



Bias mitigation for automated decision-making systems

Alice Morano

Supervised by Dr. Antonio Vetrò

Co-supervised by Elena Beretta

Department of Control and Computer Engineering
Computer Science
Polytechnic University of Turin

December 2020



Copyright ©2022 Polytechnic university of Turin

www.polito.it

First edition, March 14, 2022

Summary

The increasingly widespread availability and use of digital data - including personal data or relating to sensitive information - has inevitably contributed to the development of automated decision systems, such as ranking systems, recommendations systems, matching systems etc. The application of AI and ML brings definitely many advantages, particularly due to simplification and speeding-up of human complex tasks. Nonetheless, it should not be overlooked the fact that these systems lead as well to a wide range of critical aspects, such as data availability, data quality and features selection: indeed, the outcomes of ML algorithms will surely reflects the input. Hence, that is why data preparation is a crucial factor for an automated model building process and it requires relevant amount of time and resources.

Previous research has made progress in identifying various types of bias that can lead to discrimination in machine learning. Bias are data imbalance, which may enclose cultural/historical stereotypes, sampling or collecting issues, inaccurate labeling etc. These are sources of possible harmful information, that will be encoded by learning algorithms in the same way they gain and use helpful insights. Historical or cultural prejudices against some social groups are the kind of noxious information that we would not embed in automated processes, but at the same time we cannot help it, since they are inner part of data. For this reason and for the fact that nowadays ML and AI pervades many aspects of our society, it has been promoted the integration of ethical principles in the implementation of automated decision systems that may affect people's lives.

The contribution of our research to the delicate subject of fairness in ML lies in having structured an introductory synthesis of the principles of fairness, of bias types and tools needed to reduce discrimination in the use of data within these systems. Then, we performed experiments on two known datasets in the ML fairness field: the COMPAS dataset (for criminal recidivism score attribution) and a dataset with information related to drug consumption. Based on these data, we focused on the study of the fairness metrics and how we could improve fairness by means of *data balancing*, which can be considered a bias mitigation technique. Our results confirm the findings of previous research and highlight a possible process to maximize fairness, or at least to direct towards the choice of the most suitable metric for the evaluation of fairness in these two specific cases. The approach of bias mitigation acts as countermeasure for social inequalities, addressing to a less discriminatory outcome.

Last but not least, this research also aims to achieve a noble goal, specifically to advocate for the moral integrity of every person of science, who must bear in mind to have powerful tools to deal with, which could both build defenses and destroy as weapons, even when these are intangible.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Aims and Objectives | 2 |
| 2 | Background & Literature Overview | 3 |
| 2.1 | Fairness in Machine Learning | 3 |
| 2.1.1 | Sensitive data | 4 |
| 2.1.2 | Similarity-base measures | 6 |
| 2.1.3 | Causal reasoning | 6 |
| 2.1.4 | Preference-base measures | 7 |
| 2.2 | Social discrimination in the digital context | 8 |
| 2.2.1 | Automated Decision-making Systems (ADS or ADM) | 9 |
| 2.2.2 | Examples | 10 |
| 3 | Bias & Fairness | 15 |
| 3.1 | Bias taxonomy - past research | 16 |
| 3.1.1 | A Framework for Understanding Unintended Consequences of Machine Learning [49] | 16 |
| 3.1.2 | On Simpson's Paradox and the Sure Thing Principle [11] - Berkson's Bias, Selection Bias, and Missing Data [53] | 17 |
| 3.1.3 | Social Data: Biases, Methodological Pitfalls, and Ethical Boundaries [46] | 18 |
| 3.1.4 | Search Bias, Language Bias and Genetic Programming [54] | 20 |
| 3.1.5 | Bias amplification in Artificial intelligent systems [42] | 21 |
| 4 | Methodology background | 23 |
| 4.1 | Metrics | 23 |
| 4.1.1 | Diversity | 23 |
| 4.2 | Fairness | 24 |

| | | |
|-------------------|--|------------|
| 4.2.1 | Fairness - R library | 24 |
| 4.2.2 | Statistical metrics | 24 |
| 4.2.3 | Fairness metrics | 26 |
| 4.2.4 | Fairness criteria | 31 |
| 4.3 | Bias mitigation techniques | 35 |
| 4.3.1 | Pre-processing | 36 |
| 5 | Case studies | 39 |
| 5.1 | Datasets | 39 |
| 5.1.1 | COMPAS | 39 |
| 5.1.2 | Drugs | 59 |
| 6 | Conclusions | 69 |
| 6.1 | Achieved Aims and Objectives | 69 |
| 6.2 | Critique and Limitations | 69 |
| 6.3 | Future Work | 70 |
| Appendix A | R code | 71 |
| Appendix B | Data files | 153 |
| References | | 157 |

List of Figures

| | |
|---|----|
| 4.1 SMOTE visualisation - extract from [30] | 37 |
| 5.1 COMPAS categorical variables distributions | 43 |
| 5.2 IMAGE RECIDIVISM ON DIFFERENT COMBINATIONS | 44 |
| 5.3 COMPAS Pearson's Chi-square test | 45 |
| 5.4 COMPAS sensitivity and specificity curves | 46 |
| 5.5 COMPAS linear regression confusion matrix | 46 |
| 5.6 COMPAS predicted probabilities' densities | 47 |
| 5.7 COMPAS fairness metrics Age | 48 |
| 5.8 COMPAS fairness metrics ethnicity | 49 |
| 5.9 COMPAS fairness metrics Female | 50 |
| 5.10 Fairness metric values of ethnicity compared to: (a) Female and (b) Age. | 51 |
| 5.11 Shannon index comparison between original dataset and balances datasets. | 54 |
| 5.12 COMPAS fairness metric comparison between balanced datasets and original data. | 56 |
| 5.13 COMPAS fairness metric comparison between: (a) balanced_data2-balanced_data4 and (b) balanced_data6-balanced_data7. | 57 |
| 5.14 Mean deviations from complete fairness, for each fairness metric, for balanced classified datasets and original classified dataset | 58 |
| 5.15 Drug dataset categorical variable distributions | 60 |
| 5.16 % of coke consumption on: (a) total and (b) each sensitive attribute combination. | 62 |
| 5.17 % of crack consumption on: (a) total and (b) each sensitive attribute combination. | 62 |
| 5.18 Drug dataset Pearson's Chi-square test for Coke | 62 |
| 5.19 Drug dataset Pearson's Chi-square test for Crack | 62 |
| 5.20 Drug dataset Coke classification confusion matrix | 63 |
| 5.21 Drug dataset Crack classification confusion matrix | 63 |
| 5.22 Drug dataset Coke density probabilities | 65 |
| 5.23 Drug dataset Coke fairness metrics comparisons between Ethnicity VS (a) Age and (b) Gender. | 65 |

| | |
|---|----|
| 5.24 Drug dataset Crack fairness metrics comparisons between Ethnicity VS (a) Age and (b) Gender. | 65 |
| 5.25 Mean deviations from complete fairness, for each fairness metric in Coke case, for balanced classified datasets and original classified datasets | 67 |
| 5.26 Mean deviations from complete fairness, for each fairness metric in Crack case, for balanced classified datasets and original classified datasets | 68 |

