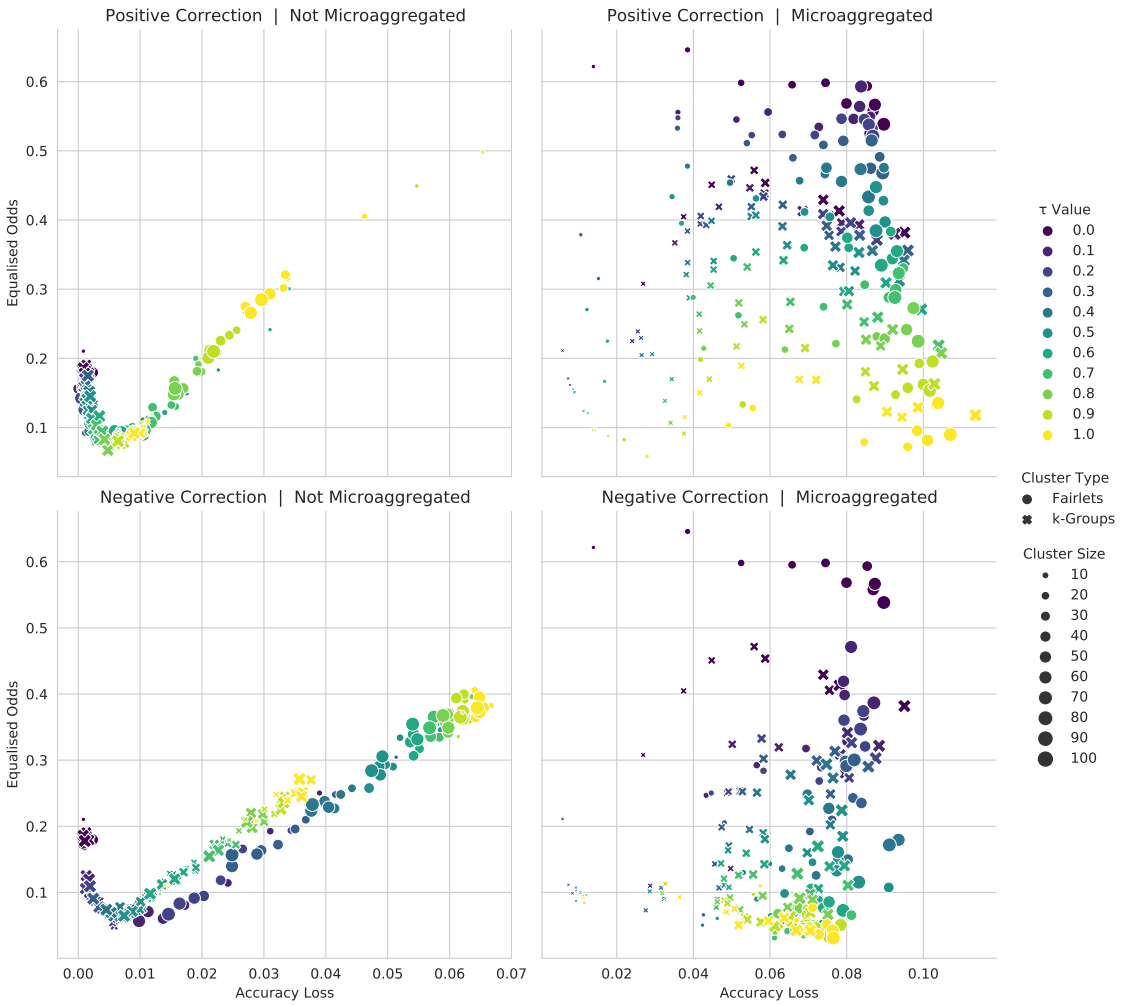


Figure 6.2 EOD/ACC trade-off on *Income* across FAIR-MDAV parameters. For both axes lower is better.

for *COMPAS* and *German*—displaying similar behaviours—are presented in Figures 6.4 and 6.5, respectively.

Fairness Correction Without Micro-Aggregation

As may be observed in the left-side of Figure 6.2, when data is not micro-aggregated fairness correction and ACC loss are strongly correlated, i.e. higher τ values not only lead to improved outcomes for the unfavoured group, but also cause the performance of the learnt classifiers to drop. The "fairest" non-micro-aggregated classifiers produced an ACC higher than 84%, with much less correction being required for it than in the *micro-aggregated* case. Depending on the correction and cluster types, τ values between

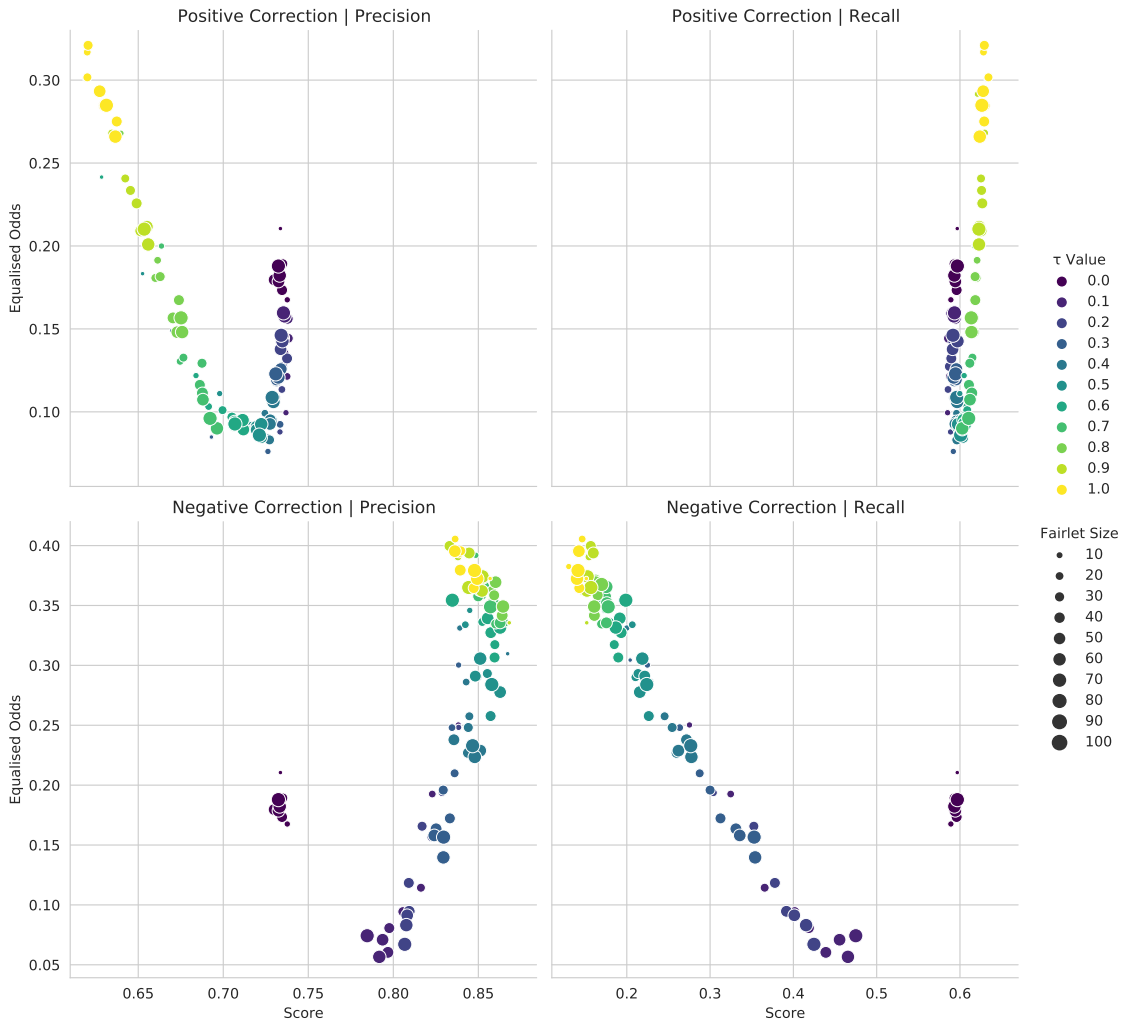


Figure 6.3 Precision and recall vs EOD for non-micro-aggregated positive and negative correction experiments on *Income*.

