eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Degree in Computer Science
Computation

Final Degree Thesis

Systems to decrease gender bias in classifiers

Author

Oier Irazabal Urrutia

2020

eman ta zabal zazu

Universidad del País Vasco   Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Degree in Computer Science

Computation

Final Degree Thesis

Systems to decrease gender bias in classifiers

Author

Oier Irazabal Urrutia

Director

Olatz Arbelaitz Gallego

Abstract

It is said that with great power comes great responsibility. Nowadays, we rely on machine learning systems that are capable of understanding text at a human-like level. Yet, relations like "man is to computer scientist what woman is to homemaker" are present in these systems.

The importance of the topic and the effect it has in the society has made it become an important research topic during the last years giving rise to different solutions.

In this work, we describe some state-of-the-art techniques that reduce gender bias in machine learning algorithms as well as assess their results employing fairness evaluation metrics.

Contents

List of Figures

List of Tables

1. CHAPTER

Introduction

Artificial Inteligence (AI) is having an increasing impact on our lives. Not only do AI systems predict the weather or perform spam filtering [Zafar et al., 2015], but also they are able to evaluate the risk of recidivism in individuals [Celis et al., 2019] or decide whether a person is worth lending credits [Dwork et al., 2017].

These systems are backed up by machine learning algorithms that are trained over large datasets and it has been shown that biased datasets entail biased systems, as well-trained models will reflect those biases [Zhang et al., 2018]. Without proper identification and reduction of such correlations, models will inadvertently magnify stereotypes [Zhao et al., 2017].

Many articles have shown concern about unfairly treated minorities in some datasets, and in turn, in some classifiers. For example, against women when applying for jobs or against African-Americans for predicting future criminals [Celis et al., 2019].

Fortunately, a wide range of solutions have already been proposed in different classification tasks ranging from creating new "neutral" datasets [Buolamwini and Gebru, 2018] or placing an adversary to the classifier that penalizes the predictability of gender/race based on the outcome of the classifier [Zhang et al., 2018].

Experts in the Natural Language Processing (NLP) field have also shown a deep concern because word embeddings (a vector representation for words) have produced relationships like "man is to programmer what woman is to homemaker" [Bolukbasi et al., 2016].

Besides, a debate resides in the community of fairness in classification on whether sensitive data such as gender, race, or religion should be used alongside other data at the training and classification stages. Moreover, the legality and ethics of using those attributes vary depending on country, jurisdiction, etc [Dwork et al., 2017].

One might be naively tempted to think that excluding sensitive data such as gender, race, or religion on corpora might eliminate the imbalance across different groups. However, this sense of "fairness through unawareness" is ineffective due to redundancy and correlation between attributes, enabling the classifier predict sensitive attributes through other piece of information [Hardt et al., 2016]. For instance, having been in the military is strongly more correlated to men than women. According to the data in `https://www.cfr.org/article/demographics-us-military`, no service at the US military (Air Force, Navy, Marine and Army) saw a proportion of women reach 20% in 2016.

Researchers are investigating various methods that may or may not directly use this attribute for training. In our case, some of the applied techniques will not only incorporate the sensitive attribute to the classification data, but they will treat them as special values, crucial for the correct performance of the models.

2. CHAPTER

Document of the Project Objectives

The aim of the project is to evaluate the impact of different state-of-the-art debiasing techniques to reduce gender bias in classification tasks. First of all, we are going to examine the bibliography and gather different techniques and practices. Then, we will choose some techniques from a group of proposed methods that rely on a different foundation from one another and analyze how they can improve fairness to different classifiers.

In order to quantify and measure the improvement, we will select the metrics that introduce a distinct notion of fairness that best fit our work. Depending on the setting, one notion may be more suitable than another, so having different metrics will allow our work to cover more potential ground.

The datasets used in the bibliography are going to be used for conducting the experiments if possible. If few datasets are available or unusable for our work, additional ones may be sourced from online database collections. This way we will ensure the validity of the techniques in multiple possible situations.

2.1 Planning

Unlike most software engineering projects, where the aim of the project is the final product and the development of it can be more precisely forecast by identifying different parts of the code and dependencies, our work is dependent on the usefulness of various state of the art techniques, which may only be fully understood once implemented and tested. That is the reason why planning has been more general and developed in an implementation-independent approach. The decomposition of the work and the planned spans for each task are recorded in figure 2.1 and table 2.1.

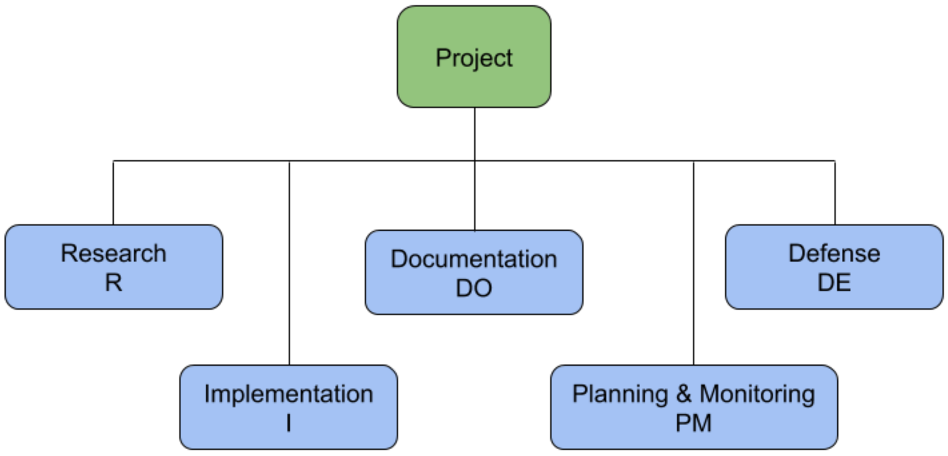


Figure 2.1: Structure of the workload breakdown

2.1.1 Risks and countermeasures

R1. Losing Code.

There are countless factors that can lead to catastrophic progress loss due to the code going missing, ranging from a malfunctioning disk to accidental deletion of files. We will also extend this risk to intermediary files such as pre-processed databases and even raw databases, although the latter can be retrieved from the Internet publicly.

In order to mitigate this issue, we will back up a folder containing every piece of data related to the project on an online storage service. The chosen service has been Google Drive, as it is widely used and user-friendly that allows sharing files to other Google users, useful for the director to immediately access them. Submitting the code to a remote git repository was also considered, but as version control is not a necessity and only one member will develop all the code, saving the progress after to the online backup will be enough.

R2. Losing the computer.

[1] An initial collection of paper was provided by the director.

Code	Task	Estimation
PM1	Planning the project	5h
PM2	Monitoring the development	5h
R1	Finding additional bibliography[1]	10h
R2	Researching over the contents in the papers	20h
R3	Selection of techniques that fit our work	10h
R4	Understanding the core functionality of the techniques	15h
R4	Gathering databases used in the bibliography	2h
R5	Gathering databases outside the bibliography	5h
R6	Researching additional bibliography for the report	25h
I1	Pre-processing databases to a common standard	10h
I2	Implementing fairness metrics	5h
I3	Implementing debiasing techniques	35h
I4	Designing and running the experiments	15h
I5	Testing and debugging	10h
DO1	Documenting useful concepts in each paper	20h
DO2	Extracting and analyzing results from I4	10h
DO3	Writing the report of the work	70h
DE1	Developing the slideshow for the defense	13h
DE2	Preparing the oral defense	15h
–	**Total**	**300h**

Table 2.1: Tasks and their planned time span.

Despite being highly unlikely, losing the computer would not only mean losing all data that had not been backed up, but also losing the working environment, constituted of the operating system, installed software, downloaded packages, and so on. Getting another computer to the same state as the previous could be painstakingly tedious.

Unfortunately, there is no viable remedy to this issue aside from the one in the previous risk, but to keep track of the dependencies the project requires, such as compilers, interpreters and working environments.

R3. Not being able to meet with the director.

Due to me living in another province relatively far from the faculty, and having to commute to there everyday, reduced time is left for when the director and myself are both free to meet. Having said that, the first two meetings have been successful and communication by e-mail has been complementary to face-to-face meetings.

This issue can be addressed by arranging virtual meetings, which could limit the ease of sharing and modifying documents, making progress more time-consuming, but can also be complemented by e-mail exchanging in the same way as in-person meetings.

R4. Delaying work.

As stated previously, spending significant time commuting every day could result in losing valuable time for any work, related or not to this project. The worst-case scenario would be where other personal or academic affairs take up all the time of the day and this project gets its work postponed.

In order to decrease the consequences of this risk, two measures can be applied:

1. Planning ahead the work according to the direction the project is taking.
2. Distributing the workload as evenly as possible across multiple days instead of concentrating it in bursts.

2.2 Monitoring

All in all, the development of the project has been smooth throughout the first half but has taken a big hit on the second half due to the consequences of the covid-19 pandemic.

The pandemic brought the risk R3 to its full extent, because the faculty remained closed for students since the second week of March. Also, the incapability of properly taking exams made professors increase workload during the term, which resulted in me reducing the dedication to this project (R4).

However, the drop of work at the beginning of the second half was compensated by increasing the dedication towards the end. This action, though risky, ended successfully and the proposed goals have been achieved.

2.2.1 Deviation from the plan

The following table shows how long each task actually took and compares the time spans to the planned times. Actual spans for tasks DE1 and DE2 have not been yet recorded as they are in progress, so we have added the estimated time to the total in brown, since they are advancing as planned.

Code	**Estimation**	**Actual**	**Deviation**
PM1	5h	5h	+0h
PM2	5h	5h	+0h
R1	10h	10h	+0h
R2	20h	20h	+0h
R3	10h	12h	+2h
R4	15h	15h	+0h
R4	2h	2h	+0h
R5	5h	2h	-3h
R6	25h	25h	+0h
I1	10h	8h	-2h
I2	5h	3h	-2h
I3	35h	30h	-5h
I4	15h	25h	+10h
I5	10h	15h	+5h
DO1	20h	15h	-5h
DO2	10h	7h	-3h
DO3	70h	80h	+10h
DE1	13h	13h	+0h
DE2	15h	15h	+0h
Total	**300h**	**304h**	**+4h**

Table 2.2: Tasks and their real time span.

3. CHAPTER

Theoretical concepts

3.1 Classification problems

Consider a scenario where we need to build a system to classify incoming user data as deciding whether a person is worth lending money to. Therefore, we will collect diverse data about the individual such as salary, marital status, bank account balance and so on. We will represent that piece of data as $(x_1, x_2, \ldots, x_d) = \bar{x}$.

What we seek is a mapping that outputs *no* or *yes* that can be represented as $z \in \{0,1\}$ when taking in the data of a given individual. Although classification problems can be generalized to outputting any value as z, we will narrow our classification tasks down to binary problems where $z = 0$ when the outcome is negative (e.g. credit not granted) and $z = 1$ when positive (credit granted). Mathematically speaking, we seek the function $f : \mathbb{R}^d \to \{0,1\}$ where $f(\bar{x}) = z$.

Which f is best is problem-specific and usually require iterative algorithms that rely on high computational power.

3.2 Supervised Learning

In supervised learning tasks, f is inferred from pre-labelled samples allegedly resembling the distribution of the entire dataset. Continuing the previous example, a bank may use data from previous credit requests classified by humans, where $\bar{x}$ is the data of the requester and y whether he/she was granted the credit (the label).

The collection of labelled samples is called the training set and algorithms use that data to optimize the problem. The set is mathematically represented as
$\mathcal{S} = \{(\bar{x}_1, y_1), (\bar{x}_2, y_2), \dots, (\bar{x}_n, y_n)\} = \mathcal{X} \times \mathcal{Y}$.

These algorithms aim to fit the mapping as close as possible to the annotated data by predicting z, comparing it to y and self-adjusting according to the comparison. This process is performed until a stopping criterion is met (acceptable error-rate, running out of data, time spent, ...). The ideal model would output the same value z as the label y for every $\bar{x}$, as generally, accuracy (the amount of correct predictions) is used to measure the performance of the model.

Before we move onto explaining the different models used in this project, we will first determine the nomenclature and then explain a simple classifier. The terminology will be the following:

Notation	**Meaning**	**Set**
x_i	sample data	$\mathcal{X}$
y_i	label (real class)	$\mathcal{Y}$
z_i	prediction (guessed class)	$\mathcal{Z}$
a_i	protected value	$\mathcal{A}$

Table 3.1: Terminology used throughout the document.

The respective upper-case letters of each term in the table indicates its collection or set such that $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$. Note that even though it is not written with a bar as in previous mentions, x is a vector and x_i will generally indicate the ith vector in the dataset, not the ith element in x.

3.2.1 Linear Classifiers

One of the most visually intuitive models are linear classifiers. For the sake of simplicity, we will demonstrate examples of planar (2D) classifiers, even though this can be generalized to n-dimensional spaces. The idea is to draw a line that divides the plane into two regions such that the upper will belong to one class and the lower to the other.

Following the equation $t = f(x_1, x_2) = a \cdot x_1 + b \cdot x_2 + c$, the model will have to find the best values for parameters a, b and c so that the most possible cases are correctly classified.

As t is a real number, we will define the prediction of the model as $z = [t > 0]$, where $[\,]$ is the indicator function[1].

Keeping the same example, suppose x_1 is the salary and x_2 the amount of years in jail of the credit requester. Considering that a and b are the weights of the input variables, we want a to be positive, so that it scales according to x_1 and b to be very negative, as each year in prison should have a stronger effect on the output than each monetary unit earned per month.

Figure 3.1 shows a linear classifier that separates two clusters of data defined by colour. The data has been acquired from the Iris dataset provided by the Python library scikit-learn. The challenge is to distinguish between two types of flower types: *iris versicolour* and *iris virginica*. We can observe that the classifier struggles to correctly classify the three orange points below the line.