List of Tables

| | | |
|-----|---|----|
| 2.1 | Overview of fairness definitions, adapted from [51] and [23]. Yellow color stands for independence; pink color for sufficiency; green color for separation. Darkest colors for equivalence; lightest colors for relaxed version. See chapter 4 for details. | 5 |
| 2.2 | Table of possible discrimination grounds. | 11 |
| 3.1 | Simpson's paradox example - extract from [11] | 18 |
| 5.1 | COMPAS dataset first rows | 41 |
| 5.2 | COMPAS dataset Yes_recidivism frequent combinations (first rows) | 42 |
| 5.3 | COMPAS dataset No_recidivism frequent combinations (first rows) | 42 |
| 5.4 | COMPAS ethnicity VS Female fairness metrics mean signed deviations (% values) | 51 |
| 5.5 | COMPAS ethnicity VS Age fairness metrics mean signed deviations (% values) . | 51 |
| 5.6 | Drug dataset first rows | 61 |

Introduction

1.1 | Motivation

In the era of big data, it is vitally important to take into consideration the social implications of the scientific research. The idea behind good science is the more discoveries made and more research done, the closer we will be to healing ourselves and the world. Hence, sometimes we are all so smitten by reaching results that we overlook the fact that progress could have consequences on the social sphere. In particular, it happens when science and technology deal with human sensitive data.

It is a fact that we live in a world of pervasive technology and we score a goal when we manage to replace human wearing efforts by automating tasks. However, the results of Artificial Intelligence and Machine Learning can bring us on the one hand to easier solutions, but on the other, to possible unequal conditions.

There are several well known cases - that became real scandals - of the combined use of data and technology, such as the Cambridge Analytica case in early 2018, for political advertising purposes [4]. Or the COMPAS software [5], used as a decision support tool by U.S. courts to determine criminal recidivism. Or again, the AI Amazon recruiting tool, biased against women [45].

In some cases, the outcomes were not intentionally discriminating categories of people, but they did, due to either data feedback loops or initial databases that reflected inner social prejudices.

Hence, when it comes to human factor, it is important to face the ethics implications of data analysis.

How we should implement algorithms for automated decisions or machine learning? Which data and which types of data should we take into account? Sometimes the legal, regulated aspects give us the answers (e.g. GDPR for data privacy), but in other circumstances, scientists should play a leading role in proper data usage and algorithms implementation, by using common sense.

We might think not to have any responsibility and treat data in the same way we treat any other bare ingredient of our work, as big black boxes, but this should not be the right approach towards sensitive data. What motivated us the most, is in fact the awareness that data are not only the “oil of the 21st century” [31], but also the identity of billions of human beings, on which we cannot speculate.

1.2 | Aims and Objectives

The aim of this thesis is to research cases of possible social inequalities, arose due to data usage in web platforms.

The objective is to give examples of bad data usage, in order to find some guidelines for avoiding indirect use of sensitive information in algorithms.

Background & Literature Overview

2.1 | Fairness in Machine Learning

Nowadays, artificial intelligence is getting more and more involved in fields that uses sensitive data, such as health care, hiring criminal justice, hence there has been a wider focus on the connections of bias and unfairness embedded in it. Some people may claim that using artificial intelligence for decision-making could get choices fairer, but this is a wrong assumption: indeed using massive data in automated systems has a double-edge. It may avoid procedural mistakes, but it reflects and amplify as well societal bias embedded in training datasets. In addition, it has to take into consideration the human decisions made during the machine learning development process, and the complex feedback loops that arise when a machine learning model is used in a real context.

Fairness is an articulated concept that depends on social context, legal aspects, different stakeholders, thus culture. It is impossible to determine a single and unique definition of fairness and it is difficult to formalize and quantify fairness principles mathematically. Nevertheless table 2.1 summarize 13 statistical measures for fairness, other formal definitions and 2 possible other context for fairness research (rows 23 and 24), which focus on different aspects of fairness and can produce entirely different outcomes. [44] proposes another classification of these fairness policies into:

- **Group fairness:** similar predictions within different groups.
- **Individual fairness:** similar predictions to similar individuals.
- **Subgroup fairness:** obtain the optimal trade-off between best properties of the group and individual notions of fairness. It selects a group fairness constraint like true positive rate parity and asks whether this constraint holds over a large collection of subgroups.

We will treat more in detail and technically some of these fairness definitions (statistical measures) in 4, while other main fairness concepts (similarity-base measures and causal reasoning)

are explained in the current chapter.

In Machine Learning, there are three main ways to improve fairness, as described in [55]: the first one examines methods to improve fairness by **optimizing algorithms**, at different phases of machine learning pipeline; the second one is about using **interactive systems**, such as HypDB (understand causality and the Simpson paradox) or Northstar, or AnchorViz (for interactive visualization, advanced analytics, classification etc. without the need for technical tasks); the third one involves **hybrid automatic and interactive systems**, such as IBM AI Fairness 360 or Google's What-If tool. In the latter case though, certain mechanisms implies assumptions that may not be appropriate in all data and model contexts. However, tools such as IBM AI Fairness 360 benefits from an open source library that can be used an customized for many different purposes.

2.1.1 | Sensitive data

The General Data Protection Regulation (GDPR) [1] is a regulation in EU law on data protection and privacy in the European Union (EU) and the European Economic Area (EEA), adopted on 14 April 2016, become enforceable on 25 May 2018.

The need of having a unified regulation for sensitive data in Europe takes its origin from the massive usage and request of personal or identifying data in online platforms. These data are commonly treated then with machine learning, artificial intelligence or big data tools.

A common sense is to quantify fairness via an associative relationship between the sensitive attributes and the outcomes of a classifier/predictor. For instance, in the United States, the Equal Employment Opportunity Commission defined the **four-fifths rule** [3] as a guideline (a good practice, not a legal definition) for comparing predicted positive outcome rates based on sensitive features (e.g. total number of hired people in a protected group, over total number of hired).

In GDPR we have a distinction between personal data and personal sensitive data. The first term is defined in Art. 4 and includes any information related to an identified or identifiable natural person, by means of which they can be directly or indirectly identified (name, identification number, location data, address, credit card, telephone number...). Sensitive personal data are instead defined in Art. 9 and they include genetic, biometric and health data, as well as personal data revealing racial and ethnic origin, political opinions, religious or ideological convictions or trade union membership. This kind of data are subject to higher level of protection.

In particular, according to the Italian privacy code [2], a narrow and closed list of data types was considered sensitive data:

- racial and ethnic origin
- religious, philosophical or other beliefs

Table 2.1: Overview of fairness definitions, adapted from [51] and [23].
 Yellow color stands for independence; pink color for sufficiency; green color for separation.
 Darkest colors for equivalence; lightest colors for relaxed version. See chapter 4 for details.

| | | | |
|----|--|---|--------------------------|
| 1 | Group fairness / Statistical parity / Equal acceptance rate / Bench-marking / Demographic parity | Based on predicted outcome | |
| 2 | Conditional statistical parity | | |
| 3 | Predictive parity / Outcome test | | |
| 4 | False positive error rate balance / Predictive equality | | |
| 5 | False negative error rate balance / Equal opportunity | Based on predicted and actual outcome | Statistical measures |
| 6 | Equalized odds / Conditional procedure / Accuracy equality / Disparate mistreatment | | |
| 7 | Conditional use accuracy equality | | |
| 8 | Overall accuracy equality | | |
| 9 | Treatment equality | | |
| 10 | Test-fairness / Calibration / Matching conditional frequencies | Based on predicted probabilities and actual outcome | |
| 11 | Balance for positive class | | |
| 12 | Balance for negative class | | |
| 13 | Well-calibration | | |
| 14 | Casual discrimination | | Similarity-base measures |
| 15 | Fairness through unawareness | | |
| 16 | Fairness through awareness | Individual fairness | |
| 17 | Counterfactual fairness | | |
| 18 | No unresolved discrimination | | Causal reasoning |
| 19 | No proxy discrimination | | |
| 20 | Fair Inference | | |
| 21 | Preferred impact | | Preference-base measures |
| 22 | Preferred treatment | | |
| 23 | Equality of resources ↔ equity | | |
| 24 | Equality of capability of functioning | | |

- political opinions
- membership of parties, trade unions, associations or organizations of a religious, philosophical, political or trade union nature
- the state of health and sexual life

For example, it has been clarified that social status, social benefits received, educational qualifications, training and work experience, the debtor's solvency, income received or assets owned are not included in the strict treatment reserved for sensitive data (but they are still protected by the privacy law).