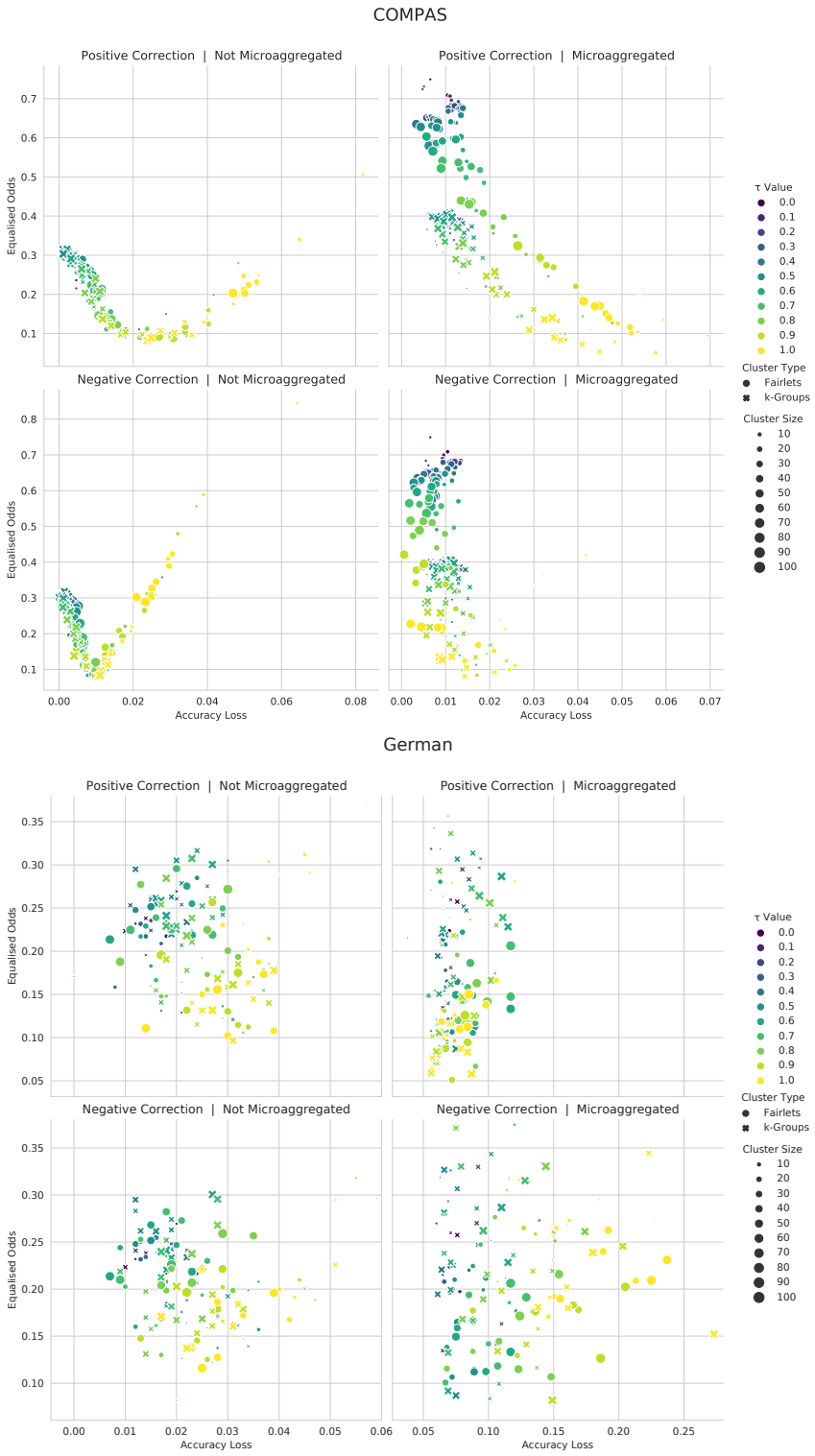


Figure 6.4 EOD/ACC trade-off on *COMPAS* and *German* across FAIR-MDAV parameters. For both axes lower is better.