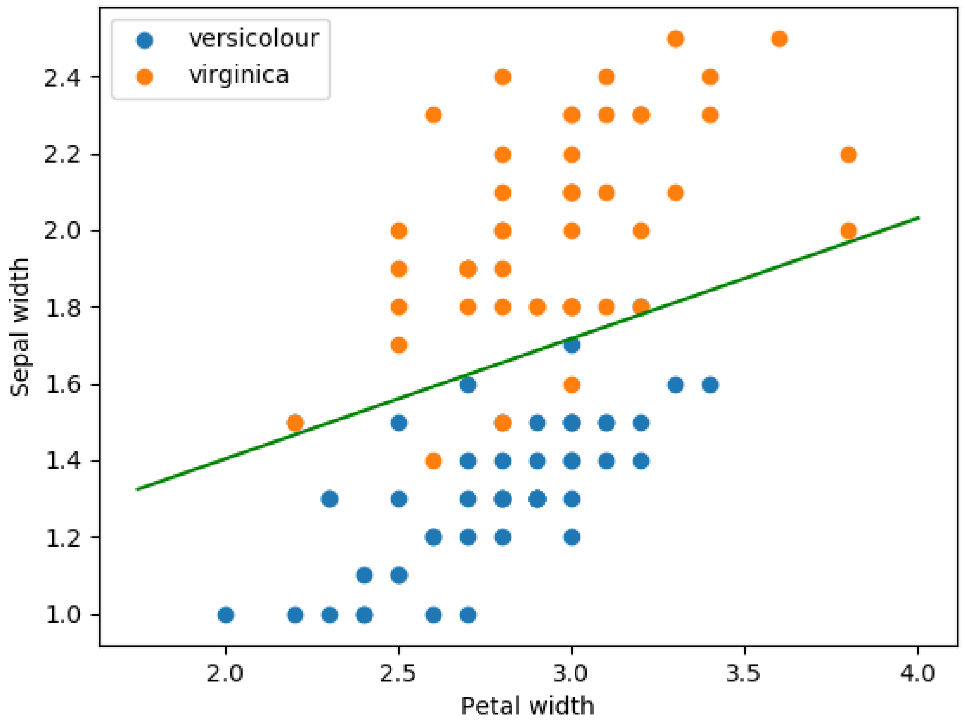


Figure 3.1: Planar linear classification example.

Due to its simplicity, this model is just capable of solving linearly separable problems,

[1]The indicator function returns 1 for elements that fulfill the boolean condition inside the square brackets and 0 for those that do not.

which means that there must exists a combination of a, b and c such that all cases are correctly classified, achieving perfect accuracy. Real world problems as the one in the previous figure, however, are almost always not linearly separable, and that is why we will use the following and more complex models.

3.2.2 Logistic Regression

Despite its name, Logistic Regression is a commonly used classification algorithm that models the distribution of probabilities of class y given feature vector x [Lee et al., 2006]. Due to the propensity of overfitting, where the classifier performs outstandingly well on training data but then underperforms on new data, regularization is often applied [Liu et al., 2009].

Regularization is an extra term added to the loss function (see section 3.2.3, but for now we can name it error function) of the classifier that penalizes large weights [Koh et al., 2007]. The parameter λ is introduced to control the tradeoff between regularization and the error. For example, L2 regularization controlled by lambda is shown in the following formula:

$$\min_{w} error(w) + \lambda \|w\|^2 \tag{3.1}$$

Regularization is required to avoid overfitting specially when there is a limited number of training examples or when there are multiple parameters to be learned [Lee et al., 2006].

3.2.3 Multilayer Perceptron

Artificial Neural Network

Artificial Neural Networks began in the 1950s, emerged in the 1980s and are still utilized [Daniel, 2013]. Their widespread use and success comes from the effectiveness in a variety of fields like medicine, finance and security [Daniel, 2013], and for its high computation rate due to their massive parallelism [Pal and Mitra, 1992]. In fact, there are tailored networks for different purposes, which consist of different types of neurons. There is an extensive list of different architectures in the following address:

https://www.asimovinstitute.org/neural-network-zoo/

Designed to mimic the human brain to emulate human intelligence [Pal and Mitra, 1992], artificial neural networks are composed of vertices and edges that act as neurons and axons in biological neural networks, respectively. The set of vertices and edges can be viewed either as a directed graph or as a function.

See figures 3.3 and 3.4 for illustrative examples of the perceptron, a single-layered network, and a more complex multilayer network.

Each neuron computes a weighted sum of its inputs, adds a real-valued bias and applies a non-linear function to the overall sum. This last function is called *activation function*. Activation functions are usually non-linear functions for limiting the output of a neuron, also referred to as squashing functions [Karlik and Olgac, 2011] [Daniel, 2013]. Two commonly used functions and their activation range are shown in figure 3.2.

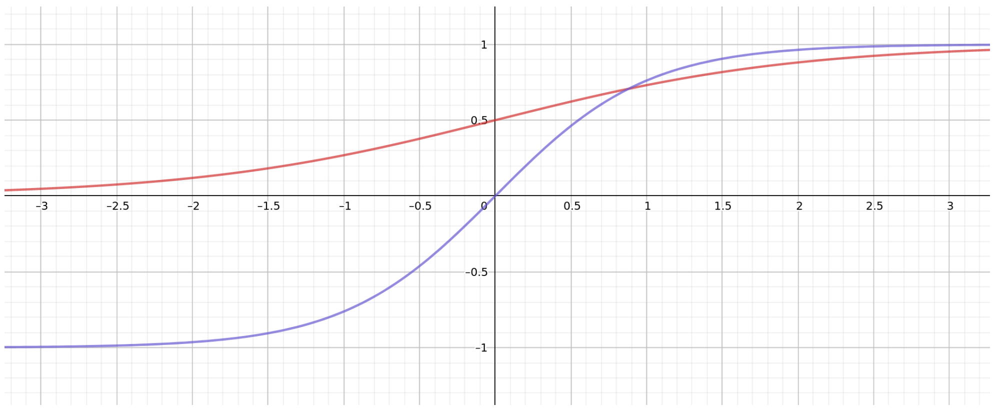


Figure 3.2: Sigmoid function in red and the hyperbolic tanget function in blue.

Perceptron

Perceptrons are the most simple artificial neural networks, composed of a single neuron. The following figure shows the architecture and calculations on a perceptron:

Usually, $\sum_{i=1}^{n} w_i x_i$ is written as $w_i^T x_i$, for a tidier representation and because modern computers are optimized for working with vectorized data. We can even place the bias into the vector multiplication as the equation 3.2 holds true:

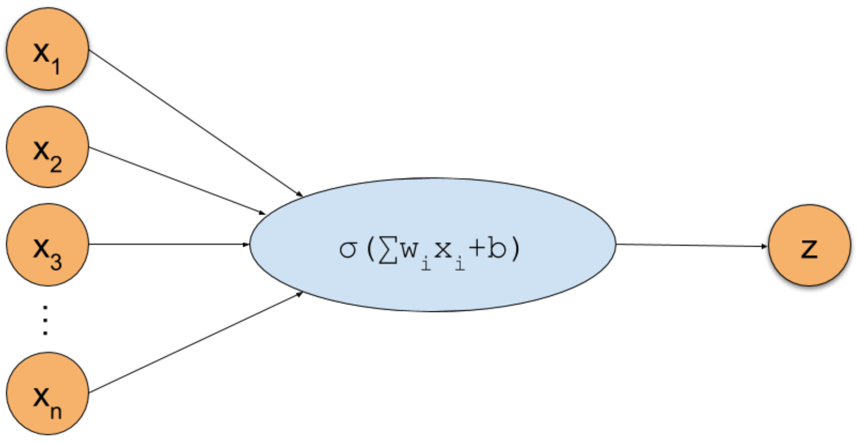


Figure 3.3: Structure and operations on a perceptron. The subscript *i* on *x* does refer to the *i*th element of the feature vector on this figure.

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \begin{bmatrix} x_1, x_2, \ldots, x_n \end{bmatrix} + b = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b \cdot 1 = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix} \begin{bmatrix} x_1, x_2, \ldots, x_n, 1 \end{bmatrix} \tag{3.2}$$

This is why sometimes bias is represented as another input with a constant value of 1 and weight b.

Perceptrons, just like linear models, are only capable of solving linearly separable problems, because they compute a linear combination of the inputs, even though the result is non-linearized.

Multilayer Perceptron

Multilayer perceptrons (MLPs) are a chain of layers of perceptrons, where each neuron takes the input of the previous layer and its output is "fed" into the next layer. These networks are called feed-forward networks because the calculation is sequentially performed layer by layer up to the output layer, although operations in each layer can be done in parallel. A three layer MLP is shown in figure 3.4.

The last layer of the network is called the output layer, because each neuron outputs a value that corresponds to the output vector. Sometimes the inputs are considered as another layer that does not compute any mathematical operation but is fully connected with the next layer [Gardner and Dorling, 1998]. The rest of the layers are called hidden layers because the user does not interact with those layer. This is why multilayer perceptrons can be considered black-box learners.

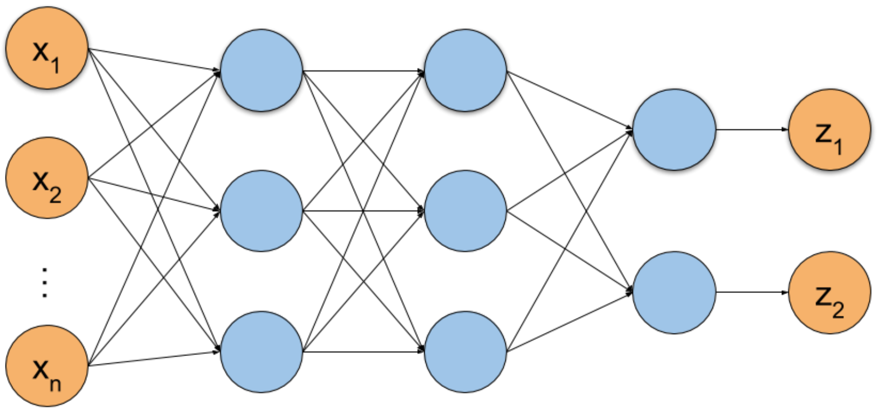


Figure 3.4: A multilayer perceptron with 3, 3 and 2 neurons in each layer

It is essential that multilayer perceptrons have a non-linear activation function, since otherwise, the whole network would be a linear transformation and a single layer perceptron would be able to exactly compute it [Lippmann, 1987]. Another essential property is having differentiable activation functions, as we will discuss on a subsequent section.

Loss function

Choosing the best parameters for optimizing the problem is what makes a model improve. But we must specify a mathematical expression associated to the fitness of the model that will quantify how appropriate the chosen parameters are. This function is called *loss function* and the goal is to find parameters that will converge this function to 0.

Here are some common loss functions:

1. Mean Absolute Error (MAE): $\frac{1}{n}\sum_{i=1}^{n} \mid y_i - z_i \mid$
2. Mean Square Error (MSE): $\frac{1}{n}\sum_{i=1}^{n} (y_i - z_i)^2$
3. Cross Entropy Loss[2]: $-\sum_{i=1}^{n}(y_i \log(z_i) + (1-y_i)\log(1-z_i))$

Gradient Descent

Gradient Descent is by far one of the most common method to optimize neural networks [Ruder, 2016a]. It is a strategy to minimize a loss function L by comprised of parameters θ by computing the gradient of L respective to the parameters. The gradient on a function indicates the steepest direction uphill.

Gradient Descent aims to move the parameters on the opposite direction of the gradient taking a step towards the steepest downhill direction, according to the following formula:

$$\theta' = \theta - \alpha \nabla L(\theta), \tag{3.3}$$

where α is the step size or learning rate and $\nabla L(\theta)$ is the gradient of L with respect to θ.

We can rewrite the loss function as $L(x, \theta)$, as it depends on the input data as much as on its parameters. This means that each sample will result in a different gradient, so the gradient should be computed for every x before updating θ, which could result computationally expensive.

This is why Stochastic Gradient Descent (SGD) is instead used, where only the gradient is computed just once (with one random instance x_r) before updating the parameters [Johnson and Zhang, 2013]. While this reduces the cost of computing the gradient considerably, it may skip some gradients that point towards the global minimum. This means that we have a trade-off between faster computation and faster convergence [Johnson and Zhang, 2013]. However, it has been shown that SGD shows the same convergence behaviour when slowly decreasing the learning rate [Ruder, 2016b].

Backpropagation

Finding a proper method for efficiently reaching to good parameters to converge the loss function is fundamental, as some artificial neural networks can have millions of parameters. For instance, AlexNet [Krizhevsky et al., 2012] has about 60 million. Therefore, an efficient method must be applied to rapidly adjust the parameters in a relatively inexpensive manner.

The most widely used tool in MLPs is backpropagation, which was invented back in 1986

[2]for cases where $y_i, z_i \in \{0, 1\}$. Also called binary logistic loss. It can be extended to allow multiclass cases.

by [Rumelhart et al., 1986]. Backpropagation repeatedly adjusts the parameters of the network to minimize the loss function and it works by calculating gradients efficiently in large systems made up of blocks represented by differentiable functions [Werbos, 1990]. This is why the non-linear activation functions are required to be differentiable (or slightly massaged to be so)[3]and why sigmoid and tanh are so popular, as pointed out in section 3.2.3.

In other words, it is based on the chain rule when differentiating composite functions. For example: suppose we are using a perceptron, the sigmoid function as the activation function and the cross entropy loss. Therefore:

$$\begin{cases} t = f(x) = w^T x + b \\ z = g(t) = \sigma(t) \\ l = l(y,z) = -(y\log(z) + (1-y)\log(1-z)) \end{cases} \implies \begin{cases} \dfrac{\partial f}{\partial w} = x \\ \dfrac{\partial f}{\partial b} = 1 \\ \dfrac{\partial g}{\partial t} = \sigma(t)(1-\sigma(t)) = z(1-z) \\ \dfrac{\partial l}{\partial z} = \dfrac{z-y}{z(1-z)} \end{cases} \tag{3.4}$$

Given that $loss = (l \circ g \circ f)(x)$, in order to calculate the gradient, we need to find $\dfrac{\partial l}{\partial w}$ and $\dfrac{\partial l}{\partial b}$, which according to the chain rule are:

$$\begin{cases} \dfrac{\partial l}{\partial w} = \dfrac{\partial l}{\partial g}\dfrac{\partial g}{\partial f}\dfrac{\partial f}{\partial w} = \dfrac{z-y}{z(1-z)} \cdot z(1-z) \cdot x = x(z-y) \\ \dfrac{\partial l}{\partial b} = \dfrac{\partial l}{\partial g}\dfrac{\partial g}{\partial f}\dfrac{\partial f}{\partial b} = \dfrac{z-y}{z(1-z)} \cdot z(1-z) \cdot 1 = z-y \end{cases} \tag{3.5}$$

3.2.4 SVM

Support Vector Machines (SVMs) behave similarly to linear classifiers with the difference that they introduce the concept of margins. Margins are hyperplanes parallel to the separator and can be visualized on figure 3.5. The goal of SVMs is to separate the dataset so that every point falls outside the margins in their corresponding region, choosing the separator for which the margin is maximized [Bishop, 2006].