2.1.2 | Similarity-base measures

On the account of table 2.1, we define the following similarity-base measures:

- **Causal discrimination:** a classifier satisfies this definition if it produces the same classification for any two individual with the exact same set of attributes.
- **Fairness through unawareness:** a classifier is said to achieve fairness through unawareness if sensitive attributes are not explicitly used in the prediction process.
Satisfying this principle is not sufficient to avoid discrimination if other connected data are available, for instance in [20] is shown an heuristic method to retrieve ethnicity attribute given surname and geolocalization.
- **Fairness through awareness:** a classifier satisfy this definition if it relies on the principle that similar subjects should have similar classification. In particular, the similarity is defined by a distance metric: for a set of subjects V , a distance metric between applicants $d_1 : V \times V \rightarrow \mathbb{R}$. We can formalize the awareness guideline by using the *Lipschitz condition* on the classifier, as in [18]: if we consider a classifier as a mapping from subjects to distributions D over outcomes with its own metric $d_2 : D \times D \rightarrow \mathbb{R}$, the *Lipschitz condition* requires that for any two individuals x, y that are at distance $d_1(x, y) \in [0, 1]$ mapped to distributions $D(x)$ and $D(y)$, respectively, holds the inequality $d_2(D(x), D(y)) \leq K \cdot d_1(x, y)$, for a certain constant value K . In other words, the distributions over outcomes observed by x and y are indistinguishable up to their respective space distances d_1 and d_2 .

2.1.3 | Causal reasoning

Causal reasoning is the process of research for causality that leads to an effect. This process has a leading role in discrimination avoidance, because it covers all the non-observational fairness concepts (i.e. the ones that do not depend only on the joint distribution of predictor, protected attribute, features and outcome, as defined in [38]), which can allow us to resolve

fairness conclusively, overcoming the inherent limitations of observational criteria. However, some researches proved that counterfactual reasoning may generate hindsight bias ad outcome bias [8], [48]. On the account of table 2.1, we define the following causal reasoning principles:

- **Counterfactual fairness:** this concept is strictly bounded to individual fairness, because it requires modeling counterfactuals on an individual level (e.g. answer to questions such as "What would have happened to me, if I had belonged to another subgroup?"). The formal definition of a counterfactually fair classifier is the following (adapted from [39]):

The classifier C is **counterfactually fair** if under any context attribute $X = x$, sensitive attribute $A = a$ and actual outcome $Y = y$

$$\mathbb{P}(C_{A=a} = y | X = x, A = a) = \mathbb{P}(C_{A=a'} = y | X = x, A = a) \quad \forall y \in Y \text{ and } a' \neq a \in A.$$

where the $C_{A=a}$ has to be interpreted as the outcome of the classifier C if A had taken value a .

- **No unresolved discrimination:** it means avoiding discriminatory influences between variables and sensitive attributes.

A **resolving variable** is a variable influenced by the sensitive attribute, in a manner that we accept as non-discriminatory. This implies that in a causal graph, any variable directly connected to the sensitive attribute should be non-resolving and any other variable should only be indirectly connected to the sensitive attribute by non-resolving variables.

- **No proxy discrimination:** it means avoiding proxy variables interference between possible sensitive attribute and predicted outcome connections. A **proxy variable** is a variable whose value can be used to derive a value of another variable. If there is no a direct connection between a proxy to a feature, unawareness can avoid proxy discrimination.

- **Fair inference:** it means avoiding illegitimate paths between sensitive attribute and predicted outcome.

An **illegitimate path** is a connection between sensitive attribute and predicted outcome through another variable, that does not make a logic sense.

2.1.4 | Preference-base measures

In [6] are defined two distinct notions concerning unfairness in a decision-making process: **disparate treatment** if the decisions are somehow based on the individual's sensitive attributes and **disparate impact** if the decision process outcomes unequally treat people with certain sensitive attribute values. Hence, we introduce preference-base measures as a relaxation of

two statistical measures: preferred treatment is the relaxed version for treatment equality and preferred impact is a relaxed version for demographic parity (or better, proportional parity). Accordingly to [56], if we denote with B_a as the fraction of benefits received by users sharing a certain value of the sensitive attribute $a \in A$, then we can define classifiers that satisfy these two fairness concepts as follow:

- **Preferred treatment:** a classifier C offers preferred treatment when $B_a(C_a) \geq B_a(C_{a'})$, $\forall a, a' \in A$. This means that the preferred condition is preserved if each group gets more benefits from its own group-conditional classifier than any other classifier. In this way no group members would feel treated better by switching their group membership. Note that, if a classifier C does not make group-base distinctions (i.e. $C_z = C$, $\forall a \in A$), as a group-conditional classifiers does, it by default satisfies preferred treatment.
- **Preferred impact:** a classifier C offers preferred impact over a classifier C' if it achieves higher group benefit for each sensitive attribute subgroup, namely $B_a(C) \geq B_a(C')$, $\forall a \in A$.

2.2 | Social discrimination in the digital context

How is it possible to have social discrimination in a digital context? It is a fact that when we use personal and sensitive data we may incur in discrimination. In particular, this issue occurs when data are mined and then processed by algorithms, which should relieve and speed up the manual human workload, as for artificial intelligence and, to be more specific, machine learning. In [7] Barocas and Selbst lists five mechanisms by which data mining may lead to disproportional outcomes; the sixth one is a human factor; the seventh one is a type that does not exclude the co-occurrence of the others:

1. Defining target variable, class labels and a model: indeed some of them could have a greater or lesser adverse impact on unprivileged groups. For instance, if we choose "arrive at work on time" as a label to predict if my employees are good, then people in disadvantaged economic conditions might be mistreated because they have to take public transports if they cannot afford a car. Besides, functional misspecification, such as choosing a linear model on a variable that depends quadratically on the inputs, could infer discrimination on the whole process.
2. Labelling examples for the training data: when data are labelled (as for spam emails identification), the labels might be given on a biased base or, if they are given manually, humans can make mistakes that can lead to deviated predictions.
3. Collecting training data: if we train our model on biased dataset, namely if we collected and used biased data, we may learn from a bias sample and reproduce the bias.