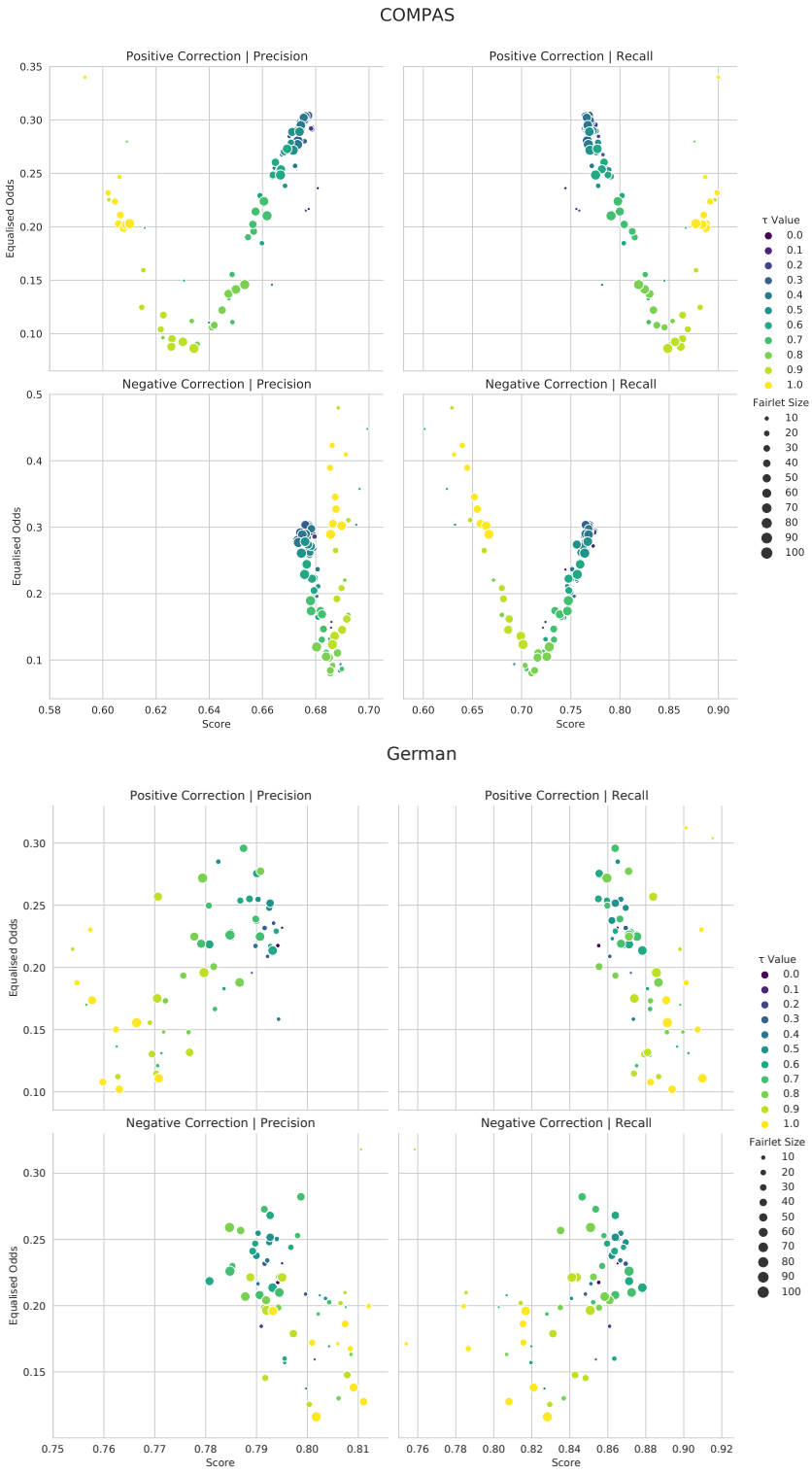


Figure 6.5 Precision and recall vs EOD for non-micro-aggregated positive and negative correction experiments on *COMPAS* and *German*.

0.6 and 0.8 produced optimal EOD. Larger τ values lead to a fairness over-correction, i.e. the unfavoured group's PRs become much better than the favoured group's, and hence unfairness happens, only in a reversed way, i.e. the favoured group becomes unfavoured and vice versa.

The left-side plots of Figure 6.2 also show that the cluster-type used to enforce fairness has an effect on the correction amount: lower τ-values are required to achieve an equivalent amount of benefit for U in fairlets than in k-groups. This is only natural, since FAIR-MDAV's fairness correction only has an effect on heterogenous-PA groups, i.e. where members of both PA groups are present (see Figure 6.1). In contrast with the micro-aggregated case, on the non-micro-aggregated datasets the effect of the cluster sizes is small to non-existent on both fairness and ACC.

Finally, the effect of τ with negative fairness correction is stronger than with positive correction for both cluster types. However, negative correction also causes an increased drop in ACC for the same τ-value. The author believes that, besides the experimental results shown, negative correction is harder to ethically justify, since it implies punishing members of the favoured group without an obvious gain to the unfavoured group, i.e. fairness is achieved with a lessened amount of overall well-being.

Fairness Correction With Micro-Aggregation

The right-side plots of Figure 6.2 showcase the effect of τ correction over micro-aggregated *Income* datasets. The sparsity in ACC loss is mostly explained by micro-aggregation, with cluster-size being the dominant factor that determines it: on both micro-aggregated plots, the larger the cluster, the more ACC will be lost with respect to the original dataset. Fairness correction, on the other hand, has a slight impact on ACC loss (much less severe than in the non-micro-aggregated experiments case), while having a large, descending impact on EOD. An interesting difference with the non-micro-aggregated case is that, on the micro-aggregated datasets, setting $\tau = 1$ actually achieves the best possible fairness for every cluster size and type. The best EOD scores achieved by any micro-aggregated dataset have an ACC loss of roughly 8%, while the best EOD scores were achieved on the non-micro-aggregated datasets with an ACC loss of roughly 1%. However, in the micro-aggregated case this fairness loss also includes the k-anonymity and t-closeness privacy guarantees provided by design through FAIR-MDAV.

Precision and Recall

Figure 6.3 shows the resulting precision and recall across the micro-aggregated experiments on *Income*. As may be seen, fairness correction impacts precision and recall in opposite directions, and it does so differently for positive and negative correction: while for positive correction higher τ-values cause worse precision and better recall scores, for negative correction higher τ-values imply better precision but worse recall.

The precision/recall trade-off is a well-known phenomenon [17], but in FAIR-MDAV's case it follows from the way it corrects fairness: positive correction relabels negative records as positive, relaxing the constraints upon which a data instance would be classified as positive. This, in turn, causes the learnt classifier to predict fewer false negatives (improving recall) at the cost of labelling more false positives (worsening precision). With negative correction the effect is precisely the opposite: the constraints for being classified as positive are stronger, hence there will be fewer false positives (better precision) but more false negatives (worse recall). Since classification tasks may find one performance metric to be more relevant than the other (e.g. for fraud detection the false negative rate, i.e. recall, should be as small as possible), the choice of positive vs negative correction can aid in improving the relevant performance metric, as a bonus to correcting fairness.

Comparison with Existing Fairness Methods

FAIR-MDAV is compared with the following algorithms: *preferential sampling (PREF)* [90], *learning fair representations (LFR)* [170], *adaptive sensitive reweighting (ASR)* [97], *ADAFAIR* [81], *SMOTEBOOST* [25] and *parametrised correction (*PARDS*)* [65]. The fairness and ACC metrics for these methods refer to the *best* reported values in Kamiran and Calders [90], Zemel et al. [170], Iosifidis and Ntoutsi [81], González-Zelaya et al. [65]. Likewise, for methods evaluated on more than one classifier, the best one is presented.

Table 6.2 compares DPD, EOD and ACC scores on *Income* for FAIR-MDAV when used purely for fairness correction against state-of-the-art methods. FAIR-MDAV achieves slightly worse DPD scores than PARDS. When compared to PREF, FAIR-MDAV achieves a better DPD score, however it does so at a higher cost in ACC.

When compared with EOD-optimising methods, FAIR-MDAV achieved the best score with a big loss in ACC. However, for an EOD score of 0.05, the same as the second best EOD-optimising method (ASR), FAIR-MDAV's ACC is 2% better than ASR, and just 1% below PARDS, which achieved a much worse EOD.

Table 6.3 presents the best DPD and EOD for FAIR-MDAV in the k-anonymity and t-closeness cases, and these are compared with *PFLR** [164]. When only k-anonymity is

Table 6.2 Fairness metrics and ACC comparison of FAIR-MDAV with related fairness-correcting methods over *Income*. The top half compares with DPD-correcting methods, while the bottom half compares with EOD-correcting methods. The compared metrics were obtained by training different classifiers, namely LR, decision trees (DT) and AdaBoost (AB). The best result for each metric is highlighted.

Method	Classifier	DPD	EOD	ACC
FAIR-MDAV	LR	0.01	—	0.80
LFR [170]	LR	0.20	—	0.68
PARDS [65]	LR	**0.00**	—	0.83
PREF [90]	DT	0.03	—	**0.84**
FAIR-MDAV (t-close)	LR	—	**0.03**	0.79
FAIR-MDAV (same EOD as ASR)	LR	—	0.05	0.85
ASR [97]	LR	—	0.05	0.82
ADAFAIR [81]	AB	—	0.08	0.83
PARDS [65]	AB	—	0.08	**0.86**
SMOTEBOOST [25]	AB	—	0.47	0.81

guaranteed, the same parameter value ($k = 100$) yields the best score for both DPD and EOD. In the t-closeness case, $t = 0.05$ ($k = 10$) produced a better ACC and the same DPD as PFLR* (with $\varepsilon = 10$). The best EOD resulted from FAIR-MDAV with $t = 0.19$ ($k = 20$). PFLR* with $\varepsilon = 0.1$ presents the overall best DPD, at the cost of higher ACC loss.

Table 6.3 Fairness and ACC comparison of FAIR-MDAV with PFLR* [164].