[3] Although ReLU, $f(x) = \max(0,x)$, is not differentiable at $x = 0$, we can manually override $f'(x) = 1$.

Figure 3.5 shows the classification of a linear SVM over the same dataset as in figure 3.1, where the continuous line is the decision boundary and the dashed ones the margins.

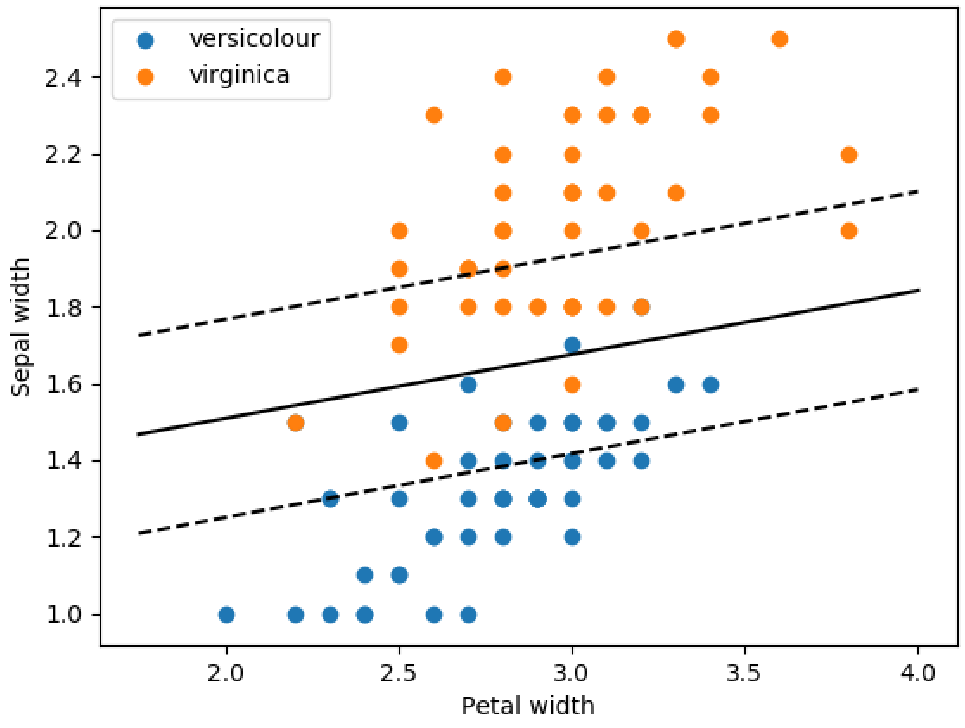


Figure 3.5: SVM margins.

However, there are few cases in the real world where a linear separator can achieve perfect accuracy, so we must introduce an error term to quantify and penalize misclassified cases.

Objective function

Just for this explanation, we will adapt the labels y to be -1 for the negative label. Then, we will define the function $f(x) = w^T x + b$ for the classifier, which is the one used for perceptrons without the activation function. The function will, in other words, return the distance from the hyperplane defined by parameters w and b.

Maximizing the margins is represented by $\arg\min_{w,b} \frac{1}{2} w^T w$, where b is implicitly determined by changes in w [Bishop, 2006]. Then, we will introduce the error ξ_i is introduced for each sample, which is 0 for correctly classified data points and $\mid y_i - z_i \mid$ for those laying on the wrong side of the margin. Notice that even though being on the right side of the separator, points between the separator and margins are also considered erroneously categorized, but will have a maximum error of 1, happening when the point lies just above the separator ($z_i = 0$).

We will combine both terms in the following equation:

$$\arg\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i, \tag{3.6}$$

where C will be an adjustable parameter which will specify the strength of the penalty. C can also be viewed as the parameter that determines the tradeoff between the magnitude of w and the tolerance of error [Smola and Schölkopf, 2004].

Kernel functions for non-linear classification

Once again, we have explained another linear classifier, which will not be able to solve most of real-world problems accurately. This time, we can achieve non-linearity without modifying the internals of the classifier by applying a kernel function.

Kernel functions are a mapping from the space of the input data to a highly dimensional space called *feature space*. By choosing the right mapping, the problem may become linearly separable at the feature space and thus, allowing the SVM to accurately solve the classification task. [Amari and Wu, 1999].

We will not explain different kernel functions, as it is outside the scope of this report. Instead, we will mention which one we have used in section 5.1.2.

3.3 Adversarial Learning

Originally designed to automatically adapt the model to changes such as spammers circumventing mail filters by altering certain words classified as spam, adversarial classifiers were introduced back in 2004 by [Dalvi et al., 2004]. These models consist on facing a classifier C against an adversary A which tries to "fool" C by attempting to make it classify some positive instances as negative.

A decade later, built upon adversarial learning, [Goodfellow et al., 2014] proposed Generative Adversarial Networks (GANs), where a different approach was used. A generative model G and a discriminative model D are simultaneously trained so that G produces new samples similar to the ones in the training set while D aims to determine whether the sample comes from G or the dataset. G and D are analogous to currency counterfeiters and the police.

We have implemented an adaptation of GANs proposed by [Zhang et al., 2018] ourselves, which we will explain in section 4.2.3.

3.4 Evaluation

One of the key principles that define a good model is the degree of generalization, which means how good the model is at extracting key information about seen data and applying it to correctly classify new cases. This is crucial because in the real world, classifiers are trained over a finite amount of samples whose distribution might not be the same as the one that the model will face when it is deployed.

This is why evaluating the performance of the model on the same set of data as the one used for fitting the model may be deceiving. Therefore, we split the dataset into the *training set*, used for optimizing the model, and *testing set*, used for evaluating the model.

In our case, we will use a technique called 10-fold cross-validation. This method splits the the database into 10 parts where a separate model is trained and tested for each part. For each $1 \leq i \leq 10$, the ith fold is used for testing and the rest for training. Table 3.2 shows the division of the dataset on each iteration.

Training folds	Testing fold
1, 2, 3, 4, 5, 6, 7, 8, 9,	10
1, 2, 3, 4, 5, 6, 7, 8, , 10	9
1, 2, 3, 4, 5, 6, 7, , 9, 10	8
...	...
1, 2, , 4, 5, 6, 7, 8, 9, 10	3
1, , 3, 4, 5, 6, 7, 8, 9, 10	2
, 2, 3, 4, 5, 6, 7, 8, 9, 10	1

Table 3.2: Cross validation folding

This method ensures that the entirety of the database is tested whilst no sample used for training is evaluated.

4. CHAPTER

Project development

4.1 Background

An increasing concern about fairness (or lack thereof) on multiple fields has been rising [Zafar et al., 2015] due to a higher academic interest in machine learning [Slack et al., 2020], which has led to an emergence of multiple distinct approaches for sorting out the consequent issues. The following list analyzes several techniques for dealing with unfairness proposed by different authors for different scenarios:

- [Buolamwini and Gebru, 2018] show a deep worry about the disparity in accuracy of classifiers with respect to gender and skin colour and point out how urgently attention is needed on this matter if companies want to rely on fair systems. They blame the underrepresentation of minorities in datasets and introduce a collection balanced both in gender and skin colour.
- [Dwork et al., 2017] show how machine learning systems inherently face a tradeoff between fairness and accuracy. They present the notion of Decoupled Classifiers, training separate classifiers for each group, and extend their work by using transfer learning to address underrepresentation of minority groups.
- [Slack et al., 2020] propose two algorithms:
    1. A model-agnostic algorithm that provides interpretable conditions for when a model is behaving fairly.

2. A meta-learning algorithm that intends to efficiently train models with few samples at hand while maintaining balanced fairness and accuracy.

- [Zafar et al., 2015] introduce a mechanism to control the degree of fairness relatively precisely and apply the mechanism to two classifiers: logistic regression and support vector machines. In addition, they publicly provide a Python implementation of their work.

- The same authors also introduce a new notion of unfairness in another paper called *disparate mistreatment* [Zafar et al., 2017], defined in terms of misclassification rates. They then provide measures of *disparate mistreatment* for decision boundary-based classifiers.

- [Beutel et al., 2017] use an adversarial training procedure to remove information about the sensitive attribute and observe the effects on fairness properties. They find out that a small dataset is enough for their models and that data itself guides a notion fairness to the model.

- [Zhang et al., 2018], frequently citing and comparing [Beutel et al., 2017], present a framework using adversarial learning for mitigating bias. Their objective is to maximize the ability of the predictor to correctly output the class while minimizing the adversary's ability to predict the protected attribute.

- [Zhao et al., 2017] focus on bias on visual recognition problems with supported captions sourced from web corpora. They (1) identify and quantify bias on those datasets, (2) evaluate how models amplify that bias and (3) propose a calibration algorithm that introduces corpus-level constraints to classifiers for reducing bias.

- [Garg et al., 2018] show that word embeddings (vector representation of words) used in Natural Language Processing (NLP) convey 100 years of gender and ethnic stereotypes and demonstrate how dynamics of embeddings reflect the change of stereotipical attitudes towards women and ethnic minorities.

- [Zhao et al., 2019] perform a thorough analysis on the gender bias on ELMo embeddings (deep contextualized word representations [Peters et al., 2018]) and explore two methods for mitigating the disparity. [Bolukbasi et al., 2016] also present a similar concern over the discrimination of embeddings trained on Google News and provide a methodology for removing that bias.

- Nevertheless, a countering criticism of debiased embeddings has already emerged. [Gonen and Goldberg, 2019] argue that although bias is reduced by definition, the actual effect is hidden instead instead of removed, making the debiasing superficial.

4.2 Selected methods

Due to inexperience with NLP tasks and a lack of understanding on how to manipulate word embeddings, this work will be focused on regular binary classifiers. Terminology has been suited to our naming method and the classifiers will be working with one sensitive attribute with binary values, as well as a binary class.

Consequently, we have chosen to compare and contrast the performances of Decoupled Classifiers [Dwork et al., 2017], Fairness Constrains [Zafar et al., 2015] and Adversarial Learning [Zhang et al., 2018]. These techniques deal with binary protected attributes in general, not necessarily gender.

Nonetheless, we have sourced different concepts and fairness metrics from many other papers, which will complete the comparisons.

4.2.1 Decoupled Classifiers

The core concept of Decoupled Classifiers resides in training a separate classifier for each group. In our case, one classifier would process women and a distinct would process men. This decoupling technique can be applied on top of any black-box type machine learning model, allowing us to test it with the different models presented in chapter 3.

This decoupling procedure requires a specific loss function that takes into account the outputs from both classifiers. In order to introduce fairness into the equation, the function needs to penalize differences across the predictions in each group. The following loss function is presented in the paper:

$$\frac{\lambda}{n}\sum_{i=1}^{n}|y_i - z_i| + \frac{1-\lambda}{n}\sum_{k=1}^{K}\left|\sum_{i:a_i=k} z_i - \frac{1}{K}\sum_{i} z_i\right|, \tag{4.1}$$

where K is the number of groups. The left term in the outermost sum computes the mean absolute error (MAE) and the right term calculates the difference in positives across groups, which can be summarized as adding accuracy and fairness, respectively. $\lambda \in [0,1]$

is used as an interpolating value for weighing the relevance of accuracy and fairness in the loss, which makes the relation between them explicit.

We can adapt the equation to our context, provided that the sensitive binary attribute will be gender, resulting in the following equation:

$$\frac{\lambda}{n}\sum_{i=1}^{n}|y_i - z_i| + \frac{1-\lambda}{n}\left(\left|\sum_{men} z_i - \frac{1}{2}\sum_{all} z_i\right| + \left|\sum_{women} z_i - \frac{1}{2}\sum_{all} z_i\right|\right) \tag{4.2}$$

The second term in the outermost sum computes the deviation of positives in men and women from the half of all positives. Therefore, the classifier gets penalized for leaning towards one of the groups proportionally to λ.

A positive side effect of decoupling is that we can extract how the relevance of each attribute varies from group to group. For instance, how differently the classifiers relate the number of children to the outcome depending on the gender of an individual.

Transfer Learning

Due to the nature of machine learning algorithms, a big threat to the algorithm's performance is the imbalance of data. Given the case where 10% of data in the training set is positive ($\mathcal{Y} = 1$) and 90% negative ($\mathcal{Y} = 0$), there are two main issues with the distribution:

- Depending on the size of the dataset, a tenth of its length might result in a too small number for the model to properly train.
- The severe disproportion in data may make the model tend to classify much better negatives than positives. In some cases, the model might even classify all instances as negative if the loss is proportional to the accuracy, which will be 90%.

The same issue happens when we divide the dataset by the sensitive attribute, as a common source of bias is skewed data [Beutel et al., 2017]. In our case, if 90% of the dataset is made up of men, the decoupled classifier for women will most surely fall short in having enough samples to train with. We will call this unevenness *underrepresentation*.

For instance, patients of African ancestry are more likely to be incorrectly diagnosed with a life-threatening heart condition known as hypertrophic cardiomyopathy due to the fact that over 80% of all genetics data are collected from individuals of European ancestry [Manrai et al., 2016].

We will stress on how underrepresentation affects on decoupled classifiers, as the minority group struggles in both accuracy and fairness due to the limitation in the amount of data. Hence, we will allow the minority group to source some data from the majority group.

The authors of decoupled classifiers refer to this technique as *Transfer Learning* and propose an algorithm for down-weighing or down-sampling out-group[1] data via an transfer learning algorithm. However, we will approach it in a different manner by specifying a parameter $\theta \in \{0, 0.05, 0.1, ..., 0.3, 0.35\}$ that will control the proportion of out-group samples that will be added to the training set. The best theta will be chosen by running a 10-fold cross-validation and returning the model that minimizes the joint loss.

We have chosen a maximum theta of 0.35 because according to experiments, both accuracy and fairness start showing an asymptotic tendency after $\theta \sim \frac{1}{3}$ and it pollutes the idea of decoupling. For the same reasons, we will not apply transfer learning to the majority class, as neither metric improves. As an example, an experiment has been carried out testing different values for theta and the results are presented in appendix A.

4.2.2 Fairness Constraints

The authors of the article provide two different approaches and formulations for balancing accuracy and fairness. The first one aims to maximize accuracy under fairness constraints and the second aims to maximize fairness under accuracy constraints. We will employ the former method because it enables a granular control over the strictness of fairness requisites.

Following the original article, we will explain the concept by using a linear separator where the decision boundary will be denoted by $w^T x = 0$. In turn, $w^T x$ will indicate the signed distance to the decision boundary. We have integrated the bias in the weight vector as mentioned in section 3.2.3, so the last element in x will always be a 1.