4. Feature selection: when we process data in AI, we have to make decisions about which attributes observe and involve in our analyses. This may have consequences on the protected groups if they are not well represented in the selected features.
5. Proxies: this is equivalent to the proxy discrimination in case of proxy variables, already described in this chapter.
6. Intentional discrimination: in some cases, it may happen that discrimination takes its origins from a malicious intent by the AI developers and this is not an aspect to be overlooked. For instance, if an organization wants to discriminate against pregnant women, it is pretty accurate a prediction of pregnancy based on women shopping purchases.
7. Feedback loops: a possible cause of unfair predictions, generated by the outcomes of a system, used again as inputs. If the first outputs are biased due to an algorithm systematic error or to one of the previous discrimination causes, then the second outputs will be biased at a higher rate. This cycle could be performed several times and increase exponentially the discrimination.

In [36], Kamiran et al. formalize a model of discrimination in decision making. We adapted it by using the same notation as in 4.2.2.

Kamiran claims that a classifier consists of three main parts:

1. A score function $r = R(X)$, where X does not include the sensitive attribute.
2. A discrimination **bias function**:

$$B(A) = \begin{cases} b & A = \text{privileged} \\ -b & A = \text{unprivileged} \end{cases}$$

3. The final decision function $Y = C(R(X) + B(A))$.

Therefore, the final score to which we submit the classifier will be $r^* = r \pm b$ and the final decision will be given by $C(r^*)$. The classifier C has to define a decision acceptance threshold θ and accept individuals for which $r^* \geq \theta$.

This discrimination model has serious implications: the bias is more likely to affect subjects that are already close to the decision threshold by their individual score r . Then, if a subject is far from the decision boundary, it may lead to deceptive conclusions, because adding or subtracting the discriminatory bias b will not change the final decision.

We will focus on bias in the next chapter.

2.2.1 | Automated Decision-making Systems (ADS or ADM)

Many companies have sought to automate their processes to achieve greater responsiveness and lower costs. Anyway, processes that involve high-level decision-making are usually left

to managers who make decisions based on the interpretation of the data itself. They are procedures in which initial decisions (in whole or in part) are delegated to a person or corporate organization, then they would use automatically enforced decision-making models to perform an action. The aim of ADS is to minimize human intervention in ongoing decision-making. Their aim is to sense situations, use codified knowledge, and react properly in the presence of people. Existing systems are derived from both artificial intelligence and decision-support tools, where they often involve both business rule processing and statistical analysis or algorithms. The Automated decision making systems are mainly used in business analytics and informatics to make decisions, such as:

- Recommending a decision to the ultimate decision maker responsible.
- Guiding a user through facts, legislation, policy etc.
- Supporting decision systems, providing useful information for the ultimate decision maker, on the whole decision process.
- Providing preliminary assessment for individuals, as a self-assessment tool.
- Entirely substituting a decision process, by fully automatizing it.

The last case is, by far, the most critical one: decision making by machines can lead to discrimination of certain groups of people or negatively impact the consumer market by triggering lock-in effects. A major challenge is indeed how to protect people's rights, especially with regard to data protection, and whether existing specific protection rules for the ones who use the ADS. Lastly, it should be discussed who is responsible if something does not work (algorithmic accountability) and who has the responsibility for supervising those who create and implement the algorithms (ADS auditing and control).

2.2.2 | Examples

The following table 2.2 summarizes cases of possible discrimination, related to direct or indirect use of sensitive data. For each ground, are listed known studies regarding cases of discrimination that actually happened and a brief description of them.

Table 2.2: Table of possible discrimination grounds.

| Ground | Type of discrimination | Description | Ref. |
|------------------------|------------------------|--|------|
| Search engines | Political orientation | A study on 4,556 undecided voters with different demographic characteristics in United States and India. It demonstrates that biased search rankings can shift the voting preferences of undecided voters by 20% or more. | [21] |
| | Gender | Gender unequal distribution in photos retrieved by Bing for the query "person" and for queries based on 68 character traits (e.g., "intelligent person") in four regional markets. | [47] |
| | Language | A research that shows that cultural stereotyped biases from textual data propagate to artificial intelligence (AI) technologies in widespread use. | [12] |
| Freelance marketplaces | Gender and Race | Gender and race are significantly correlated with worker evaluations on TaskRabbit and Fiverr. This study is on 13,500 worker profiles: It gathers information about workers' gender, race, customer reviews, ratings, and positions in search rankings. | [26] |
| Continued on next page | | | |

Table 2.2 – continued from previous page

| Ground | Type of discrimination | Description | Ref. |
|------------------------|------------------------|--|------|
| Resume search engines | Gender | Indeed, Monster, and CareerBuilder are tools that allow recruiters to proactively search for candidates based on keywords and filters. This study shows gender indirect discrimination that leads to disadvantage rankings for some candidates. | [14] |
| Ad delivery systems | Race | This study concerns the problem of raising questions as to whether Google's advertising technology exposes racial bias in society and how ad and search technology can develop to assure racial fairness. | [50] |
| Intimate platforms | Sexual orientation | Despite the fact that that romantic and sexual choices are understood as intensely personal, this article aims to analyse if intimate platform designers (such as Tinder, OkCupid or Grindr) can help alter troubling patterns of social interaction, without unduly interfering with individual intimate choices. | [32] |
| Continued on next page | | | |

Table 2.2 – continued from previous page

| Ground | Type of discrimination | Description | Ref. |
|----------------------------|------------------------|---|------|
| Social networks | Race | Twitter's automatic cropping tool algorithm appeared to be systematically favoring white faces. There have not been conducted formal studies yet about this issue, but Twitter company declared to have checked the absence of algorithmic biases before using the tool. This discovery was made fortuitously by Twitter users. | [29] |
| Language translation tools | Race | This simple experiment is based on translating 11 occupation titles from one gender-inflected language to another, taking into consideration German, Italian, Polish, Spanish and French. It shows that Google Translate systematically changes the gender of translations when they do not fit with stereotypes, especially when it comes to single-words translations, not placed in a context. | [37] |

Bias & Fairness

In chapter 2 we introduced the bias function to depict how bias can affect discrimination in a decision-making process. The bias function is a formal concept that we find in every estimator or decision rule and it is an objective property of the estimator. However, there is a distinction between the statistical formalization of bias and bias that we find for decision systems. We could deeply analyse bias in the statistical sphere, but this is not the aim of our research: we will limit the mention of the statistical bias to show how it is related to the decision process in machine learning.

As written in [17], "The bias of a learning algorithm - if it can be formulated explicitly - provides a specification for the desired behavior of the algorithm and clarifies the design and implementation of machine learning algorithms. [...] The bias of a learning algorithm is the persistent or systematic error that the learning algorithm is expected to make when trained on training sets". In [24] is demonstrated that, given a learning algorithm L , its average error at a point x on a sample of size m is equal to the squared statistical bias plus the variance:

$$\text{Error}(L, m, x) = \text{Bias}(L, m, x)^2 + \text{Variance}(L, m, x)$$

Therefore, a bias in a learning algorithm is an error and the objective should be to reduce it as much as possible. In the same paper, Dietterich and Kong distinguish between two types of ML algorithm bias:

- **Absolute bias:** an assumption by the learning algorithm that the target function to be learned is definitely a member of some designated set of functions (such as the set of linear discriminant functions or the set of boolean conjunctions).
- **Relative bias:** an assumption that the function to be learned is more likely to be from one set of functions than from another (e.g. the decision tree algorithms consider small trees before larger ones).