Algorithm	Privacy	DPD	EOD	ACC
FAIR-MDAV (k-anon. both)	$k = 100$	0.05	0.04	0.78
FAIR-MDAV (t-close, DPD)	$t = 0.05$ ($k = 10$)	0.02	0.11	**0.79**
FAIR-MDAV (t-close, EOD)	$t = 0.19$ ($k = 20$)	0.04	**0.03**	**0.79**
PFLR*	$\varepsilon = 0.1$	**0.00**	—	0.75
PFLR*	$\varepsilon = 1$	0.01	—	0.76
PFLR*	$\varepsilon = 10$	0.02	—	0.76

6.5 Conclusion

FAIR-MDAV is a fairness-correcting preprocessing method with privacy guarantees. It outperforms existing fairness-correcting methods on its EOD/ACC trade-off, and is competitive on its DPD/ACC trade-off as well. FAIR-MDAV is modular, allowing for

privacy guarantees to be set separately from fairness correction. It is also definition-agnostic for group-fairness definitions. The effects of privacy over fairness were shown by comparing three algorithms with different privacy guarantees (t-closeness, k-anonymity and ε-differential privacy) as well as the interactions resulting from the different method parameters.

In the future, FAIR-MDAV could be extended to work on multi-class PA and label, different distance definitions could be tried on the method's clustering phase, and FAIR-MDAV's effect on individual fairness could be analysed.

Chapter 7

Conclusions

Contents

Summary

Algorithmic fairness is an active area of research in both the industry and in academia. Besides being enforced by regulations such as the GDPR, fairness is actively pursued by society. Along the continued debate between different fairness definitions, regulations make implementing fair mechanisms a necessity for anyone interested in automated decision-making solutions. This thesis has presented an overview of algorithmic fairness with a focus on fairness correction for classification tasks through data preprocessing. Three different fairness-correcting algorithms, PARDS, FAIRPIPES and FAIR-MDAV, were presented, with differentiating specifications that make each of them suitable for specific scenarios. Regardless of their differences, all three algorithms share their fairness-definition and classifier agnosticism, providing them with the flexibility to be integrated into existing ML solutions, and adhering to different fairness world-views.

7.1 Summary of Contributions

This thesis' research aim was **to design, implement and evaluate data preprocessing methods that correct unfair predictions in classification tasks**. This general aim was divided into five specific objectives. In this section, each of these objectives is elaborated upon with the intent of discussing how the thesis addressed them.

The first research objective was **to understand and formalise the concept of algorithmic fairness**. After the real-world examples presented in Chapter 1 that motivate its study, Chapters 2 and 3 present material aimed specifically at this objective. Chapter 2 presented essential notions of classifiers, performance metrics as well as a compendium of popular preprocessing operators. Chapter 3 formalised the notion of algorithmic fairness, presenting the group fairness definitions used throughout the thesis, elaborated on the fairness/accuracy trade-off, and presented related work on fairness correction for classification tasks. These three chapters serve not only as background to position and understand the subsequent chapters, but could also work as an introduction to algorithmic fairness for anyone interested in the topic.

The second objective was **to determine whether the choice and order of the non-fairness-specific preprocessing operators have an impact on the fairness of the predictions made by a classifier learnt from the resulting data**. This objective was addressed in Chapter 4, where it was found that both the choice of preprocessing operator for a specific task, as well as the order in which a set of these tasks is applied to the data have an impact on both fairness and performance metrics. This finding is exploited by FAIRPIPES

to find near-optimal preprocessing pipelines with respect to either fairness, accuracy, or a linear combination of both metrics through a genetic-algorithm search.

The third objective was **to estimate the trade-off between fairness and performance for classifiers, generating a range of possible trade-offs from which to choose**. This objective was also addressed in Chapter 4, where the genetic-algorithm search performed by FAIRPIPES presents the user with an approximation of the preprocessing-pipeline space's Pareto front, allowing them to select a trade-off adequate to their needs. The Pareto front estimation was shown to be close to the ground-truth Pareto front and obtained efficiently, by evaluating roughly 6% of the possible preprocessing pipelines.

The fourth objective was **to design a fairness-definition and classifier agnostic preprocessing method to optimise the specified fairness definition for a classifier without incurring a big accuracy loss**. Making PARDS, FAIRPIPES and FAIR-MDAV agnostic to a particular fairness definition allows these methods to be useful for use-cases that adhere to the user's world-view. Classifier agnosticism is an inherent advantage of using the preprocessing approach to fairness correction which gives them the flexibility of being incorporated into existing ML solutions. PARDS, presented in Chapter 5, is the method that achieves the best fairness/accuracy trade-off out of the three, bench-marking better scores than state-of-the-art solutions.

The fifth and final objective was **to analyse the connection between fairness and privacy in the classification context and design a method capable of correcting fairness while providing privacy guarantees for the resulting dataset**. In Chapter 6, a direct effect between privacy and fairness was found empirically on the three tested datasets—*Income*, *COMPAS* and *German*—where higher levels of privacy produced better EOD for the resulting classifiers, albeit with a loss in the classifier's accuracy. The change in both of these quantities, accuracy loss and fairness gain depends on the choice of privacy guarantee. The second half of this objective was addressed by the design and implementation of FAIR-MDAV, a micro-aggregation method that achieves k-anonymity by design and performs a local fairness correction over each of the resulting micro-aggregates.

7.2 Lessons Learnt

The following list presents this thesis' key insights:

- There are many different fairness definitions, and it is usually impossible to enforce more than one of them at the same time. While a family of definitions focuses on individuals, a second one, known as *group fairness*, focuses on similar clas-

sifier behaviours across PA groups; the fairness-correcting methods presented in Chapters 4, 5 and 6 work on group fairness definitions.

- Among group fairness definitions, some focus on similarity across the PAs' predicted *outcomes*, e.g. demographic parity, while others focus on similarity across the PAs' *predictive performance*, e.g. equality of opportunity and equalised odds; these definitions align with different world-views regarding fairness. Fairness definitions and further discussion of their differences are presented in Chapter 3.

- There is a trade-off between fairness and performance whenever a classifier is learnt from biased data. However, these trade-offs will vary across different fairness definitions. In Chapters 4 and 5, demographic parity presented the highest accuracy-loss cost among the tested fairness definitions.

- The choice of preprocessing pipelines was found to impact the fairness of classifiers learnt from the preprocessed data. Optimal choices for preprocessing operators, however, are data-dependant. Therefore, a heuristic approach to estimate the optimal pipeline is necessary. FAIRPIPES, presented in Chapter 4, is a genetic-algorithm approach that presents the user with a near-optimal estimation of the preprocessing-pipeline space's fairness/accuracy Pareto front.

- One of the main sources of unfairness is training data. The reasons behind it may be both societal, e.g. data reflecting a historical discrimination against a PA-group, or technical, e.g. data were not collected adequately. Therefore, correcting the data by preprocessing it before training a classifier from it is a sound approach. Metaphorically, correcting the data may be thought of as sampling from a fairer world, in order to correct the real one: PARDS and FAIR-MDAV, presented in Chapters 5 and 6 are preprocessing fairness-correcting mechanisms that operate like this. Further discussion of the source of unfairness may be found in Chapter 3.

- Fairness and privacy share a common element: the need to hide or protect certain attributes of the available data. Who the data are hidden from and the reason for hiding them forms the essential difference between these two disciplines: while privacy seeks to protect the attributes from being collected (or inferred) by a third-party, fairness strives to prevent the classifier from learning potentially discriminatory biases with respect to these attributes. This common element makes it possible to fix both issues simultaneously. FAIR-MDAV, presented in Chapter 6, is a fairness-correcting mechanism that guarantees *k-anonymity*, a particular privacy definition, for the corrected data.