They propose to measure unfairness as the covariance between the users' sensitive attribute a and the signed distance from the users' data to the decision boundary $w^T x$. Covariance is computed as follows:

$$\frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})w^T x_i \tag{4.3}$$

[1] from the other group's dataset

The goal of this method is to find the parameters w that minimize the corresponding loss function (which is cross entropy loss) under the following constraints:

$$\begin{aligned} \min \quad & L(w) \\ \text{s.t.} \quad & \frac{1}{n}\sum_{i=1}^{n}(a_i-\bar{a})w^T x_i \leq c \\ & \frac{1}{n}\sum_{i=1}^{n}(a_i-\bar{a})w^T x_i \geq -c \end{aligned} \tag{4.4}$$

where L is the loss function and $c \geq 0$ is the covariance threshold. Lowering it c towards 0 will introduce stricter constraints that will result in a higher satisfaction of the p% rule (see section 4.3.1), whilst potentially sacrificing loss.

Remark: the sensitive attribute will not be used for training the model. In other words, a will not be part of x.

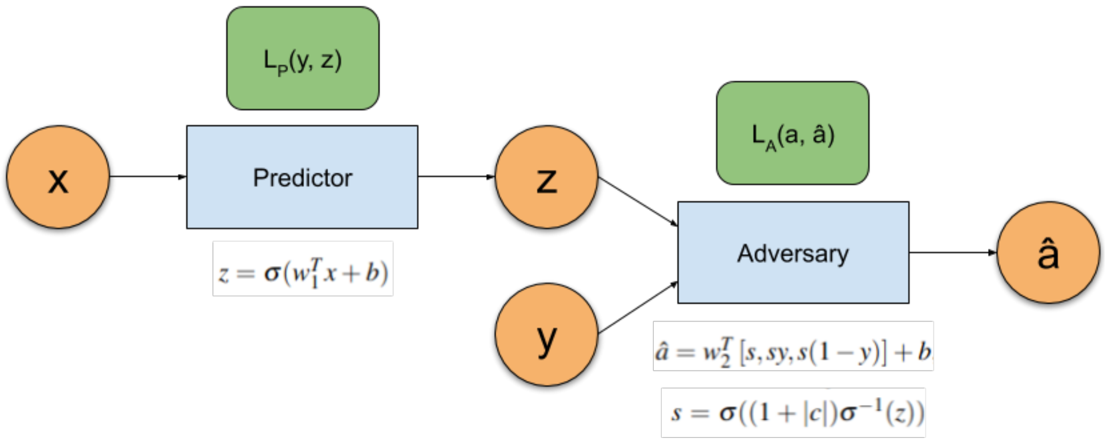
Logistic Regression

In our experiment, we will apply this technique to logistic regression in order to compare the results with the decoupled logistic regression. Therefore, we have to change the equations to the following:

$$\begin{aligned} \min \quad & L(w) = -\sum_{i=1}^{n}(y_i \log(w^T x_i) + (1-y_i)\log(1-w^T x_i)) \\ \text{s.t.} \quad & \frac{1}{n}\sum_{i=1}^{n}(a_i-\bar{a})w^T x_i \leq c \\ & \frac{1}{n}\sum_{i=1}^{n}(a_i-\bar{a})w^T x_i \geq -c \end{aligned} \tag{4.5}$$

4.2.3 Adversarial Learning

Based on the concept of GANs, the authors of this technique have adapted the model so that the generative model will act as a predictor by determining $\mathcal{Z}$ given $\mathcal{X}$ and the adversary will try to guess $\mathcal{A}$ based on $\mathcal{Z}$ and $\mathcal{Y}$. The objective is to minimize the ability to predict $\mathcal{A}$ based on $\mathcal{Z}$, in order to dissociate outcomes with, in our case, gender. Figure 4.1 illustrates the structure of the GAN.



$L_P(y, z)$
x
Predictor
z
$z = \sigma(w_1^T x + b)$
$L_A(a, \hat{a})$
Adversary
y
â
$\hat{a} = w_2^T [s, sy, s(1-y)] + b$
$s = \sigma((1+|c|)\sigma^{-1}(z))$

Figure 4.1: Structure of our GAN

We will use the same network as the one used in the article for the experiment **UCI Adult Dataset** on section 6. The predictor will be a regular logistic regression of the form $z = \sigma(w_1^T x + b)$ and the adversary will calculate $\hat{a} = w_2^T [s, sy, s(1-y)] + b$, where $s = \sigma((1+|c|)\sigma^{-1}(z))$, w_2 a three weight vector and $\hat{a}$ denotes the prediction of the sensitive attribute a. We add 1 to $|c|$ to keep the adversary away from ignoring y by setting $c = 0$, which can be a tough local minimum to escape.

The article does not specify any restrictions or guidelines about loss functions, so we have chosen MSE (Mean Square Error) for both models. But what the article specifies is how to modify parameter w_1 of the predictor according to the losses of the predictor and adversary, which we will denote L_p and L_a respectively. w_1 (represented as w for conciseness) will be modified according to the following expression:

$$\nabla_w L_p - proj_{\nabla_w L_a} \nabla_w L_p - \alpha \nabla_w L_a \tag{4.6}$$

For example, SGD (Stochastic Gradient Descent) would update the parameters under the learning rate lr the following way:

$$w' = w - lr(\nabla_w L_p - proj_{\nabla_w L_a} \nabla_w L_p - \alpha \nabla_w L_a) \tag{4.7}$$

Ignoring the middle term in equation 4.6, we can see that the weights for the predictor need to be adjusted according to the gradients of both loss functions. The loss regarding the adversary is allegedly scaled by α to avoid getting stuck on local minima. The authors propose to dynamically set α for the Adult dataset. However, our experiments have shown that a value of 0.1 works overall fine with the majority of test cases.

In the absence of the middle term, the predictor would modify the weights in a direction that would actually help the adversary. That term cancels that effect, as can be seen in figure 4.2. The weights of the adversary will be adjusted with an algorithm based on SGD called Adam [Kingma and Ba, 2014].

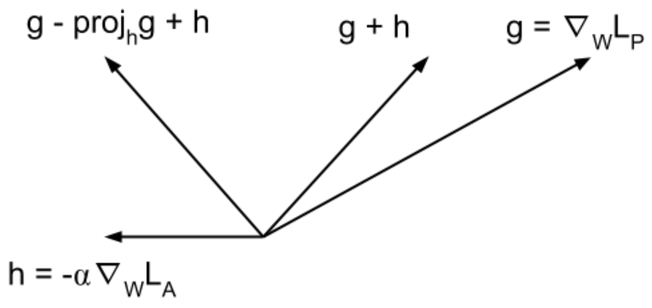


Figure 4.2: Direction of gradients and their projections.

4.3 Fairness metrics

We will reemphasize that models aim to fit the mapping as close as possible to the annotated data, and this is the reason why biased datasets create biased models, mentioned in the introduction. Accuracy, despite being important, is oblivious to fairness. In fact, it is considered that there is an inherent tradeoff between accuracy and fairness [Slack et al., 2020].

This tradeoff involves to carefully finding a delicate balance. Hence, we will take a look at multiple measurements that consider differences between groups.

4.3.1 Demographic parity

Also known as the lack of disparate impact [Slack et al., 2020], demographic parity is the ratio between positives in the minority and majority group. Mathematically, we can write it as:

$$\frac{P(Z=1|A=0)}{P(Z=1|A=1)} \tag{4.8}$$

A value closer to 1 indicates that the proportion of positives in both groups is similar, achieving demographic parity when the ratio equals 1.

A closely related metric is the p% rule, satisfied when:

$$\min\left(\frac{P(Z=1|A=0)}{P(Z=1|A=1)}, \frac{P(Z=1|A=1)}{P(Z=1|A=0)}\right) \geq \frac{p}{100} \tag{4.9}$$

4.3.2 Equalized odds

Equalized odds is met when the classification has equal true positive rates (TPR) and false positive rates (FPR). It can be written as:

$$\frac{P(Z=1|A=0,Y=y)}{P(Z=1|A=1,Y=y)}, y \in \{0,1\} \tag{4.10}$$

where the ratio between TPRs is calculated when $y = 1$ and FRPs when $y = 0$.

This measure requires that the proportion of both positive and negative predictions are leveled across groups. This notion of fairness can be observed in the joint loss function of decoupled classifiers (equation 4.2).

4.3.3 Equal opportunity

Equal opportunity is the relaxed version of equalized odds, where $y = 1$ and, in turn, $Y = 1$. This metrics only requires non-discrimination in the positive group [Hardt et al., 2016], often allowing for a higher accuracy [Slack et al., 2020].

In this report, as we are just measuring instead of requiring these metrics, we will calculate both values for equalized odds.

4.3.4 Bias score

Proposed by [Zhao et al., 2017], this metric evaluates the correlation between a sensitive attribute and another non-sensitive attribute. In our case, we will observe how correlated the gender and the outcome are by computing the following value:

$$\frac{c(Z=1,A=0)}{c(Z=1,A=0)+c(Z=1,A=1)} \tag{4.11}$$

where $c(Z=z,A=a)$ counts the number of appearances of $Z=z$ and $A=a$ altogether.

In other words, we will observe the proportion of each group in the positively classified classes. A value closer to 0.5 indicates less bias.

Notice that this metric is similar to demographic parity, but instead of probabilities, it uses counts. Therefore, if the proportion of males and females in the database is not balanced, this metric might come into conflict with previous ones. For instance, if 5000 males and 100 females account for the whole dataset and exactly half of the people in each group are positively classified, demographic parity will be ideal (1), but this metric will indicate severe disproportion (~ 0.02).

However, this metric can be of great use when the equality in the amount of positives between groups is required instead of the equality in the proportion. For example, in 50/50 campaigns that demand parity in the number of men and women.

5. CHAPTER

Experiments

5.1 Methodology

On the one hand, we have gathered different databases and transformed them in order to follow a common standard. On the other hand, we have implemented our fairness metrics and the chosen techniques to mitigate bias. Finally, we have tested the techniques and evaluated the performances over the transformed databases.

5.1.1 Database preprocessing

Sourcing different databases from different sites and collected by different people for different purposes results in having each database with a different arrangement and representation of data. Therefore, we will apply the following transformation to all datasets:

- The proposed sensitive attribute in the article that uses the database (see section 5.2) will be used. The minority will hold the value $\mathcal{A} = 0$ and majority $\mathcal{A} = 1$. If no sensitive attribute is recommended, gender will be taken as the protected attribute. If none of the two conditions before are met, another attribute that we regard as useful will be taken.

- Regarding the label, the value that corresponds to the positive outcome will be set to $\mathcal{Y} = 1$ and negative to $\mathcal{Y} = 0$.

- Continuous values like age will be linearly clamped to fit the range of $[0,1]$ by applying the following formula: $v'_i = \dfrac{v_i - \min v}{\max v - \min v}, \forall v_i \in v$.
- Discrete attributes with less than 10 unique values will get one-hot encoded. This means creating a new column for each value where that value gets the value of 1 and the rest 0.
- Discrete attributes with 10 or more unique values replace the most common with the value of 1 and the rest with 0, in order to avoid excessive dimensionality.

A case-by-case description of the preproccessing for $\mathcal{A}$ and $\mathcal{Y}$ of each database is collected in appendix B.

5.1.2 Classification

The whole project has been coded in Python and generally using Jupyter notebooks. Python provides relatively well-performing libraries that are optimized for mathematical operations like *numpy*, *scipy* and have a variety of customizable models like *scikit-learn* or *pytorch*.

Jupyter notebooks allows alternating code and results by grouping a scope of code in cells and outputting the result below. For instance, it makes it convenient for plotting figures under an experiment.

Decoupled Classifiers

In order to record the performances of the coupled and decoupled variants for the classifiers, we have to create three sets of training data:

1. Coupled set. This set will include the whole dataset.
2. Majority set. This set will just hold samples where $\mathcal{A} = 1$.
3. Minority set. This set will just hold samples where $\mathcal{A} = 0$.

The first set will be used for training a regular (coupled) classifier while the latter pair will be for its respective decoupled classifier. Our experiments will compare the results of decoupling on top of logistic regression, MLP and SVM in order to observe the impact of decoupling in different classifiers.

The logistic regression we have used has been sourced from the library scikit-learn[1], which computes the probabilities by applying formula 5.1, where C is the (inverse) regularization factor.

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(w^T x_i + b)) + 1) \tag{5.1}$$

The algorithm we will use to find the optimal parameters we will be the defaulted *lbfgs*, which is the one recommended by the guide of scikit-learn due to its robustness. Limited-memory BFGS (lbfgs) is an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) using a limited amount of computer memory.

For the multilayer perceptron, we will also use the implementation provided by scikit-learn[2]with 2 hidden layers, each with 30 neurons. The rest of the parameters are left default but for the maximum iterations before convergence, which has been set to 10,000.

Once again, we have employed the scikit-learn library and chosen their SVC[3]implementation for SVMs, which solves equation 3.6. We have not modified any default parameters, letting $C = 1$ in order to apply error regularization. *Radial Basis Function* kernel (RBF) will be used.

Fairness Constraints

Fortunately, the authors of the paper publicly provide the implementation for their work. It is coded in Python (version 2) and they provide an introductory guide for using the code in their repository[4].

Therefore, our task reduced to preparing the data for matching their intended shape. For example, they specify the negative outcome as -1, and they do not allow the protected attribute in the training data.

[1]https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[2]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[3]https://scikit-learn.org/stable/modules/svm.html#svc

[4]https://github.com/mbilalzafar/fair-classification/

For each database, we will run two tests: one without the fairness constraints and the other with maximum constraints by setting the covariance threshold $c = 0$.

Adversarial Learning

We have developed the adversarial model using the Pytorch[5]library. It's high modularity allows the user to create, run and evaluate complex algorithm with a high level of abstraction as well. For example the following code shows the specification of our predictor and adversary and it's forward step:

Listing 5.1: Definition of the predictor and adversarial model

```
class Predictor(nn.Module):
    def __init__(self, n_input):
        super(Predictor, self).__init__()
        self.fc = nn.Linear(n_input, 1)
        self.sg = nn.Sigmoid()

    def forward(self, x):
        return self.sg(self.fc(x))


class Adversary(nn.Module):
    def __init__(self):
        super(Adversary, self).__init__()
        self.c = nn.Parameter(torch.rand(1))
        self.sg = nn.Sigmoid()
        self.w2 = nn.Linear(3, 1)

    def forward(self, x, y):
        s = self.sg((1 + torch.abs(self.c)) * (torch.log(x) - torch.log1p(-x))).view(-1)
        x = torch.stack((s, s * y, s * (1 - y))).transpose(0, 1)
        return self.w2(x)
```

The most delicate part of the process has been implementing the formula 4.6, as it is a custom procedure that needed to be coded "by hand", as no method in the library would make such calculation.