There is a relationship between ML absolute-relative bias and statistical bias: if the relative bias is strong, then the algorithm will have low variance; on the opposite, if it is weak, the

algorithm will have high variance. If the absolute bias is inappropriate (it does not contain any good approximations to the target function), the algorithm will have high statistical bias. However, we cannot properly analyse discrimination and fairness in ML by focusing on statistical bias or algorithmic ML bias: even though they might have an impact on sensitive data and personal information, usually they are not the main cause of discrimination, then we have to move further, on other more specific biases that we find in previous ML fairness studies.

In this chapter, we will analyse where and when errors and bias can occur during a decision process in machine learning. We will research for the causes and we will try to give a schematic taxonomy of the types of bias we may encounter in our following experiments, based on past research.

3.1 | Bias taxonomy - past research

3.1.1 | A Framework for Understanding Unintended Consequences of Machine Learning [49]

In this paper are described 6 sources of bias that may be incurred at different steps of the ML process:

1. **Historical bias:** when there is a misalignment between real world expectations and model outcomes. It requires understanding and studying the application and generation of data over time. Even if we have perfectly-measured features, they might still reflect historical factors, for instance conditions that we find only in poorer neighborhoods. Accordingly to this, we can even reflect the world perfectly, but still inflict harm on a population.
2. **Representation bias:** when defining and sampling a population still under development, then certain part of the input space are underrepresented and others are overrepresented. This bias may be due to two main reasons: (1) the sampling methods reach only a part of the population and (2) the target population has changed or is anyway different from the original training population.
3. **Measurement bias:** when measuring features and labels in a prediction problem, we add random noise. This might be caused by three main reasons: (1) different measurement process for different groups or (2) different data quality for different groups and (3) oversimplification of the classification model (e.g. selecting or having available too few features for a good prediction).
4. **Aggregation bias:** when a single model is used for all groups, which require in fact different specific models, due to different conditional distributions, backgrounds, cultures

etc. Usually with this kind of bias the model will be optimal only for the dominant population.

5. **Evaluation bias:** when the evaluation and/or test data for an algorithm does not represent the target population. Misrepresented test data lead to the development of models that are optimized only for a subset of the population.
6. **Deployment bias:** when there is a misalignment between the problem a model is intended to solve and the way in which it is actually carried out after deployment (e.g. when we adapt a specific model to a generic task).

3.1.2 | On Simpson's Paradox and the Sure Thing Principle [11] - Berkson's Bias, Selection Bias, and Missing Data [53]

There are two well-known paradoxes in fairness theories that may introduce bias, due to wrong data interpretation:

1. **Simpson's bias:** it is derived from Simpson's paradox, which claims that it is possible to have $\mathbb{P}(A|B) < \mathbb{P}(A|B')$ and at the same time $\mathbb{P}(A|BC) \geq \mathbb{P}(A|B'C)$, $\mathbb{P}(A|BC') \geq \mathbb{P}(A|B'C')$, where C and C' are two distinct groups.

To be more clear, in table 3.1 are shown data regarding a new medical treatment on local patients (C) and patients from Chicago (C'). If we look at the overall treatment data, we are prone to say that it is a bad treatment compared to the standard one (Table a: 11% vs 46% alive people), but if we look at data divided per groups, we would notice that the treatment seems paradoxically good (Table b: higher rate of alive people in both groups).

2. **Berkson's bias:** this is based on the Berkson's paradox, which says that given two independent events, if you take into account only outcomes where at least one occurs, then they become negatively dependent. In terms of probability it becomes: $0 < \mathbb{P}(A) < 1$, $0 < \mathbb{P}(B) < 1$ and $\mathbb{P}(A|B) = \mathbb{P}(A)$, then $\mathbb{P}(A|B, A \cup B) < \mathbb{P}(A|A \cup B)$.

In general, Berkson says that if we only focus on a portion of data, we will be falsely tempted to say that there is a negative correlation between two attributes (e.g. the assumption that all Hollywood movies are bad, compared to their books, is due to the fact that we are just considering books and movies that overcome a certain "goodness" threshold, overlooking all bad movies and bad books).

Table 3.1: Simpson's paradox example - extract from [11]

| | | (a) | |
|--------|--|------------|------------|
| | | Treatment | |
| | | Standard | New |
| Dead: | | 5950 | 9005 |
| Alive: | | 5050 (46%) | 1095 (11%) |

| | | (b) | |
|--------|--|-----------------|------------------|
| | | C patients only | C' patients only |
| | | Standard | New |
| | | Standard | New |
| Dead: | | 950 | 9000 |
| Alive: | | 50 (5%) | 1000 (10%) |
| | | | 5000 (50%) |
| | | | 95 (95%) |

3.1.3 | Social Data: Biases, Methodological Pitfalls, and Ethical Boundaries [46]

This paper defines biases that can be encountered when analyzing social data, in particular taken from social platforms. From 1 to 6 are listed biases of selection phase (**selection bias**), depending for instance on the choice of the social platform from which to collect data. We report here the exact definitions of the paper:

1. **Population bias:** differences in demographics or other user characteristics between a population of users represented in a dataset or platform and a target population.
 - Different socio-demographic groups behave differently on social platforms -> we cannot generalize the behavior of a group for the whole of society.
 - Some studies use proxy for prediction, but proxies for personal traits have different reliability across different demographic groups (e.g. if we use political twitter posts to understand political orientation, politicians will get much higher accuracy than normal users).
2. **Behavioral bias:** differences in user behavior across platforms or contexts, or across users represented in different datasets.
 - **Interaction biases:** concern the different ways users interact with each other in social platforms, due to different personalities and different generations.

- **Content consumption biases:** regard different kind of contents interactions for different users, due to disparate interests and pool of contents. They may be expressed as well as lexical, syntactic, semantic, and structural differences in the contents generated by users.
 - **Linking biases:** behavioral biases that are expressed as differences in the attributes of networks obtained from user connections, interactions or activity. They regards external factors and attributes that may influence and change users' behaviour.
 - **Response biases:** are due to "missing" reports, inaccurate self-reports made by users. Most of the time, people lie about themselves or are not accurate regarding the information they provide.
3. **Temporal variations:** differences in populations or behaviors over time. This concept is similar to the historical bias. It is connected as well to behavioral biases, because over time users' behavior change rapidly on social platforms.
4. **Functional biases:** biases that are a result of platform-specific mechanisms or affordances, that is, the possible actions within each system or environment. They are strictly related to behaviour or population, but through the social platform design.
- introducing a new feature, or changing an existing feature on a platform, impacts usage patterns.
 - users react and behave in different ways, based on what they see and how they see, namely not only the contents, but also the order and disposition of the information they access (e.g. ranking algorithms).
5. **Normative biases:** biases that are a result of written norms or expectations about unwritten norms describing acceptable patterns of behavior on a given platform.
- on different platforms we may face different terms and conditions and simply different codes of conduct, hence people have to act in different ways.
 - users might be influence by other users' impact.
 - herding effect (influence of previous ratings and reviews), friendship paradox (popular individual can steer the behaviour of common users), anchoring effect (tendency to conform), male names for users and organizations could have a stronger an more positive impact...
6. **External biases:** biases resulting from factors outside the social platform, including considerations of socioeconomic status, ideological/religious/political leaning, education, social pressure, privacy concerns, topical interests, language, personality, and culture.
- socio-cultural elements that are reflected on social platform data.