7.3 Future Research Directions

In the foreseeable future, the need for fair decision-making processes will continue to exist, and will probably become even more ubiquitous. The conclusions of Chapters 4, 5 and 6 present specific suggestions for future research with respect to PARDS, FAIRPIPES and FAIR-MDAV, respectively. The following is a list of interesting ideas on which future research could be done.

Multiple PA Optimisation Although a partial solution was presented in Chapter 5, optimising fairness for multiple PAs at once requires further research. Regarding the presented solution, which consists in evaluating a linear combination of each instance's PRs with respect to every PA, finding a criterion to add weight to this linear combination might enhance the resulting correction. Another problem which the proposed solution does not address is *fairness gerrymandering* [93], an infrequent case in which the combination of two or more *unfavoured* PAs might become a *favoured* subgroup.

Fairness as a Dynamic System To iterate fairness correction across time and explore different paths towards "fair in n years" scenarios. It may also be possible to progressively migrate across different fairness definitions, i.e. once a particular kind of fairness is achieved, strive for another kind of fairness without losing the one already attained. Specifically, we may think of periods of time, say a year, on which a certain decision, e.g. become eligible for credit, must be made and which will become the 'ground truth' for the following period, i.e. given an original training set T_0, a classifier $\hat{Y}_0$ may be learnt, which would assign new labels to the training set, obtaining a *new* training set T_1 with (possibly) modified labels. T_1 in turn would become the training set to learn $\hat{Y}_1$ and so on. In this sense, the ground truth would be continuously evolving in a similar fashion to a Markov process, but using a classifier instead of a set of equations. The resulting dynamic system could be analysed to look for convergence into a fixed state or 'unfairness loops' (where a fair state evolves into an unfair state and vice versa). In this setting, it may well be the case that adopting a more gradual fairness correction approach converges to a fair state, while extreme corrections may cause the mentioned unfairness loops, in which the favoured and unfavoured (e.g. male and female) groups would continuously swap roles.

An *AI for AI* Approach to Fairness To let the decision-making system self-regulate, by changing the optimisation problem depending on its current state, e.g. when the

system satisfies *demographic parity*, focus on *equality of opportunity* instead, or once fairness is sufficient, focus on the *performance* instead.

Support Individual Fairness or Causality Definitions To investigate whether individual or causal fairness can be enforced through data preprocessing. If so, to design and implement the necessary correcting algorithms, and explore the compatibility of these definitions with group-fairness ones.

References

[1] Agarwal, A., Beygelzimer, A., Dudík, M., Langford, J., and Wallach, H. (2018). A reductions approach to fair classification. *arXiv preprint arXiv:1803.02453*.

[2] Agarwal, A., Dudík, M., and Wu, Z. S. (2019). Fair regression: Quantitative definitions and reduction-based algorithms. In *International Conference on Machine Learning*, pages 120–129. PMLR.

[3] Andersson, F. O., Kaiser, R., and Jacobsson, S. P. (2004). Data preprocessing by wavelets and genetic algorithms for enhanced multivariate analysis of lc peptide mapping. *Journal of pharmaceutical and biomedical analysis*, 34(3):531–541.

[4] Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine bias. *ProPublica, May*, 23(2016):139–159.

[5] Asadollahi, T., Dadfarnia, S., Shabani, A. M. H., Ghasemi, J. B., and Sarkhosh, M. (2011). QSAR models for CXCR2 receptor antagonists based on the genetic algorithm for data preprocessing prior to application of the PLS linear regression method and design of the new compounds using in silico virtual screening. *Molecules*, 16(3):1928–1955.

[6] Aydin, O. U., Taha, A. A., Hilbert, A., Khalil, A. A., Galinovic, I., Fiebach, J. B., Frey, D., and Madai, V. I. (2021). On the usage of average hausdorff distance for segmentation performance assessment: hidden error when used for ranking. *European Radiology Experimental*, 5(1):1–7.

[7] Backurs, A., Indyk, P., Onak, K., Schieber, B., Vakilian, A., and Wagner, T. (2019). Scalable fair clustering. In *International Conference on Machine Learning*, pages 405–413.

[8] Bagdasaryan, E., Poursaeed, O., and Shmatikov, V. (2019). Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems*, pages 15479–15488.

[9] Bantilan, N. (2018). Themis-ml: A fairness-aware machine learning interface for end-to-end discrimination discovery and mitigation. *Journal of Technology in Human Services*, 36(1):15–30.

[10] Bellamy, R. K., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilović, A., et al. (2019). AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development*, 63(4/5):4–1.

References

[11] Berger-Tal, O., Nathan, J., Meron, E., and Saltz, D. (2014). The exploration-exploitation dilemma: a multidisciplinary framework. *PloS one*, 9(4):e95693.

[12] Berk, R., Heidari, H., Jabbari, S., Kearns, M., and Roth, A. (2018). Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, page 0049124118782533.

[13] Berrar, D. (2019). Cross-validation.

[14] Biddle, D. (2017). *Adverse impact and test validation: A practitioner's guide to valid and defensible employment testing*. Routledge.

[15] Branke, J., Kaußler, T., Smidt, C., and Schmeck, H. (2000). A multi-population approach to dynamic optimization problems. In *Evolutionary design and manufacture*, pages 299–307. Springer.

[16] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[17] Buckland, M. and Gey, F. (1994). The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19.

[18] Calders, T., Kamiran, F., and Pechenizkiy, M. (2009). Building classifiers with independency constraints. In *2009 IEEE International Conference on Data Mining Workshops*, pages 13–18. IEEE.

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

[20] Calmon, F., Wei, D., Vinzamuri, B., Natesan Ramamurthy, K., and Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. *Advances in Neural Information Processing Systems*, 30:3992–4001.

[21] Cason, T. E. (1999). Titanic Dataset. `https://biostat.app.vumc.org/wiki/Main/DataSets`. [Online; accessed 25-May-2021].

[22] Celis, L. E., Huang, L., Keswani, V., and Vishnoi, N. K. (2019). Classification with fairness constraints: A meta-algorithm with provable guarantees. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 319–328.

[23] Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.

[24] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[25] Chawla, N. V., Lazarevic, A., Hall, L. O., and Bowyer, K. W. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*, pages 107–119. Springer.

[26] Chen, I., Johansson, F. D., and Sontag, D. (2018). Why is my classifier discriminatory? *arXiv preprint arXiv:1805.12002*.

[27] Chiappa, S. (2019). Path-specific counterfactual fairness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33-01, pages 7801–7808.

[28] Chierichetti, F., Kumar, R., Lattanzi, S., and Vassilvitskii, S. (2018). Fair clustering through fairlets. *arXiv preprint arXiv:1802.05733*.

[29] Chouldechova, A. (2017). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163.

[30] Chowdhury, R. (2021). Sharing learnings about our image cropping algorithm. `https://blog.twitter.com/engineering/en_us/topics/insights/2021/sharing-learnings-about-our-image-cropping-algorithm`.

[31] Cooper, A. F., Abrams, E., and NA, N. (2021). Emergent unfairness in algorithmic fairness-accuracy trade-off research. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 46–54.

[32] Crisp, R. (2014). *Aristotle: Nicomachean Ethics*. Cambridge University Press.

[33] Crone, S. F., Lessmann, S., and Stahlbock, R. (2006). The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research*, 173(3):781–800.

[34] Crookston, N. L. and Finley, A. O. (2008). yaimpute: an r package for knn imputation. *Journal of Statistical Software. 23 (10). 16 p.*

[35] Cummings, R., Gupta, V., Kimpara, D., and Morgenstern, J. (2019). On the compatibility of privacy and fairness. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, pages 309–315.