All the process of training and testing has been also done with Pytorch, as it is optimized for those tasks. It even supports GPU computing, but we have not used that feature because of the lack of available high-end GPUs and because of the relatively small datasets.

[5]https://pytorch.org/

5.1.3 Performance measuring

After training every model, we will retrieve the vector (or vectors, see remarks below) of predictions $\mathcal{Z}$ for every technique and then apply the five following metrics in order to calculate the performances:

- **Accuracy**: ranging from 0 to 1. Higher accuracy means more correct predictions.
- **Loss**. A value towards zero indicates better classification.
- **Equalized odds**. $\mathcal{Y} = 0$ and $\mathcal{Y} = 1$ (equal opportunity). The ideal value is 1.
- **Demographic parity**. The ideal value is 1.
- **Bias score**. The ideal value is 0.5.

Remark 1: The scores for the coupled variant of the decoupled classifier have also been recorded because it resembles the performance of a regular classifier and can be viewed as a baseline.

Remark 2: In the case of Fairness Constraints, the two tests with high and low covariance threshold have been recorded for the same reason.

Remark 3: Demographic parity and the bias score are also tested with the dataset in order to provide a sense of baseline as well. This was done by replacing $\mathcal{Z}$ by $\mathcal{Y}$ in the equations.

5.2 Datasets

A first collection of datasets has been selected because they have been used in some paper in our references. Then, others have been additionally sourced from the online collection of databases `https://openml.org` in order to broaden the variety of datasets. One extra dataset was provided by the director as well. Table 5.1 shows a list of the used datasets and their origin.

We will take a brief look at the characteristics of each dataset before having undergone the preprocessing stage.

Adult	[Celis et al., 2019], [Zafar et al., 2015], [Zhang et al., 2018]
Bank	[Zafar et al., 2015]
COMPAS	[Celis et al., 2019], [Slack et al., 2020], [Zafar et al., 2017]
Credit	openml.org
Hepatitis	openml.org
Infor	Provided by the director
Nurse	openml.org

Table 5.1: Origin of each dataset

5.2.1 Adult

The database is intended to be used for predicting whether an individual's income is greater than $50,000 per year based on census data. There are 32561 cases with 14 attributes. The last unnamed column is the class and we have taken column "sex" as sensitive. These are the statistics for the class and sensitive attribute:

	$\mathcal{Y} = 0$	$\mathcal{Y} = 1$	Total
$\mathcal{A} = 0$	9592	1179	10771
$\mathcal{A} = 1$	15128	6662	21790
Total	24720	7841	32561

Table 5.2: Distribution of Adult

5.2.2 Bank

The database is intended to be used for predicting whether a bank client will subscribe a term deposit. There are 45211 cases with 17 attributes. Column "y" is the class and "age" the sensitive value. Statistics:

	$\mathcal{Y} = 0$	$\mathcal{Y} = 1$	Total
$\mathcal{A} = 0$	1288	709	1997
$\mathcal{A} = 1$	38634	4580	43214
Total	39922	5289	45211

Table 5.3: Distribution of Bank

5.2.3 Compas

We have discarded some attributes due to a high number of missing values. Therefore, we will use attributes *sex, race, age, juv_fel_count, juv_misd_count, juv_other_count, decile_score, priors_count, c_charge_degree, c_charge_desc, is_recid, is_violent_recid* from the "people" SQL table of this database for predicting the risk of recidivism of a jail inmate. There are 11038 cases with 12 attributes. "is_recid" has been taken as class and "race" as sensitive. Statistics:

	$\mathcal{Y}=0$	$\mathcal{Y}=1$	Total
$\mathcal{A}=0$	3326	2175	5501
$\mathcal{A}=1$	4009	1528	5537
Total	7335	3703	11038

Table 5.4: Distribution of Compas

5.2.4 Credit

The database is intended to be used for predicting whether a requester is worth lending the credit. There are 1000 cases with 20 attributes. As the name suggests, the column "class" has been taken as class and "gender" as protected. Statistics:

	$\mathcal{Y}=0$	$\mathcal{Y}=1$	Total
$\mathcal{A}=0$	191	499	690
$\mathcal{A}=1$	109	201	310
Total	300	700	1000

Table 5.5: Distribution of Credit

5.2.5 Hepatitis

The database is intended to be used for predicting whether a patient diagnosed with hepatitis will survive. There are 125 cases with 19 attributes. However, due to its high number of missing values, columns "ALK_PHOSPHATE", "ALBUMIN" and "PROTIME" have been removed, we have imputed the mean for numerical columns "BILIRUBIN" and "SGOT" and imputed the most common value for discrete columns. "Class" and "SEX" are the class and sensitive attribute, respectively. Statistics:

	$\mathcal{Y}=0$	$\mathcal{Y}=1$	Total
$\mathcal{A}=0$	0	16	16
$\mathcal{A}=1$	32	107	139
Total	32	123	155

Table 5.6: Distribution of Hepatitis

5.2.6 Infor

The database is intended to be used for predicting whether kids around 10 to 12 years depict a computer programmer as a woman. There are 866 cases with 20 attributes. "MarrazkiarenGeneroa" is the column for the class and "Sexua" (sex) the sensitive attribute. Statistics:

	$\mathcal{Y}=0$	$\mathcal{Y}=1$	Total
$\mathcal{A}=0$	214	243	457
$\mathcal{A}=1$	31	378	409
Total	245	621	866

Table 5.7: Distribution of Infor

5.2.7 Nurse

The database is intended to be used for classifying applicants to a medical school in Ljubljana, Slovenia. There are 12960 cases with 8 attributes. Once again, "class" is the class but "finance" is the protected variable. Statistics:

	$\mathcal{Y}=0$	$\mathcal{Y}=1$	Total
$\mathcal{A}=0$	2160	4320	6480
$\mathcal{A}=1$	2160	4320	6480
Total	4320	8640	12960

Table 5.8: Distribution of Nurse

5.3 Results

First, we will discuss the general results of the Compas dataset, what each figure represents and how to interpret them. Then we will point out singularities in other datasets but their respective figures will be shown in section 5.3.4. Finally we will explain the Nurse database, which is an unusual case.

Figure 5.1 shows 6 different metrics recorded for the performances of various classifiers. For each subplot, the horizontal axis has 5 groups:

- The first three correspond to decoupled classifiers, where M1 is logistic regression, M2 multilayer perceptron and M3 SVM. The blue bar represents the performance of the coupled classifier and the orange bar beside its decoupled variant.
- The forth group corresponds to the two classifiers for Fairness Constraints. The green bar shows the performance of the unconstrained logistic regression (infinite tolerance of covariance) while the red bar shows the performance where the tolerance is 0.
- The rightmost bar corresponds to the adversarial model.

The horizontal lines in the 4 bottom plots, the light blue marker indicates the ideal value of the metric whereas the yellow one in the last two figures indicates the measurement of the database, which will serve as a baseline.

5.3.1 Compas dataset

Regarding accuracy, there is a minimal improvement of the performance of decoupled classifiers over their coupled versions but there is a considerable gap with respect to the adversarial model. The remaining case backs the notion of the tradeoff between accuracy and fairness because the accuracy of the constrained classifier (red bar) is slightly lower but the joint loss improves significantly, being the best of all.

The reason of such phenomenon is that the joint loss takes into account the differences of the performance across groups. The red bars in the bottom 4 figures shows a relatively big improvement with respect to the green bar, which makes the joint loss decrease.

Both figures in the middle row are missing one bar, which is the result of a division by zero in the calculation of the metric. The left one, which measures the ratio of FPRs between groups, suggests that the adversarial classifier did not output any false positives for the non-protected group ($\mathcal{A} = 1$). Likewise, the coupled SVM did not output any true positives. Note that this value is infinity, not zero, but the plotting software ignores infinite values.

Nevertheless, both figures show a significant improvement in equalized odds as the second bar is much closer to 1 than the first. In the case of equal opportunity of the adversarial, the value is just shy of 1, which can be considered almost optimal.

Demographic parity and bias scores show no clear tendency for decoupled classifiers, but they do for the constraint models and the adversarial model's performance is around the database measurement.

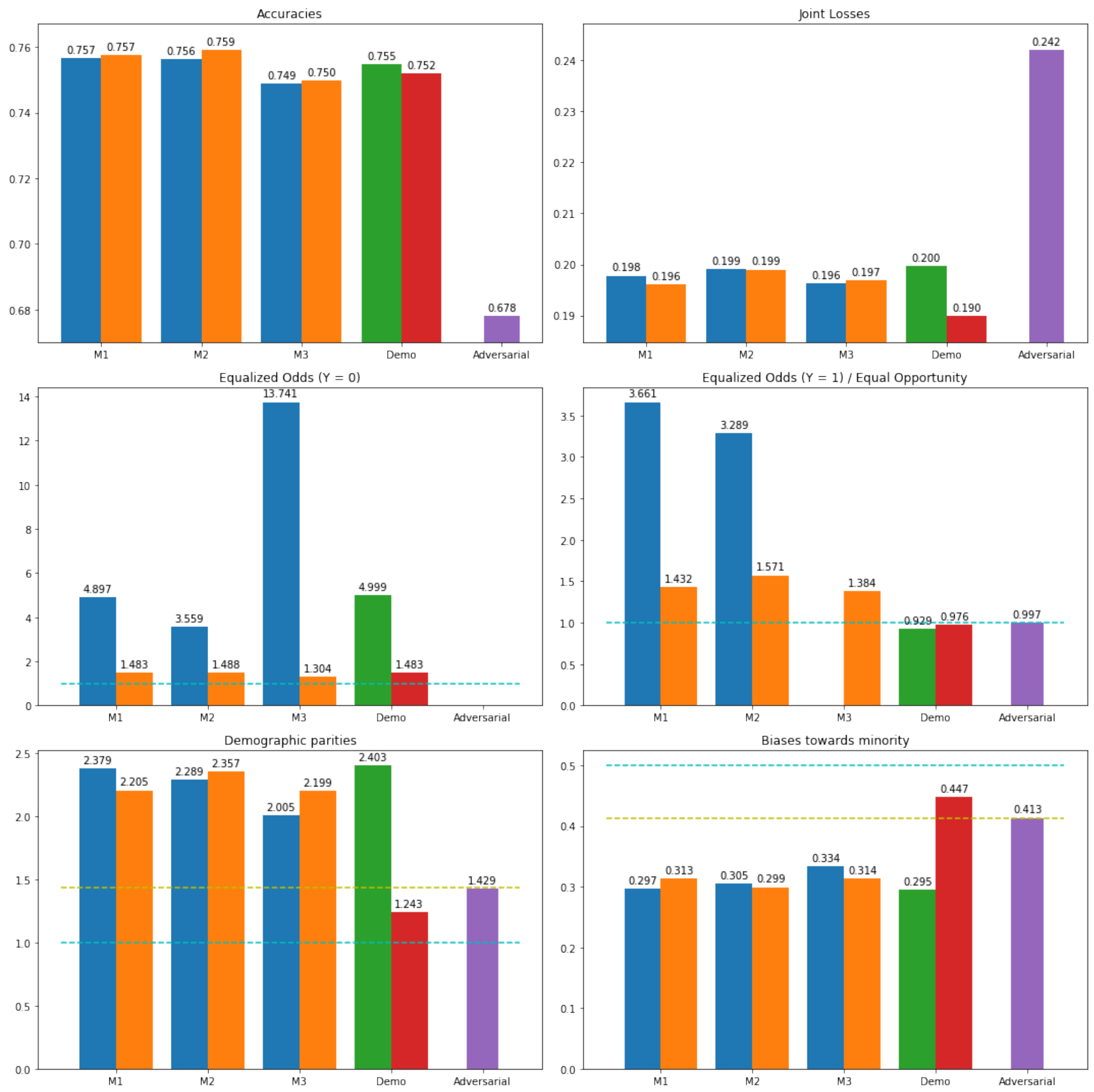


Figure 5.1: Results of the Compas dataset

5.3.2 Other remarks

Overall, decoupling results in an increase in both accuracy and fairness as the majority of accuracies (Adult-M1 and Hepatitis-M2 are the exceptions) grow while equalized odds tend to converge towards 1, M2 being the best performing model among the three. However, demographic parity seems to improve as often as it worsens. On the other hand, fairness constraints almost consistently improves fairness metrics while slightly sacrificing accuracy. Finally, the adversarial model scores the worst accuracies, and while sometimes achieving perfect scores in fairness, it intermittently outputs unplottable fairness values.

There are multiple cases where the adversarial model has no bar on the plots, which is a concerning issue because it means that one of the groups has an score of 0 for that metric. This may be due to the classifier getting stuck on a local minimum where all samples from one group are classified with the same label. This is very risky because it means that regardless of the information about the individual, the classifier is going to output the same value, which is extremely unfair.

Equalized odds in the Hepatitis dataset are the most extreme case of the previous issue, where the coupled, unconstrained and the adversarial (when $\mathcal{Y} = 0$) models demonstrate the same behaviour.

Remember that underrepresentation can impact on bias scores and make it fall in contradiction with other scores if the database presents a large disproportion of men and women (or whichever values the sensitive attribute may hold), as stated in section 4.3.4.

5.3.3 Nurse dataset

This case is very interesting as all the classifiers are capable of perfectly classifying the cases both in terms of accuracy and fairness. The cause of this singularity lies on our choice of the protected attribute and class distribution (specified in appendix B). By mere coincidence the following two phenomena occur:

All cases where the column "health" is "not_recom", the class is "not_recom", which we assigned the value 0. On the contrary, all cases where "health" is not "not_recom", the class takes the value of 1 because we grouped all other values for the class together. Besides, there is the same number of cases in each group where the class is positive, having no difference across classifiers and making the classifier optimally fair. Thus, the performance metrics consider the classifiers perfect in accuracy and fairness.

Although this case can be avoided by choosing another distribution of class or protected attribute, it shows a ridiculous extreme of a dataset and why fairness metrics can sometimes lead to non-numerical unplottable values.