- elements with high social impact (e.g. media communications, disasters, global pandemics...) are reflected and have a strong influence on social media.
 - different kind of contents are spread in different ways throughout social platforms, both due to automated mechanisms and human curation.
- 7. Data collection biases:** biases introduced due to the selection of data sources, or by the way in which data from these sources are acquired and prepared.
- data acquisition.
 - data querying (APIs).
 - data filtering.
- 8. Data processing biases:** biases introduced by data processing operations such as cleaning, enrichment, and aggregation.
- normalization of geographical references.
 - inserting default options.
 - automatic processes for data enrichment (e.g. made with machine learning).

3.1.4 | Search Bias, Language Bias and Genetic Programming [54]

As we already said at the beginning of this chapter, when bias regards ML sphere, it is a matter of choice an hypothesis over many other: we have to address a system towards a solution, even if we do not know whether it exists or not. This is called *heuristic knowledge* and it aims to reach reasonable generalisation, with limited information.

This paper defines three major kind of biases, with reference to the hypothesis choice:

1. **Selection bias:** it generates when we have to choose an hypothesis over another. For instance, in terms of evolutionary biology, this type of bias is the result of a fitness selection. Besides, every parameter and measure that we choose to represent an individual is a selection bias.
2. **Language bias:** in the article, Whigham claims that if we have an exiguous language pool, the hypothesis we can create might be restricted as well. By language here we mean all the descriptive tools we need to formulate a decision process.
3. **Search bias:** it consists in the process of turning one hypothesis into another, in order to search for the best one. It can be represented by the crossover during a random forest algorithm and the maximum depth of any created tree.

3.1.5 | Bias amplification in Artificial intelligent systems [42]

This paper divides bias in 5 main categories. We report here the exact definition the authors gave for each of them:

- **dataset bias:** it refers to bias that results from a too-small or too homogenous dataset, which then perpetuates inaccurate generalizations. This kind of bias concerns any preliminary issue related to the dataset.
- **association bias:** it occurs when data used to train an algorithm reinforces and multiplies cultural bias, when applied to a larger problem. (e.g. even due to unintentional discrimination, made by feedback loops based on cultural inference, it may be possible that some job advertisements are more likely to be shown to males than females).
- **automation bias:** describes situations in which the AI fails to take social or cultural factors into consideration (e.g. some machines may enforce the European stereotype of beauty by selecting greatly light skin rather than dark-skinned subjects).
- **interaction bias:** occurs when machines are empowered to learn without safeguards in place to identify and exclude harmful or pernicious beliefs (e.g. a bot that learns what to say by its users, may end up learning offensive or dangerous language if not controlled).
- **confirmation bias:** occurs when information is oversimplified or personified and makes improper generalizations or assumptions about a group or individual (e.g. in online purchasing there are often suggestions such as "you may also like these products...", but it is not rare that they mispredict what we like).

Methodology background

4.1 | Metrics

4.1.1 | Diversity

The diversity indicates abundance or lack of different species in a community (dataset) and we chose the **Shannon index** as a measure for it.

The Shannon index takes its origin from ecology, where it is also known as Shannon's diversity or Shannon's entropy. The idea beyond this index is that the more different individuals there are and the more equal their proportional abundances (number of individuals per species) in the category/cluster of interest, the more difficult it is to correctly predict the next individual in the category. The aim of this index is indeed to quantify the entropy associated with this prediction.

The Shannon index takes into consideration both the species richness (number of species present) and species abundance.

$$H' = - \sum_{i=1}^R p_i \ln p_i$$

where p_i is the proportion n/N of n individuals of one particular species i , divided by the total number of individuals found N ; \ln is the natural logarithm, R is the sum of the calculations, and s is the number of species.

With this index we quantifies the uncertainty in predicting the species identity of an individual chosen randomly from the community.

The greater is the difference between the abundances of the types, the smaller is the corresponding Shannon index value. If hypothetically there is only one species in the dataset, Shannon diversity exactly equals zero (no uncertainty in predicting the type of the next randomly chosen individual). An important notice is that if the true diversity doubles between two experiments, it does not mean that the index doubles as well: we can only evaluate a comparative measure, indeed whether an experiment gets more or less diversity than another.

4.2 | Fairness

In this section we will give an outlook of the main fairness metrics we used in order to evaluate the fairness level of our classification algorithm. It is important to stress that fairness in ML could be evaluated in many different ways and it also depends on anti-discrimination laws that states in different countries, concerning the usage of sensitive attributes. In our case, we focused on the fairness metrics in "fairness" CRAN package.

4.2.1 | Fairness - R library

The fairness R package provides the calculation and comparison of commonly used fairness measures across population subgroups.

By using this library we could assess our predictive model whether its results reinforce existing social biases. We identified the sensitive attribute that gave the highest disparate impact on the subjects of each dataset, then we evaluated and compared the fairness metrics values for this attribute.

In the below section 4.2.3 we listed all the fairness metrics offered in R Fairness package that are the ones we relied on for our research. For all the metrics, the ideal goal is to reach an equal value (parity) in each subcategory, which represents a perfect fairness situation. Since we are not virtually able to reach exactly the same values, we quantify how much a dataset lacks of fairness towards the sensitive attribute, by calculating the **mean signed deviation** of the subcategories values from the objective fair value.

4.2.2 | Statistical metrics

Sometimes it happens that, in order to avoid disparate treatment, we might be led to use a classifier with a distribution that does not take into account the sensitive attributes (this is called **unawareness**, i.e. $C = c(x, A) = c(X)$), but it has been proved that this strategy does not avoid discrimination. In fact, there could be variables strictly related or correlated to the sensitive ones (e.g. postal code related to ethnicity), that serve as **proxies** for the sensitive attributes.

All the following definitions refers to a binary classification under these assumptions:

- ◊ $X \in R^d$: quantified features of the individual (e.g. number of priors, misdemeanor...).
- ◊ $A \in \{unprotected, protected\}$: a binary sensitive attribute (e.g. Male/Female).
- ◊ $C := c(X, A) \in \{negative, positive\}$: binary predicted outcome (e.g. rejected or hired, which makes decision based on a score $R := r(x, a) \in [0, 1]$).
- ◊ $Y \in \{negative, positive\}$: target variable with the actual outcome (e.g. if the individual truly rejected or hired).

- ◊ X, A, Y generated from an elemental distribution D , namely $(X, A, Y) \sim D$.
- ◊ The best accuracy is gotten with $C(X, A) = Y \in (X, A, Y) \sim D$.
- ◊ We use the notation $\mathbb{P}_{protected}[c] := \mathbb{P}[C = c | A = protected]$ for the conditional probability of being classified as c , given that .

When we want to describe the performance of supervised learning classification model, we usually rely on a **confusion matrix**, which is a particular contingency table with rows and columns that represent respectively the instances of the predicted and actual classes. Predictive and actual classes can assume *positive* or *negative* value if we are using a binary classifier. On the main diagonal in the confusion matrix, we find the correct predictions, whereas on the secondary diagonal the prediction errors.

In the confusion matrix cells we find the following data, that will help understanding the fairness metrics:

- **TP - true positive:** both predicted and actual outcome are in the positive class.
- **FP - false positive:** predicted is in the positive class, but actual outcome is in the negative.
- **FN - false negative:** predicted is in the negative class, but actual outcome is in the positive.
- **TN - true negative:** both predicted and actual outcome are in the negative class.