[36] Danks, D. and London, A. J. (2017). Algorithmic bias in autonomous systems. In *IJCAI*, volume 17, pages 4691–4697.

[37] Dastin, J. (2018). Amazon scraps secret AI recruiting tool that showed bias against women. `https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G`.

[38] Datta, A., Sen, S., and Tschantz, M. C. (2018). Correspondences between privacy and nondiscrimination: why they should be studied together. *arXiv preprint arXiv:1808.01735*.

[39] Deb, K. (2002). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-2. *IEEE Trans. Evol. Comput.*, 6(2):182–197.

[40] Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., et al. (2013). Orange: data mining toolbox in python. *the Journal of machine Learning research*, 14(1):2349–2353.

[41] Dieterich, W., Mendoza, C., and Brennan, T. (2016). Compas risk scales: Demonstrating accuracy equity and predictive parity. *Northpointe Inc*.

[42] Domingo-Ferrer, J. and Torra, V. (2005). Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *Data Mining and Knowledge Discovery*, 11(2):195–212.

[43] Dressel, J. and Farid, H. (2018). The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 4(1):eaao5580.

[44] Dua, D. and Graff, C. (2017). UCI machine learning repository.

[45] Dwork, C. (2006). Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer.

[46] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. (2012). Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 214–226, New York, NY, USA. Association for Computing Machinery.

[47] Ekstrand, M. D., Joshaghani, R., and Mehrpouyan, H. (2018). Privacy for all: Ensuring fair and equitable privacy protections. In Friedler, S. A. and Wilson, C., editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 35–47, New York, NY, USA. PMLR.

[48] Equivant (2019). Practitioner's guide to COMPAS Core. [Online; accessed 06-10-2021].

[49] Farrar, D. E. and Glauber, R. R. (1967). Multicollinearity in regression analysis: the problem revisited. *The Review of Economic and Statistics*, pages 92–107.

[50] Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268.

[51] Fernando, M.-P., César, F., David, N., and José, H.-O. (2021). Missing the missing values: The ugly duckling of fairness in machine learning. *International Journal of Intelligent Systems*, 36(7):3217–3258.

[52] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., and Hutter, F. (2019). Auto-sklearn: efficient and robust automated machine learning. In *Automated Machine Learning*, pages 113–134. Springer, Cham.

[53] Fitkov-Norris, E., Vahid, S., and Hand, C. (2012). Evaluating the impact of categorical data encoding and scaling on neural network classification performance: the case of repeat consumption of identical cultural goods. In *International Conference on Engineering Applications of Neural Networks*, pages 343–352. Springer.

[54] Foulds, J. R., Islam, R., Keya, K. N., and Pan, S. (2020). An intersectional definition of fairness. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1918–1921. IEEE.

[55] Friedler, S. A., Scheidegger, C., and Venkatasubramanian, S. (2021). The (im) possibility of fairness: different value systems require different mechanisms for fair decision making. *Communications of the ACM*, 64(4):136–143.

[56] Friedler, S. A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E. P., and Roth, D. (2019). A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 329–338. ACM.

[57] Gajane, P. and Pechenizkiy, M. (2017). On formalizing fairness in prediction with machine learning. *arXiv preprint arXiv:1710.03184*.

[58] García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M., and Herrera, F. (2016). Big data preprocessing: methods and prospects. *Big Data Analytics*, 1(1):9. ISBN: 2058-6345.

[59] Garg, S., Perot, V., Limtiaco, N., Taly, A., Chi, E. H., and Beutel, A. (2019). Counterfactual fairness in text classification through robustness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 219–226.

[60] Gold, J. I. and Shadlen, M. N. (2002). Banburismus and the brain: decoding the relationship between sensory stimuli, decisions, and reward. *Neuron*, 36(2):299–308.

[61] González, J., Osborne, M., and Lawrence, N. D. (2016). Glasses: Relieving the myopia of bayesian optimisation. *Journal of Machine Learning Research*.

[62] González-Zelaya, V. (2019). Towards explaining the effects of data preprocessing on machine learning. *2019 IEEE 35th International Conference on Data Engineering (ICDE)*.

[63] González-Zelaya, V., Missier, P., and Prangle, D. (2019). Parametrised Data Sampling for Fairness Optimisation. Presented on the *2019 XAI Workshop at SIGKDD, Anchorage, AK, USA*.

[64] González-Zelaya, V., Salas, J., Megías, D., and Missier, P. (2021a). Fair Classification with Privacy Guarantees. Submitted to the *Data Mining and Knowledge Discovery (DAMI)* Journal.

[65] González-Zelaya, V., Salas, J., Prangle, D., and Missier, P. (2021b). Optimising Fairness Through Parametrised Data Sampling. In *Proceedings of the 2021 EDBT Conference*.

[66] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[67] Gunn, S. R. et al. (1998). Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16.

[68] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

[69] Hajian, S., Domingo-Ferrer, J., Monreale, A., Pedreschi, D., and Giannotti, F. (2015). Discrimination- and privacy-aware patterns. *Data Min. Knowl. Discov.*, 29(6):1733–1782.

[70] Hamilton, I. A. (2019). Apple cofounder Steve Wozniak says Apple Card offered his wife a lower credit limit. https://www.businessinsider.com/apple-card-sexism-steve-wozniak-2019-11.

[71] Hao, K. (2019). Police across the US are training crime-predicting AIs on falsified data. https://www.technologyreview.com/2019/02/13/137444/predictive-policing-algorithms-ai-crime-dirty-data/.

[72] Hardt, M., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323.

[73] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with numpy. *Nature*, 585(7825):357–362.

[74] Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., and Prasath, V. (2019). Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, 10(12):390.

[75] He, H. and Ma, Y. (2013). *Imbalanced learning: foundations, algorithms, and applications*. Wiley-IEEE Press.

[76] Hébert-Johnson, Ú., Kim, M., Reingold, O., and Rothblum, G. (2018). Multicalibration: Calibration for the (computationally-identifiable) masses. In *International Conference on Machine Learning*, pages 1939–1948.

[77] Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

[78] Hooker, S. (2021). Moving beyond "algorithmic bias is a data problem". *Patterns*, 2(4):100241.

[79] Hosmer Jr, D. W., Lemeshow, S., and Sturdivant, R. X. (2013). *Applied logistic regression*, volume 398. John Wiley & Sons.

[80] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95.

[81] Iosifidis, V. and Ntoutsi, E. (2019). Adafair: Cumulative fairness adaptive boosting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 781–790.

[82] Jagielski, M., Kearns, M. A., Sharifi-Malvajerdi, S., Mao, J., Oprea, A., Roth, A., and Ullman, J. (2019). Differentially private fair learning. In *International Conference on Machine Learning*, pages 3000–3008. PMLR.

[83] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

[84] Janssen, K. J., Donders, A. R. T., Harrell Jr, F. E., Vergouwe, Y., Chen, Q., Grobbee, D. E., and Moons, K. G. (2010). Missing covariate data in medical research: to impute is better than to ignore. *Journal of clinical epidemiology*, 63(7):721–727.

[85] Jiang, T., Gradus, J. L., and Rosellini, A. J. (2020). Supervised machine learning: a brief primer. *Behavior Therapy*, 51(5):675–687.

[86] Johnson, K. (2021). Twitter's Photo Crop Algorithm Favors White Faces and Women. `https://www.wired.com/story/twitter-photo-crop-algorithm-favors-white-faces-women/`.

[87] Jung, C., Kearns, M., Neel, S., Roth, A., Stapleton, L., and Wu, Z. S. (2019). Eliciting and enforcing subjective individual fairness. *arXiv e-prints*, pages arXiv–1905.