5.3.4 All results

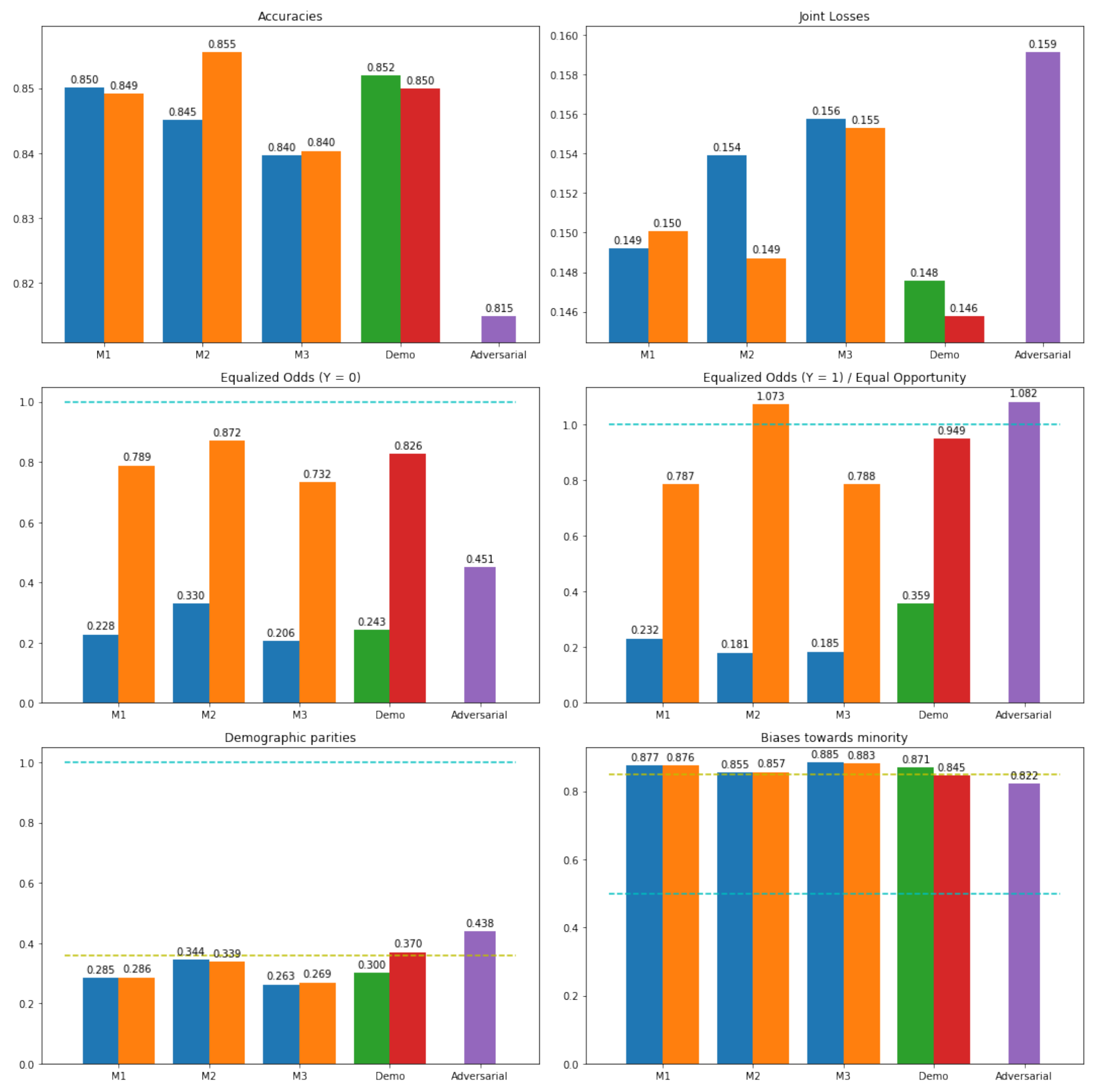


Figure 5.2: Results of the Adult dataset

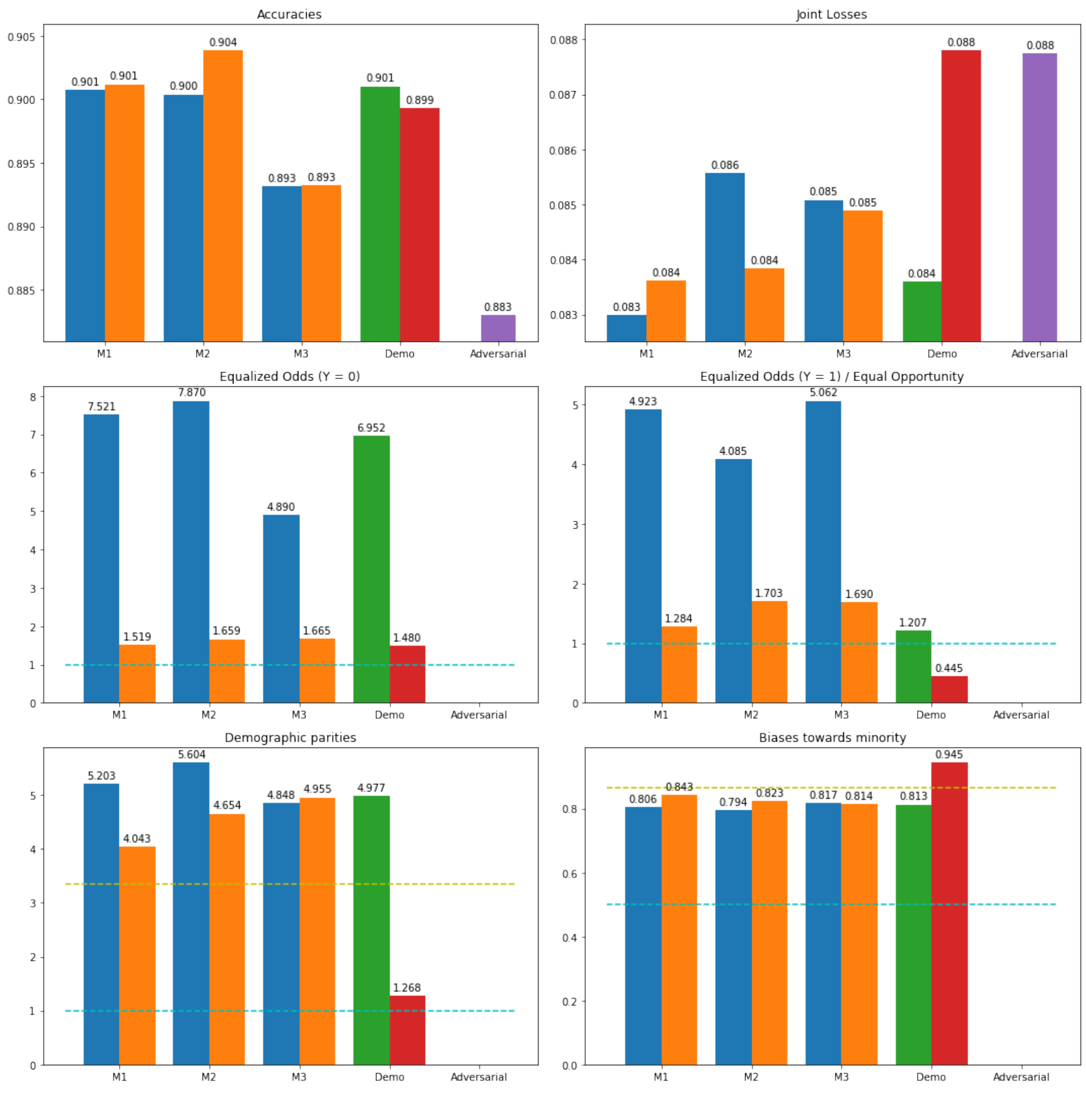


Figure 5.3: Results of the Bank dataset

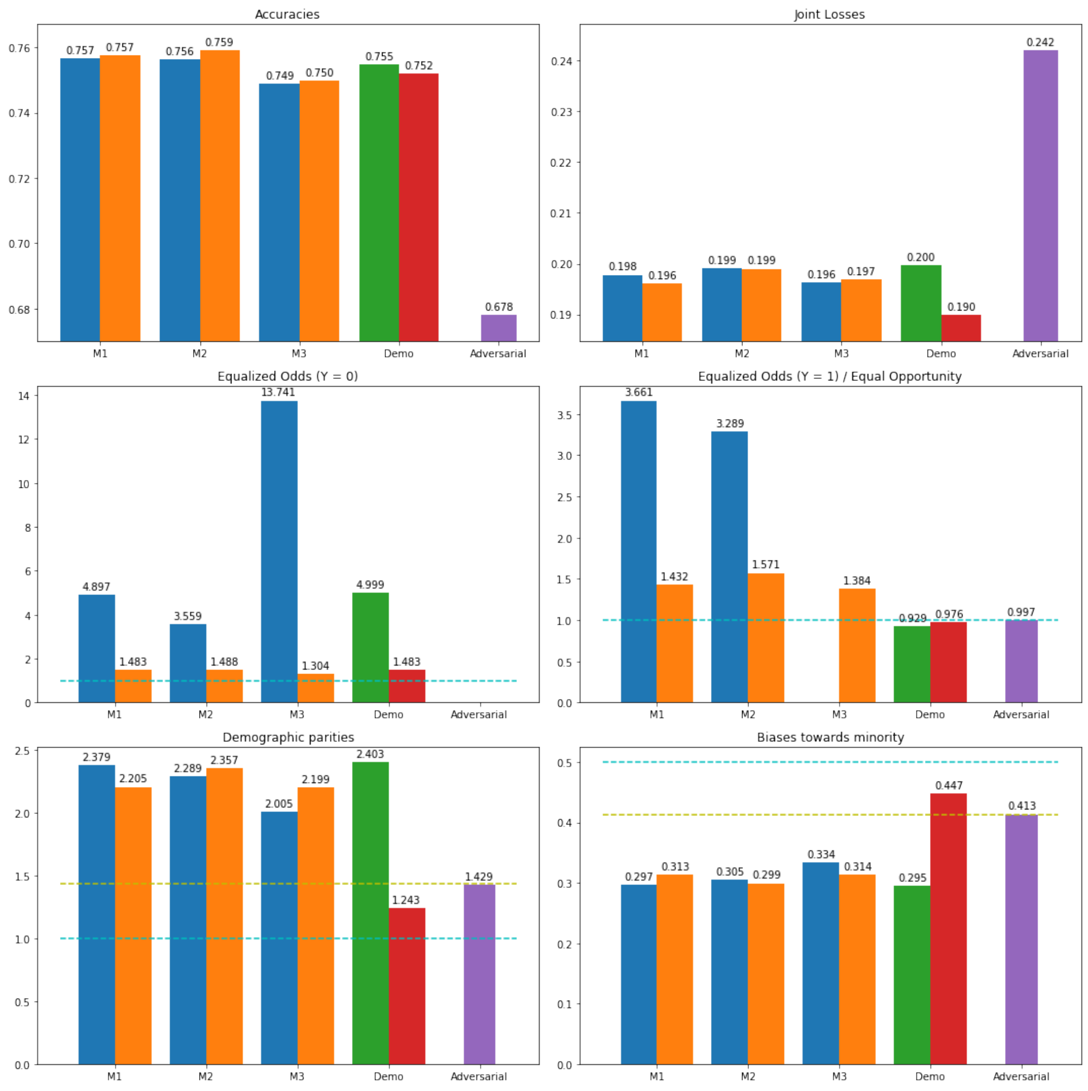


Figure 5.4: Results of the Compas dataset

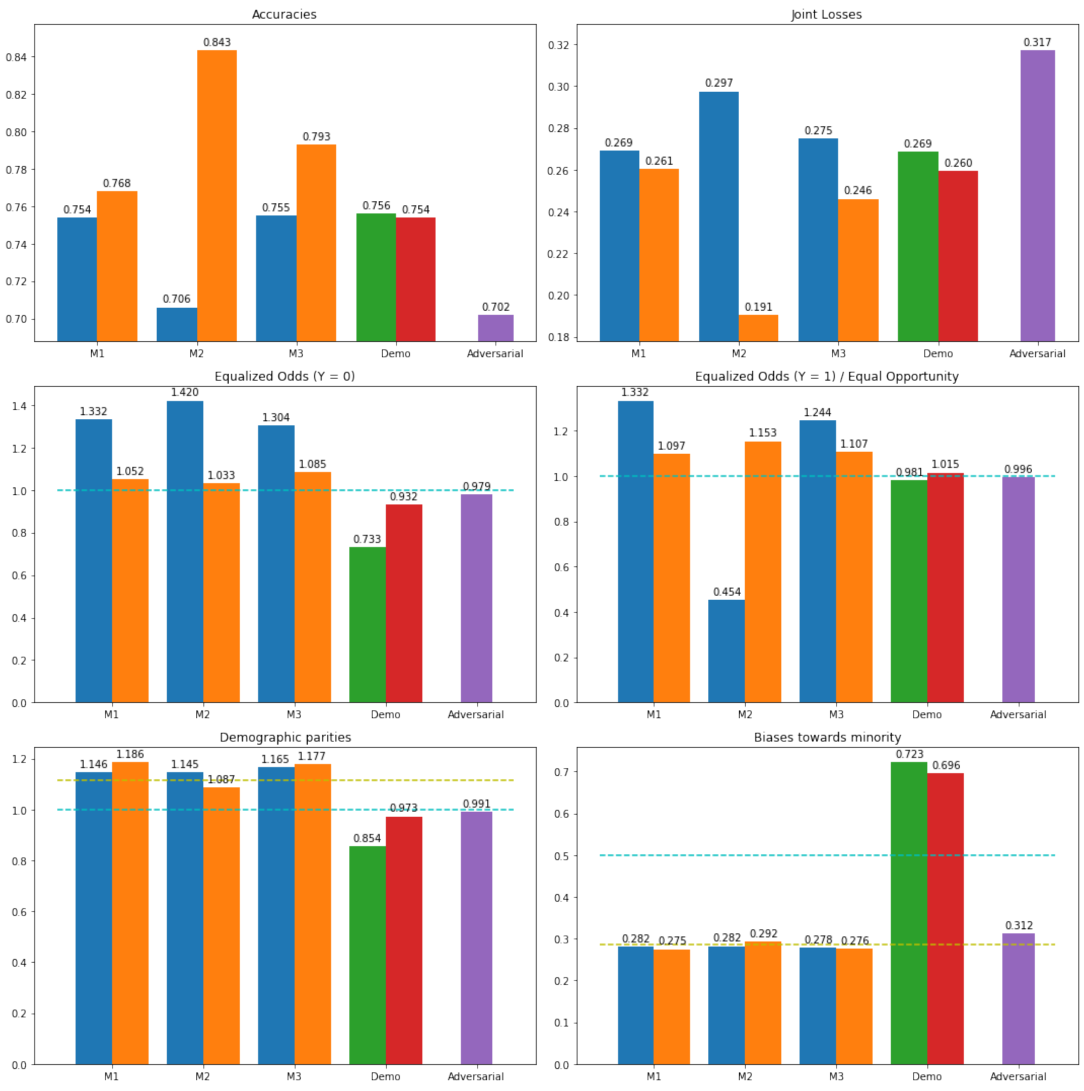


Figure 5.5: Results of the Credit dataset

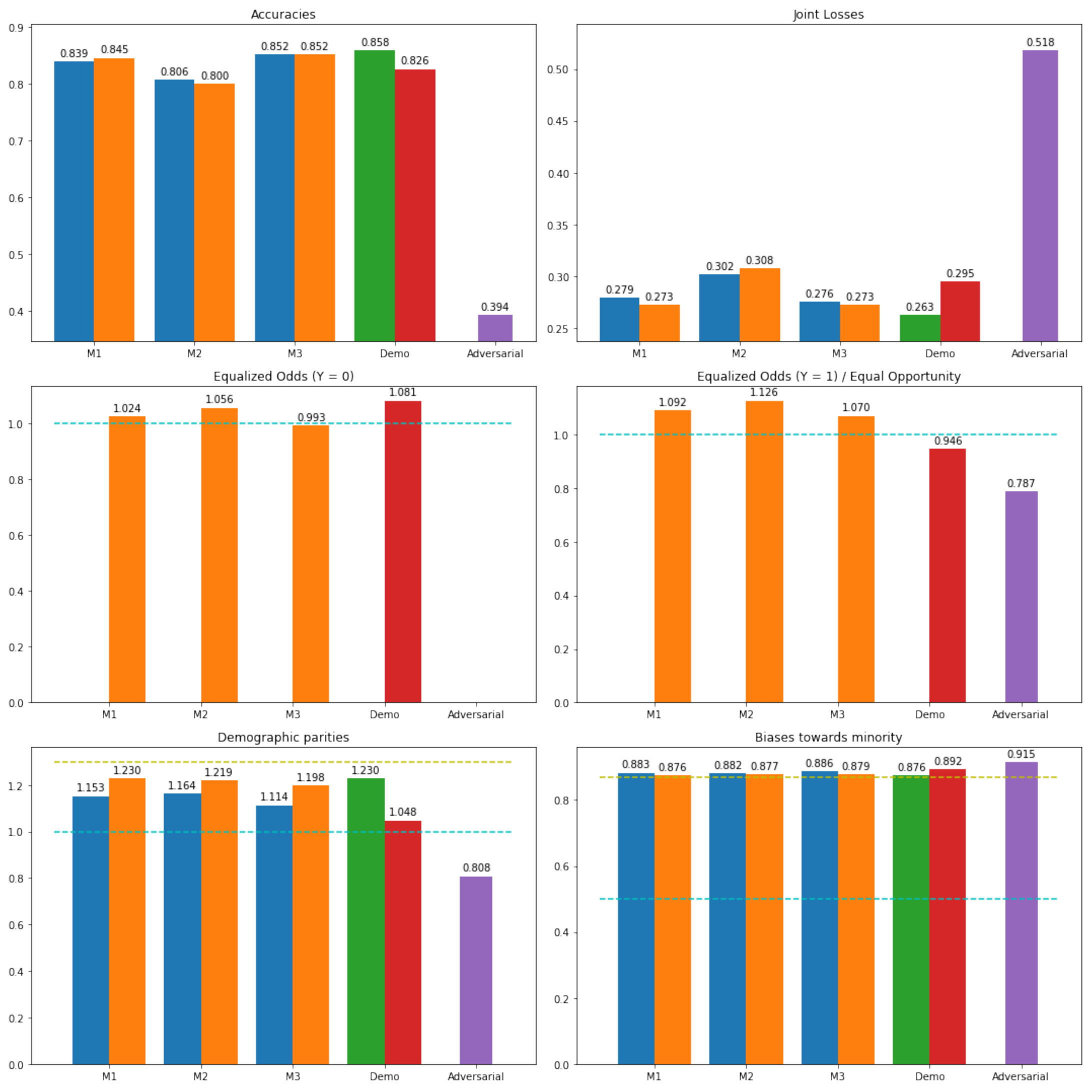


Figure 5.6: Results of the Hepatitis dataset

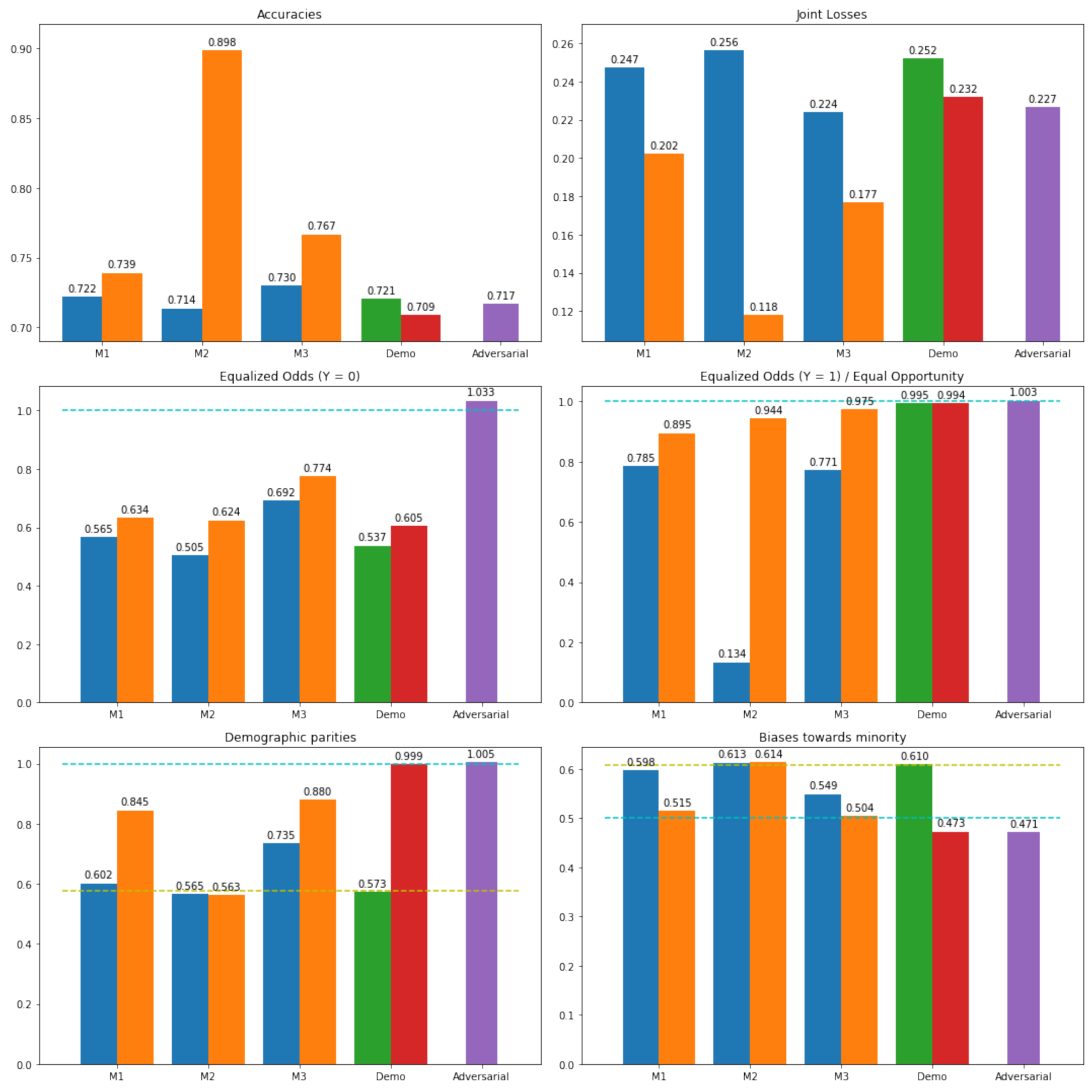


Figure 5.7: Results of the Infor dataset

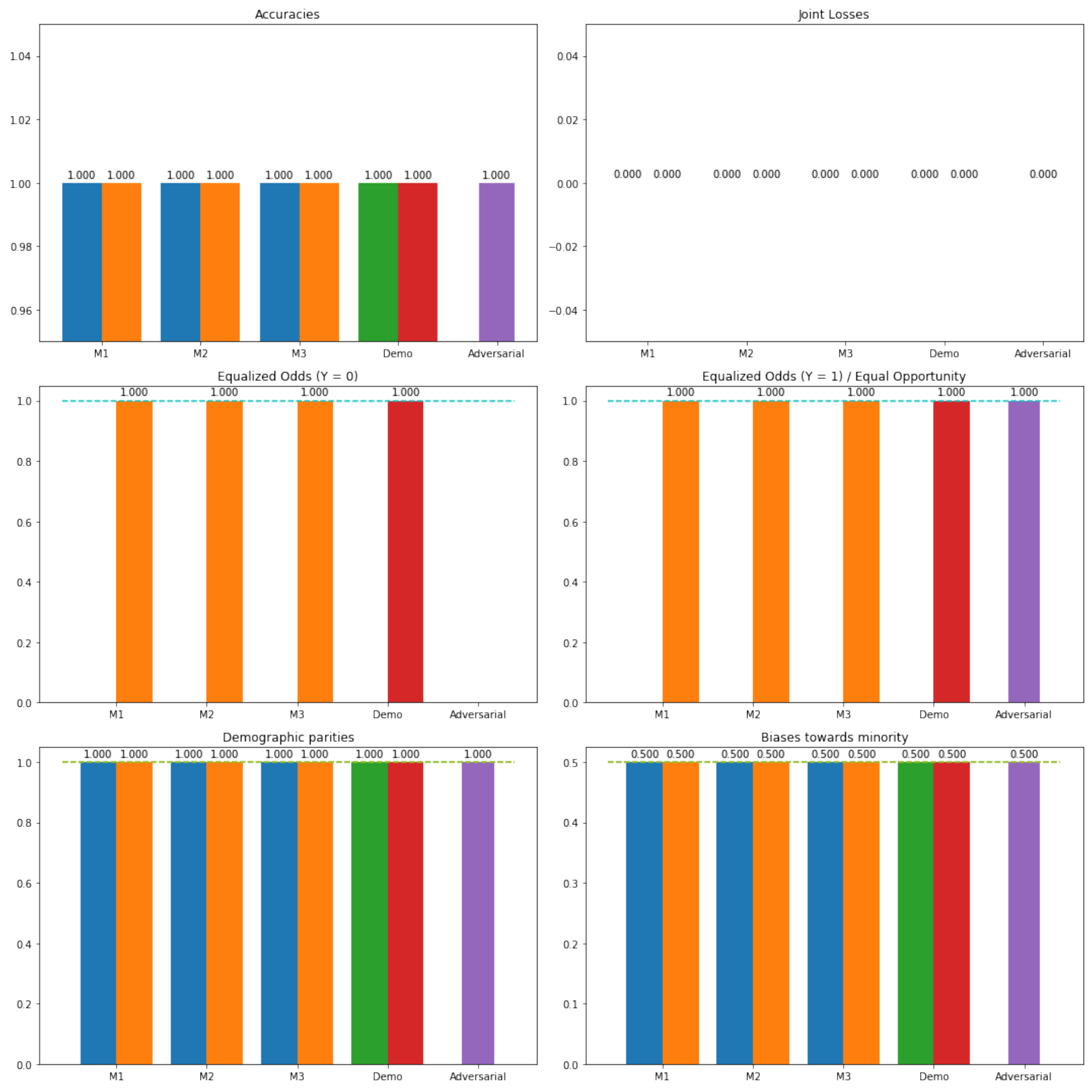


Figure 5.8: Results of the Nurse dataset

6. CHAPTER

Conclusions

In this report, we have incorporated state-of-the-art bias reducing techniques to different supervised classifiers. Their impact has been evaluated using several metrics that measure fairness in classification tasks. Multiple datasets with different characteristics have been used for testing our work to assess the suitability of the techniques in different scenarios.

Having carried out experiments in distinct datasets, we believe that the variety has served as a metric of how well the various models adapt to different circumstances. Having said that, we consider the constrained classifier the most appropriate method closely followed by the decoupled classifiers.

The biggest negative aspect of decoupled classifiers is the inconsistency with demographic parity. The difference with respect to their coupled respectives seems to not have any clear tendency if we compare results from different datasets. There are even datasets, e.g. Credit, where some models improve the metric while others do not. Nonetheless, decoupling shows a slight improvement in accuracy and a more remarkable improvement in equalized odds overall.

On the contrary, the constrained logistic regression seems to always improve fairness at a small cost of accuracy with respect to its unconstrained variant, the only exception being equal opportunity in the Bank dataset. Moreover, the results are competing with the ones of decoupled classifiers and are sometimes significantly better as it is in the case of demographic parity in many datasets.