From these four cases, we can derive the following metrics:

- **PPV - positive predictive value:** it is the probability of an individual with a positive prediction to truly belong to the positive class $\mathbb{P}(Y = positive | C = positive)$. Alternatively, it is the fraction of correctly predicted positive cases, out of all predicted positive cases $\frac{TP}{TP+FP}$. It is also named **precision**.
- **FDR - false discovery rate:** it is a false acceptance, namely a wrong prediction of a truly negative case to be in the positive class $\mathbb{P}(Y = negative | C = positive)$. Alternatively, it is the fraction of incorrectly predicted negative cases (that truly belong to the positive class) out of all predicted positive cases: $\frac{FP}{TP+FP}$.
- **FOR - false omission rate:** it is an incorrect rejection, namely wrong prediction of a truly positive case to be in the negative class $\mathbb{P}(Y = positive | C = negative)$. Alternatively, it is the fraction of incorrectly predicted positive cases (that truly belong to the negative class), out of all predicted negative cases: $\frac{FN}{TN+FN}$.

- **NPV - negative predictive value:** it is the probability of an individual with a negative prediction to truly belong to the negative class $\mathbb{P}(Y = \text{negative} | C = \text{negative})$. Alternatively, it is the fraction of correctly predicted negative cases, out of all predicted negative cases $\frac{TN}{TN+FN}$.
- **TPR - true positive rate:** it is the probability of truly positive cases to be predicted correctly $\mathbb{P}(C = \text{positive} | Y = \text{positive})$. Alternatively, it is the fraction of correctly predicted positive cases, out of all actual positive cases $\frac{TP}{TP+FN}$. It is also named **sensitivity** or **recall**.
- **FPR - false positive rate:** it is a false alarm, namely wrong positive prediction of an actual negative case $\mathbb{P}(C = \text{positive} | Y = \text{negative})$. Alternatively, it is the fraction of incorrectly predicted negative cases, out of all actual negative cases $\frac{FP}{FP+TN}$.
- **FNR - false negative rate:** it is the probability of truly positive cases to be predicted incorrectly as negative $\mathbb{P}(C = \text{negative} | Y = \text{positive})$. Alternatively, it is the fraction of incorrectly predicted positive cases out of all actual positive cases $\frac{FN}{TP+FN}$.
- **TNR - true negative rate:** it is the probability of a truly negative case to be predicted as negative $\mathbb{P}(C = \text{negative} | Y = \text{negative})$. Alternatively, it is the fraction of correctly predicted negative cases, out of all actual negative cases $\frac{TN}{FP+TN}$.

4.2.3 | Fairness metrics

4.2.3.1 | Demographic parity

Demographic parity (also known as Independence or Statistical parity) is achieved when the percentages or rates of the positive outcomes are equal for each group.

This metric only relies on the predicted outcome, not the actual one. In terms of probability, it means that the classifier C is independent from the sensitive attribute A . We can get demographic fairness simply by equalize the number of classified positive (negative) outcomes for each subcategory of the protected attribute:

$$\#_{\text{protected}}[C = c] = \#_{\text{unprotected}}[C = c] \quad \forall c \in \{\text{negative}, \text{positive}\}$$

Usually both demographic parity and proportional parity are formulated in terms of *positive* predictions. However, since we do not have a comparison between actual and predicted outcome, the equality of positive predicted is equivalent to the equality of negative predicted. We can not formulate this principle in terms of statistical metrics (confusion matrix), because

it does not take into consideration a comparison between predicted and actual outcome.

In `fairness` package:

this metric is evaluated by counting the sum of positive outcomes for each subcategory: the base category will get 1 as default value for positive outcomes, whereas all the other categories will be compared to this one, namely $\frac{\text{\#positive outcomes for category}}{\text{\#number of positive outcome for base category}}$. In this case, the different number of individuals per category has not been taken into account, then it makes sense to use this metric only when we have a perfectly balanced dataset in terms of the sensitive attribute. In `fairness` package, for all the following described metrics we will consider a base subcategory (e.g. 'Caucasian' for ethnicity attribute or 'Male' for gender attribute) and this base category will always get 1 as fairness metric value. All the other categories will get lower or higher values: this R package will give us both the absolute value of the different metrics and the comparative value towards the base category, but it only gives a ready-plot with the comparative measures.

4.2.3.2 | Proportional parity

Proportional parity is similar to demographic parity (or another way to indicate demographic parity) because for both metrics we do not compare the predicted outcomes to the actual outcomes. The difference is that this metric is much more useful when we have an unbalanced dataset towards the sensitive attribute.

In fact, for this metric we consider also the number of individuals for each population. In terms of probability, we can formulate it as follow:

$$P_{protected}[C = c] = P_{unprotected}[C = c] \quad \forall c \in \{negative, positive\}$$

In `fairness` package:

instead of using the `sum()` function (as it is used for the demographic parity), the `mean()` function gives the metric value, so that the less populated subcategories could anyway reach the same metric value as the highest populated ones.

Differently from the previous two metrics, all the following ones take into account both the actual and the predicted values.

4.2.3.3 | Equalized odds

This metric is also known as Separation and it is achieved if the **sensitivities** (true positives divided by all positives) in the subgroups are close one another.

With a formal definition [27], we say that a predictor C satisfies *equalized odds* with respect to protected attribute A and outcome Y , if C and A are independent conditional on Y . That means equalized odds may allow C to depend on A but only through the target variable Y . In terms of probability:

$$P_{protected, Y=y}[C = \text{positive}] = P_{unprotected, Y=y}[C = \text{positive}] \quad \forall y \in \{\text{negative}, \text{positive}\}$$

Since we introduce dependency on the sensitive attribute, it goes without saying that we can not both satisfy equalized odds and demographic parity at the same time. Depending on the y value, we equalize *true positive rates* or *false positive rates*, hence this metric is quite restrictive because it requires that both rates should be equal in order to get fairness.

In terms of statistical measures (confusion matrix), we can formulate this metric as follow:

$$TPR_{protected} = TPR_{unprotected} \text{ and } FPR_{protected} = FPR_{unprotected}$$

4.2.3.4 | Equality of opportunity

This metric is a weaker version of the Equalized odds.

4.2.3.5 | Predictive rate parity

This parity is also known as **Sufficiency** and it checks if the **precision rates** are equivalent for all subgroups under consideration [51], so that the fraction of correct positive predictions should be the same for each sensitive attribute value.

In terms of probability, this means:

$$P_{protected, C=\text{positive}}[Y = y] = P_{unprotected, C=\text{positive}}[Y = y] \quad \forall y \in \{\text{negative}, \text{positive}\}$$

A classifier with equal PPV has also equal FDR, that is why we can use indistinctly all y values to equalize the probabilities. Ineed, in terms of statistical measures (confusion matrix), we can formulate this metric as follow:

$$PPV_{protected} = PPV_{unprotected} \text{ or } FDR_{protected} = FDR_{unprotected}$$

4.2.3.6 | Accuracy parity

The accuracy represents the fraction of predictions that a classifier gets correctly, therefore accuracy parity is obtained when all the sensitive attribute groups have equal prediction accuracy. The accuracy is the probability of a subject -in either positive or negative class - to be assigned to its respective class: namely, true negatives are as desirable as true positives for this fairness metric.

$$P[C = Y, \text{protected}] = P[C = Y, \text{unprotected}]$$

In terms of statistical measures (confusion matrix), we can formulate this metric as follow:

$$\left(\frac{TP + TN}{\# \text{of predictions}} \right)_{\text{protected}} = \left(\frac{TP + TN}{\# \text{of predictions}} \right)_{\text{unprotected}}$$

4.2.3.7 | False negative rate parity

This is also known as **Equality of opportunity** and it is considered as a weaker version of Equalized odds.