[88] Justesen, P. D. (2009). Multi-objective optimization using evolutionary algorithms. *University of Aarhus, Department of Computer Science, Denmark*, 33.

[89] Kamiran, F. and Calders, T. (2010). Classification with no discrimination by preferential sampling. In *Proc. 19th Machine Learning Conf. Belgium and The Netherlands*, pages 1–6. Citeseer.

[90] Kamiran, F. and Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33.

[91] Kamishima, T., Akaho, S., Asoh, H., and Sakuma, J. (2012). Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer.

[92] Kayser-Bril, N. (2020). Female historians and male nurses do not exist, Google Translate tells its European users. `https://algorithmwatch.org/en/google-translate-gender-bias/`.

[93] Kearns, M., Neel, S., Roth, A., and Wu, Z. S. (2018). Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning*, pages 2564–2572.

[94] Kilbertus, N., Carulla, M. R., Parascandolo, G., Hardt, M., Janzing, D., and Schölkopf, B. (2017). Avoiding discrimination through causal reasoning. In *Advances in Neural Information Processing Systems*, pages 656–666.

[95] Kirkpatrick, K. (2017). It's not the algorithm, it's the data. *Communications of the ACM*, 60(2):21–23.

[96] Kotsiantis, S., Kanellopoulos, D., and Pintelas, P. (2006). Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117.

[97] Krasanakis, E., Spyromitros-Xioufis, E., Papadopoulos, S., and Kompatsiaris, Y. (2018). Adaptive sensitive reweighting to mitigate bias in fairness-aware classification. In *Proceedings of the 2018 World Wide Web Conference*, pages 853–862.

[98] Kubat, M., Matwin, S., et al. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, pages 179–186. Citeseer.

[99] Kuczmarski, J. (2018). Reducing gender bias in Google Translate. `https://blog.google/products/translate/reducing-gender-bias-google-translate/`.

[100] Kumar, M., Husain, M., Upreti, N., and Gupta, D. (2010). Genetic algorithm: Review and application. *Available at SSRN 3529843*.

[101] Kusner, M., Loftus, J., Russell, C., and Silva, R. (2017). Counterfactual fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4069–4079.

[102] Lahoti, P., Gummadi, K. P., and Weikum, G. (2019). ifair: Learning individually fair data representations for algorithmic decision making. In *2019 ieee 35th international conference on data engineering (icde)*, pages 1334–1345. IEEE.

[103] Lakkaraju, H., Kleinberg, J., Leskovec, J., Ludwig, J., and Mullainathan, S. (2017). The selective labels problem: Evaluating algorithmic predictions in the presence of unobservables. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 275–284.

[104] Lakshminarayan, K., Harp, S. A., and Samad, T. (1999). Imputation of missing data in industrial databases. *Applied intelligence*, 11(3):259–275.

[105] Larionov, M. (2020). Sampling techniques in bayesian target encoding. *arXiv preprint arXiv:2006.01317*.

[106] Larson, J., Mattu, S., Kirchner, L., and Angwin, J. (2016). How we analyzed the compas recidivism algorithm. *ProPublica (5 2016)*, 9.

[107] Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 63–66. Springer.

[108] LeDell, E. and Poirier, S. (2020). H2o automl: Scalable automatic machine learning. In *7th ICML workshop on automated machine learning*.

[109] Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, 18(1):559–563.

[110] Li, N., Li, T., and Venkatasubramanian, S. (2007). t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115.

[111] Madras, D., Creager, E., Pitassi, T., and Zemel, R. (2019). Fairness through causal awareness: Learning causal latent-variable models for biased data. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 349–358.

[112] Mazilu, L., Konstantinou, N., Paton, N. W., and Fernandes, A. A. (2021). Data wrangling for fair classification. In *EDBT/ICDT Workshops*.

[113] McGinnis, W. D., Siu, C., Andre, S., and Huang, H. (2018). Category encoders: a scikit-learn-contrib package of transformers for encoding categorical data. *Journal of Open Source Software*, 3(21):501.

[114] McKinney, W. et al. (2011). pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9.

[115] Meier, P., Sacks, J., and Zabell, S. L. (1984). What happened in hazelwood: Statistics, employment discrimination, and the 80% rule. *American Bar Foundation Research Journal*, 9(1):139–186.

[116] Menon, A. K. and Williamson, R. C. (2018). The cost of fairness in binary classification. In *Conference on Fairness, Accountability and Transparency*, pages 107–118. PMLR.

[117] Micci-Barreca, D. (2001). A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, 3(1):27–32.

[118] Mitchell, S., Potash, E., Barocas, S., D'Amour, A., and Lum, K. (2021). Algorithmic fairness: Choices, assumptions, and definitions. *Annual Review of Statistics and Its Application*, 8:141–163.

[119] Moore, A. W. (2001). Cross-validation for detecting and preventing overfitting. *School of Computer Science Carneigie Mellon University*.

[120] Nakagawa, S. and Freckleton, R. P. (2008). Missing inaction: the dangers of ignoring missing data. *Trends in ecology & evolution*, 23(11):592–596.

[121] Ngatchou, P., Zarei, A., and El-Sharkawi, A. (2005). Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pages 84–91. IEEE.

[122] Olson, R. S. and Moore, J. H. (2016). Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pages 66–74. PMLR.

[123] Oreski, S., Oreski, D., and Oreski, G. (2012). Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment. *Expert systems with applications*, 39(16):12605–12617.

[124] Patro, S. and Sahu, K. K. (2015). Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*.

[125] Pearl, J. (2009). *Causality*. Cambridge university press.

[126] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

[127] Phillips, A. (2004). Defending equality of outcome. *Journal of political philosophy*, 12(1):1–19.

[128] Pujol, D., McKenna, R., Kuppam, S., Hay, M., Machanavajjhala, A., and Miklau, G. (2020). Fair decision making using privacy-protected data. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 189–199, New York, NY, USA. Association for Computing Machinery.

[129] Pyle, D. (1999). *Data preparation for data mining*. morgan kaufmann.

[130] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

[131] Rodríguez-Gálvez, B., Thobaben, R., and Skoglund, M. (2020). A variational approach to privacy and fairness. *arXiv*, pages arXiv–2006.

[132] Rubin, D. B. (1973). The use of matched sampling and regression adjustment to remove bias in observational studies. *Biometrics*, pages 185–203.

[133] Saito, K. (2013). Social preferences under risk: Equality of opportunity versus equality of outcome. *American Economic Review*, 103(7):3084–3101.

[134] Salas, J. and Domingo-Ferrer, J. (2018). Some basics on privacy techniques, anonymization and their big data challenges. *Mathematics in Computer Science*, 12(3):263–274.

[135] Salas, J. and González-Zelaya, V. (2020). Fair-MDAV: An algorithm for fair privacy by microaggregation. In *Modeling Decisions for Artificial Intelligence. MDAI 2020*, volume 12256. Springer.

[136] Salazar, R., Neutatz, F., and Abedjan, Z. (2021). Automated feature engineering for algorithmic fairness. *PROCEEDINGS OF THE VLDB ENDOWMENT*, 14(9):1694–1702.

[137] Saleiro, P., Kuester, B., Hinkson, L., London, J., Stevens, A., Anisfeld, A., Rodolfa, K. T., and Ghani, R. (2018). Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577*.

[138] Salimi, B., Rodriguez, L., Howe, B., and Suciu, D. (2019). Interventional fairness: Causal database repair for algorithmic fairness. In *Proceedings of the 2019 International Conference on Management of Data*, pages 793–810.