However, if we were to choose one decoupled classifier, it would be the decoupled multilayer perceptron (M2) as it scores the best accuracy amongst all classifier in all experiments except for Hepatitis.

Adversarial models fall into the third place because they have shown to be rather inconsistent with the scores of their performance. For example, the adversarial model gets almost perfect fairness scores with the Credit dataset but then has the worst scores in Hepatitis. We believe this is a hyperparameter issue, as the authors of the technique propose a custom hyperparameter tuning for their experiment over the Adult dataset. In fact, our results on that dataset are relatively acceptable.

All in all, we believe that fairness constraints are the most appropriate technique for mitigating binary bias, where gender takes part, due to its adaptability to several datasets where the shape of data is not uniform or balanced or where the training set is limited in size. For example: the Hepatitis dataset has only 155 samples.

Furthermore, it is the unique technique whose authors have publicly released an implementation to an online repository. This allows the community to develop and maintain the code as well as track bugs. Also, anyone using the technique can wholly rely on the code, as it is an official release instead of a questionable implementation of an unknown user.

6.1 Personal conclusions

Personally, it has been an eye-opening experience. Although I was aware of the bias, I would not imagine how deeply the issue is tethered to the training of classifiers and how urgent an effective countermeasure is.

This work has shown me the importance of carefully collecting balanced and unbiased data. This involves our social responsibility for speaking, acting, treating, etc. in a neutral way, because models are going to reflect, if not amplify, even our minuscule prejudices.

Overall, I feel that neglecting this issue nowadays is a clear sign of severe irresponsibility for a fair and equal treatment of people.

6.2 Future work

One of the main assumptions we have made in this project has been binarizing both class and the sensitive attribute. However, key information might be lost by grouping different values together. For example, how should we group men, women and non-binary people?