[139] Samarati, P. (2001). Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027.

[140] Schapire, R. E. (2013). Explaining adaboost. In *Empirical inference*, pages 37–52. Springer.

[141] Schutze, O., Esquivel, X., Lara, A., and Coello, C. A. C. (2012). Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 16(4):504–522.

[142] Sci-kit Learn Developers (2019). scikit-learn: machine learning in python.

[143] Seger, C. (2018). An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing.

[144] Singh, D. and Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97:105524.

[145] Smith, M. J., Sala, C., Kanter, J. M., and Veeramachaneni, K. (2020). The machine learning bazaar: Harnessing the ml ecosystem for effective system development. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 785–800.

[146] Stoyanovich, J., Howe, B., and Jagadish, H. (2020). Responsible data management. *Proceedings of the VLDB Endowment*, 13(12):3474–3488.

[147] Stoyanovich, J., Howe, B., Jagadish, H., and Miklau, G. (2018). Panel: a debate on data and algorithmic ethics. *Proceedings of the VLDB Endowment*, 11(12):2165–2167.

[148] Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

[149] Tan, F., Fu, X., Zhang, Y., and Bourgeois, A. G. (2008). A genetic algorithm-based method for feature subset selection. *Soft Computing*, 12(2):111–120.

[150] Tax, D. M. and Duin, R. P. (2002). Using two-class classifiers for multiclass classification. In *Object recognition supported by user interaction for service robots*, volume 2, pages 124–127. IEEE.

[151] The GPyOpt authors (2016). GPyOpt: A Bayesian Optimization framework in Python. `http://github.com/SheffieldML/GPyOpt`.

[152] Toman, M., Tesar, R., and Jezek, K. (2006). Influence of word normalization on text classification. *Proceedings of InSciT*, 4:354–358.

[153] Tsai, C.-F., Eberle, W., and Chu, C.-Y. (2013). Genetic algorithms in feature and instance selection. *Knowledge-Based Systems*, 39:240–247.

[154] Tsamardinos, I. and Aliferis, C. F. (2003). Towards principled feature selection: Relevancy, filters and wrappers. In *International Workshop on Artificial Intelligence and Statistics*, pages 300–307. PMLR.

[155] Uysal, A. K. and Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112.

[156] Van Buuren, S. (2018). *Flexible imputation of missing data*. CRC press.

[157] Van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45(1):1–67.

[158] Vijayakumar, M. and Prabhakar, E. (2018). A hybrid combined under-over sampling method for class imbalanced datasets. *International Journal of Research and Advanced Development (IJRAD)*, 2(05):27–33.

[159] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.

[160] Voigt, P. and Von dem Bussche, A. (2017). The EU General Data Protection Regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676.

[161] Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.

[162] Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85.

[163] Wu, Y., Zhang, L., and Wu, X. (2019). Counterfactual fairness: Unidentification, bound and algorithm. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*.

[164] Xu, D., Yuan, S., and Wu, X. (2019). Achieving differential privacy and fairness in logistic regression. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 594–599, New York, NY, USA. Association for Computing Machinery.

[165] Yang, K., Huang, B., Stoyanovich, J., and Schelter, S. (2020). Fairness-aware instrumentation of preprocessing pipelines for machine learning. In *Workshop on Human-In-the-Loop Data Analytics (HILDA'20)*.

[166] Yoo, S. and Harman, M. (2007). Pareto efficient multi-objective test case selection. In *Proceedings of the 2007 international symposium on Software testing and analysis*, pages 140–150.

[167] Zafar, M. B., Valera, I., Gomez Rodriguez, M., and Gummadi, K. P. (2017). Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th international conference on world wide web*, pages 1171–1180.

[168] Zafar, M. B., Valera, I., Rodriguez, M. G., and Gummadi, K. P. (2015). Fairness constraints: Mechanisms for fair classification. *arXiv preprint arXiv:1507.05259*.

[169] Zehlike, M., Castillo, C., Bonchi, F., Hajian, S., and Megahed, M. (2017). Fairness measures: Datasets and software for detecting algorithmic discrimination. *URL http://fairness-measures. org*.

[170] Zemel, R., Wu, Y., Swersky, K., Pitassi, T., and Dwork, C. (2013). Learning fair representations. In *International Conference on Machine Learning*, pages 325–333.

[171] Zhang, Z. and Neill, D. B. (2016). Identifying significant predictive bias in classifiers. *arXiv preprint arXiv:1611.08292*.

[172] Zhao, H. and Gordon, G. (2019). Inherent tradeoffs in learning fair representations. *Advances in neural information processing systems*, 32:15675–15685.

Appendix A

Reproducibility

Contents

A.1 Software Requirements

Our algorithms were written and run in Jupyter Lab 2.2.6 using a Python 3.8.5 kernel on an Ubuntu Linux 20.04 system. The following packages are required to use our notebooks:

pandas 1.1.3 Data analysis and manipulation through data frames [114].

NumPy 1.19.2 Scientific computing [73].

scikit-learn 0.24.1 Predictive data analysis [126].

imbalanced-learn 0.8.0 Tools for classification with imbalanced classes [109].

GPyOpt 1.2.6 Gaussian process optimisation [151].

Matplotlib 3.3.2 Highly customisable visualisations [80].

Seaborn 0.11.0 Simplified plotting on top of Matplotlib [161].

A.2 Data

We have performed our analyses on four benchmark datasets: *Adult Income*, *COMPAS*, *German Credit* and *Titanic*. The original CSV files may be obtained from the following URLs, as well as from our repository:

Adult Income
`https://archive.ics.uci.edu/ml/datasets/census+income`

COMPAS
`https://github.com/propublica/compas-analysis`

German Credit
`https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`

Titanic
`http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.html`

A.3 Scripts

FAIRPIPES

This repository includes two Jupyter Notebooks, as well as all of the analysed unprocessed datasets. It may be found at

`https://github.com/vladoxNCL/fairPipes`

The relevant notebooks in this repository are:

Gen_Alg.ipynb The main genetic algorithm. It may be easily modified to support additional binary classification datasets with a binary PA.

Plots.ipynb This script contains functions to find the estimated Pareto front of a particular FAIRPIPES run, compare it with the ground truth Pareto front if the necessary data is available, and generate a collection of plots for both fairness and performance.

PARDS

The repository includes both of the scripts used for our analyses, as well as all of the datasets in both their original and cleaned-up versions. It may be found at

`https://github.com/vladoxNCL/fairCorrect`

The two notebooks in this repository are:

data_cleanup.ipynb Helper notebook, used to convert the original data files into a clean and one-hot encoded version, suitable for data analysis.

Fairness-Multi.ipynb The main notebook. In the first half, the correction algorithms are coded. The user needs to specify the name of the desired dataset the scripts will be run over in the second code block, by setting the `dset` variable to the appropriate string value (`Income` by default). The second half generates most of the figures in the paper. Some of the generated plots are dataset-specific, hence the `dset` variable should be set according to the dataset to be analysed. For both scripts, all the *savefile* commands have been commented out, as the savepath needs to be specified by the user.

FAIR-MDAV

This repository includes three Jupyter Notebooks, as well as all of the analysed datasets. It may be found at

`https://github.com/vladoxNCL/fairMDAVplus`

The relevant notebooks in this repository are:

fMDAV.ipynb The main Fair-MDAV algorithm script. Will load, microaggregate and correct the fairness of a list of datasets.

Metrics.ipynb Auxiliary script to generate the metrics of our analyses.

Plots.ipynb Auxiliary script to generate plots.