On the other hand, there are many more proposals of new techniques that could be analyzed and probably some are being developed at this moment. Also, techniques may be fused together to form hybrid models that could improve our goal.

Appendices

A. APPENDIX

Transfer Learning experiment

This experiments shows the impact of using out-group samples for the minority group. The following results show different metrics in relation to parameter θ in the x axis, which controls the proportion of out-group samples that will be added to the training set. Accuracy, loss, demographic parity and equalized odds will be computed for every value of θ, ranging from 0 to 0.5 with a step of 0.05. We will analyze the Bank database for this experiment.

All figures show more or less the same tendency. Accuracy, demographic parity and equalized odds improve dramatically at first and then they flatten around $\theta = \frac{1}{3}$. Loss, on the other hand, steadily decreases and the shape of the curve is the main difference across the three figures.

All in all, the main conclusion to draw is that accuracy of the majority model is not improved at all with transfer learning, just the minority, and that after θ reaches one third, the improvement for each step is relatively small.

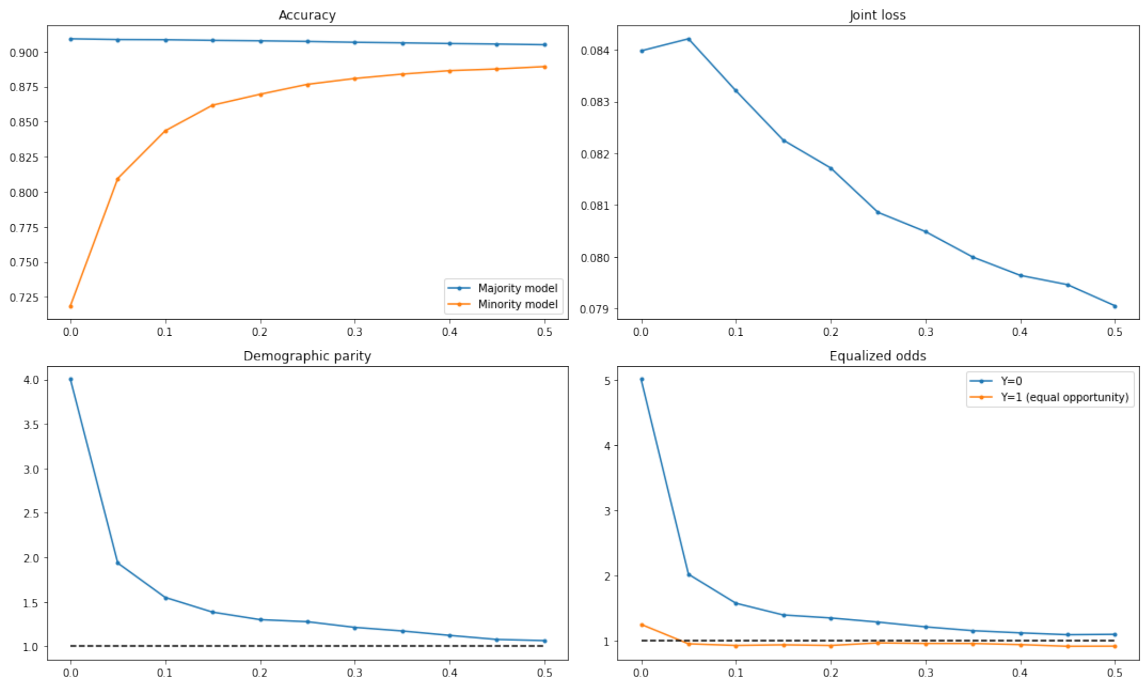


Figure A.1: Performances of the decoupled logistic regression with different transfer learning θ values.

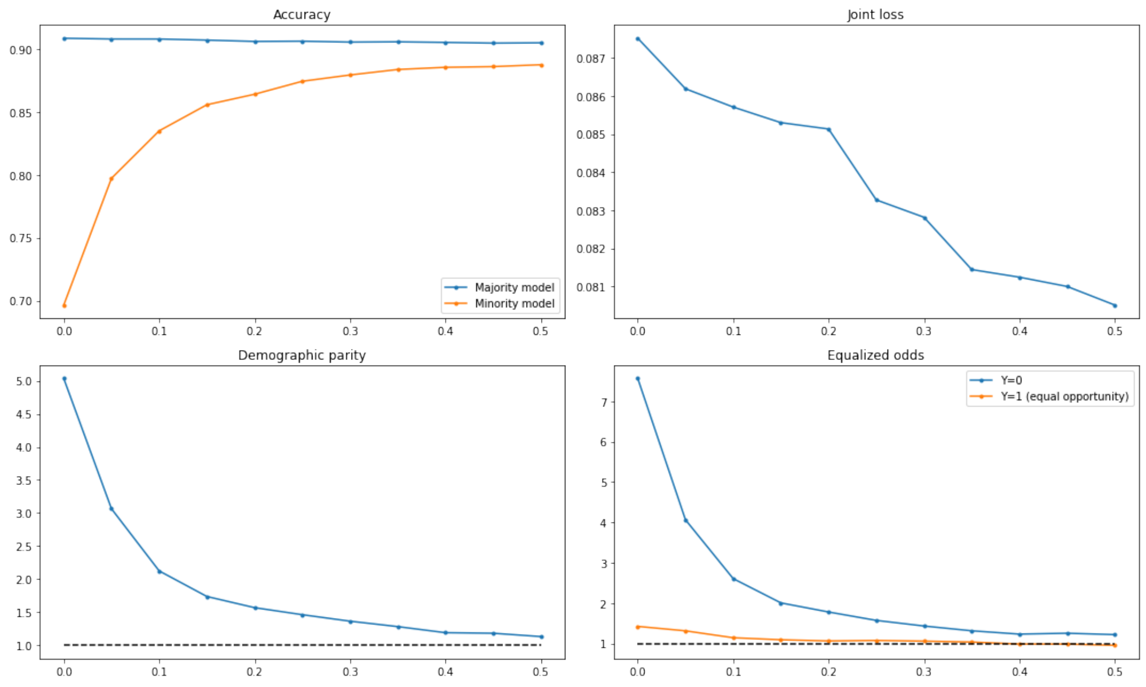


Figure A.2: Performances of the decoupled multilayer perceptron with different transfer learning θ values.

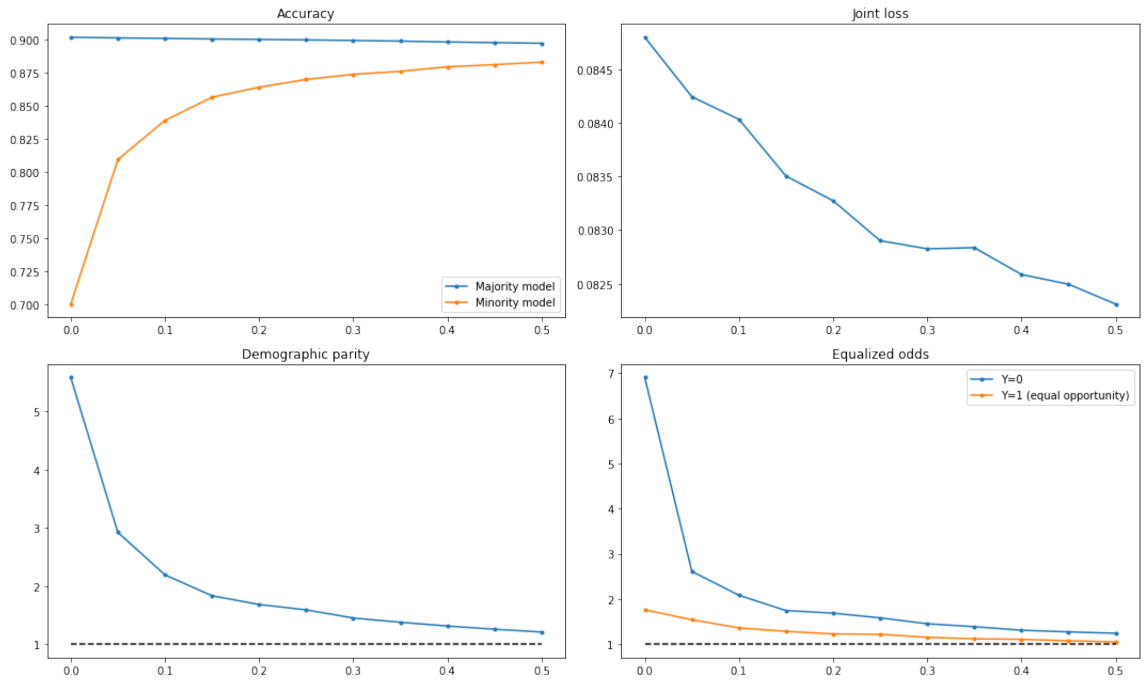


Figure A.3: Performances of the decoupled SVM with different transfer learning θ values.

B. APPENDIX

Database preprocessing details

The next page shows the table with all the conversions of the chosen columns as class and sensitive attribute for each database. Remember that the rest of attributes follow the conversion specified in section 5.1.1. Also, some attributes in Compas (section 5.2.3) and Hepatitis (section 5.2.5) have been modified or removed.

Database	Class	Conversion	Protected	Conversion
Adult	Last column	$> 50K \rightarrow 1$ $<= 50K \rightarrow 0$	"sex"	male $\rightarrow 1$ female $\rightarrow 0$
Bank	"y"	yes $\rightarrow 1$ no $\rightarrow 0$	"age"	$25 \leq adina \leq 60 \rightarrow 1$ otherwise $\rightarrow 0$
Compas	"is_recid"	not needed	"race"	*African-American* $\rightarrow 1$ otherwise $\rightarrow 0$
Credit	"class"	good $\rightarrow 1$ bad $\rightarrow 0$	"gender"	male $\rightarrow 1$ female $\rightarrow 0$
Hepatitis	"Class"	LIVE $\rightarrow 1$ DIE $\rightarrow 0$	"SEX"	female $\rightarrow 1$ male $\rightarrow 0$
Infor	"MarrazkiarenGeneroa"	EzEmakumezkoa $\rightarrow 1$ Emakumezkoa $\rightarrow 0$	"Sexua"	Mutila $\rightarrow 1$ Neska $\rightarrow 0$
Nurse	"class"	not_recom $\rightarrow 0$ otherwise $\rightarrow 1$	"finance"	convenient $\rightarrow 1$ inconv $\rightarrow 0$

Table B.1: The attributes chosen as protected and class for each database as well as the conversion applied.

Bibliography

[Amari and Wu, 1999] Amari, S. and Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783 – 789.

[Beutel et al., 2017] Beutel, A., Chen, J., Zhao, Z., and Chi, E. H. (2017). Data decisions and theoretical implications when adversarially learning fair representations.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, NY. Softcover published in 2016.

[Bolukbasi et al., 2016] Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., and Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 4349–4357. Curran Associates, Inc.

[Buolamwini and Gebru, 2018] Buolamwini, J. and Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. 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 77–91, New York, NY, USA. PMLR.

[Celis et al., 2019] 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*, FAT* '19, page 319–328, New York, NY, USA. Association for Computing Machinery.

[Dalvi et al., 2004] Dalvi, N., Domingos, P., Mausam, Sanghai, S., and Verma, D. (2004). Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 99–108, New York, NY, USA. Association for Computing Machinery.

[Daniel, 2013] Daniel, G. (2013). *Principles of artificial neural networks*, volume 7. World Scientific.

[Dwork et al., 2017] Dwork, C., Immorlica, N., Kalai, A. T., and Leiserson, M. (2017). Decoupled classifiers for fair and efficient machine learning.

[Gardner and Dorling, 1998] Gardner, M. and Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627 – 2636.

[Garg et al., 2018] Garg, N., Schiebinger, L., Jurafsky, D., and Zou, J. (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644.

[Gonen and Goldberg, 2019] Gonen, H. and Goldberg, Y. (2019). Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

[Hardt et al., 2016] Hardt, M., Price, E., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 3315–3323. Curran Associates, Inc.

[Johnson and Zhang, 2013] Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 315–323. Curran Associates, Inc.

[Karlik and Olgac, 2011] Karlik, B. and Olgac, A. V. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

[Koh et al., 2007] Koh, K., Kim, S.-J., and Boyd, S. (2007). An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine learning research*, 8(Jul):1519–1555.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Lee et al., 2006] Lee, S.-I., Lee, H., Abbeel, P., and Ng, A. Y. (2006). Efficient l˜ 1 regularized logistic regression. In *AAAI*, volume 6, pages 401–408.

[Lippmann, 1987] Lippmann, R. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4(2):4–22.

[Liu et al., 2009] Liu, J., Chen, J., and Ye, J. (2009). Large-scale sparse logistic regression. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 547–556, New York, NY, USA. Association for Computing Machinery.

[Manrai et al., 2016] Manrai, A. K., Funke, B. H., Rehm, H. L., Olesen, M. S., Maron, B. A., Szolovits, P., Margulies, D. M., Loscalzo, J., and Kohane, I. S. (2016). Genetic misdiagnoses and the potential for health disparities. *New England Journal of Medicine*, 375(7):655–665. PMID: 27532831.

[Pal and Mitra, 1992] Pal, S. K. and Mitra, S. (1992). Multilayer perceptron, fuzzy sets, classification.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations.

[Ruder, 2016a] Ruder, S. (2016a). An overview of gradient descent optimization algorithms.

[Ruder, 2016b] Ruder, S. (2016b). An overview of gradient descent optimization algorithms.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Slack et al., 2020] Slack, D., Friedler, S. A., and Givental, E. (2020). Fairness warnings and fair-maml: Learning fairly with minimal data. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 200–209, New York, NY, USA. Association for Computing Machinery.

[Smola and Schölkopf, 2004] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

[Werbos, 1990] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[Zafar et al., 2017] 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*, WWW '17, page 1171–1180, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

[Zafar et al., 2015] Zafar, M. B., Valera, I., Rodriguez, M. G., and Gummadi, K. P. (2015). Fairness constraints: Mechanisms for fair classification.

[Zhang et al., 2018] Zhang, B. H., Lemoine, B., and Mitchell, M. (2018). Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, page 335–340, New York, NY, USA. Association for Computing Machinery.

[Zhao et al., 2019] Zhao, J., Wang, T., Yatskar, M., Cotterell, R., Ordonez, V., and Chang, K.-W. (2019). Gender bias in contextualized word embeddings.

[Zhao et al., 2017] Zhao, J., Wang, T., Yatskar, M., Ordonez, V., and Chang, K.-W. (2017). Men also like shopping: Reducing gender bias amplification using corpus-level constraints.