This metric is satisfied if all the sensitive attribute groups have equal FNR (the probability of a subject in a positive class to have a negative predictive value). Since a classifier with equal FNR will get also equal TPR, we can use both indistinctly.

In terms of probability, we can formulate it as follow:

$$P_{protected, Y=positive}[C = c] = P_{unprotected, Y=positive}[C = c] \quad \forall c \in \{negative, positive\}$$

In terms of statistical measures (confusion matrix), we can formulate this metric as follow:

$$FNR_{protected} = FNR_{unprotected} \text{ or } TPR_{protected} = TPR_{unprotected}$$

If the prevalence of positive outcome for both subcategories of the sensitive attribute is the same in the entire population and does not depend on the sensitive attribute itself, this definition becomes equivalent to the demographic or proportional parity.

4.2.3.8 | Specificity parity or False positive rate parity

This metric is also known as **Predictive equality**.

In Equalized odds we already introduced the FPR, but in that case it required to satisfy both TPR and FPR equality at the same time, then FPR parity is less restrictive.

This metric is satisfied if all the sensitive attribute groups have equal FPR (the probability of a subject in the negative class to have a positive predictive value). Since a classifier with equal FPR will get also equal TNR, we can use both indistinctly.

In terms of probability, we can formulate it as follow:

$$P_{protected, Y=negative}[C = c] = P_{unprotected, Y=negative}[C = c] \quad \forall c \in \{negative, positive\}$$

In terms of statistical measures (confusion matrix), we can formulate this metric as follow:

$$FPR_{protected} = FPR_{unprotected} \text{ or } TNR_{protected} = TNR_{unprotected}$$

Specificity is also known as TNR (true negatives divided by all negatives), therefore it represents the probability of a subject in the actual negative class to be predicted as negative. This metric could be useful as it is ideally the opposite of Equalized odds.

4.2.3.9 | Negative predictive value parity

Negative predictive value parity is satisfied when the negative predictive values in the subgroups are equal. This metric could be useful as it is ideally the opposite of the Predictive rate

parity.

In terms of probability, this means:

$$P_{protected,C=negative}[Y = y] = P_{unprotected,C=negative}[Y = y] \quad \forall y \in \{negative, positive\}$$

A classifier with equal NPV has also equal FOR, that is why we can use indistinctly all y values to equalize the probabilities. Indeed, in terms of statistical measures (confusion matrix), we can formulate this metric as follow:

$$NPV_{protected} = NPV_{unprotected} \text{ or } FOR_{protected} = FOR_{unprotected}$$

In fairness package:

Negative predictive value parity and Predictive rate parity are only treated individually, but if we would require both to be satisfied, then the metric is called **Conditional use accuracy equality** [10] and it implies equivalent accuracy for sensitive attribute subcategories, from both positive and negative predicted classes.

4.2.3.10 | Matthews correlation coefficient comparison

Matthews correlation coefficient is a scalar metric used in binary classification, such as accuracy, precision, recall or F1-score.

Unfortunately, these mentioned metrics are subject to some flaws: for instance, precision and accuracy are sensitive to class imbalance [35]. Instead, recall and F1-score only depend on the positive groups and do not consider the number of negative samples misclassified as positive (otherwise it could have been an issue in problems containing class imbalanced data with many negative samples), yet they are still not utterly effective when comparing performance between classifiers [28].

Then, according to Davide Chicco and Giuseppe Jurman [15], the most informative metric to evaluate a binary classifier is the Matthews correlation coefficient (MCC):

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

or equivalently, in terms of rates

$$MCC = \sqrt{TPR \cdot TNR \cdot PPV \cdot NPV} - \sqrt{FPR \cdot FNR \cdot FDR \cdot FOR}$$

MCC is a more reliable statistical measure because it produces a high score only if the prediction obtained good results in every confusion matrix category (true positives, false negatives, true negatives, and false positives), in proportion to both the size of the positive and negative classes. It ranges from -1 to 1.

Despite of these positive aspects, some studies claims as well that MCC is not suitable for imbalanced datasets: indeed, according to a recently published research paper, if we are working

on imbalanced dataset (in terms of target class) MCC value deteriorates seriously compared to a balanced dataset [57].

4.2.3.11 | ROC and AUC comparison

The **receiver operating characteristics** (ROC) curve is a popular valuation that plots true positive rate over false positive rate, in order to provide a visualization of the offset between correctly classified positive samples and incorrectly classified negative samples.

If the dataset we are working on is highly skewed (not equally distributed on both sides of the distribution, hence not symmetrical distributed), ROC curves can present overly optimistic results and we should use instead **Precision–Recall** (PR) curves [16].

The **area under the ROC curve** (AUC) can be used as a metric to compare classifiers performance: it is both scale-invariant and threshold invariant, which means respectively that it is independent from the predicted absolute values and from the classification cutoff. An excellent model has AUC near to the 1 which means it has good measure of (separability); on the other hand, an area of 0.5 corresponds to random guessing.

4.2.4 | Fairness criteria

The three main following fairness criteria [6] are based on the concept of statistical independence and they have been already mentioned in the previous fairness metric section 4.2.3. This definition will be helpful in the comprehension: definition *[section]

Definition 1 (Conditional independence) *Given X, Y, Z three discrete random variables. If, when Z is known, the value of Y does not add any additional information about X , namely*

$$\mathbb{P}(X \leq x, Y \leq y | Z = z) = \mathbb{P}(X \leq x | Z = z) \cdot \mathbb{P}(Y \leq y | Z = z) \quad \forall x, y, z.$$

we say that X and Y are conditionally independent given Z and it is written

$$X \perp Y | Z$$

In particular, if X and Y are conditionally independent given Z , then

$$\mathbb{P}(X = x | Y = y, Z = z) = \mathbb{P}(X = x | Z = z)$$

4.2.4.1 | Independence

It requires the sensitive attribute A to be independent from the classifier C , then we could say $R \perp A$, where R is the score variable we use for the prediction of our classifier C .

We have already formally defined this as demographic parity or proportional parity in the previous section.

This principle is quite popular in data fairness, mainly due to the fact that in some realistic situations, such as hiring processes, we want to stress that human performance or qualifications

are totally independent from certain other innate attributes, such as gender or ethnicity. Anyway there is a drawback: it may happen that if we choose randomly people from a subcategory and qualified people from another subcategory, we can still satisfy independence, even though the random subjects might be people with far lower qualifications.

4.2.4.2 | Separation

Since in practical contexts we often have a correlation between the target variable and the sensitive attribute (e.g. one age category may have lower or higher probability to get a loan, due to banks business reasons), then there could be allowed a correlation between the prediction and the sensitive attribute, conditionally to the target variable Y . Namely, $R \perp A | Y$. This principle implies both true positive rate parity and false positive rate parity and in some situations it is considered more helpful than independence because it aims to reduce errors uniformly in all subgroups.

However, the main ethical compromise in achieving this principle is that it may arise gaps between subgroups: for instance, if we have high qualified subjects in one group and low qualified subjects in another, while satisfying separation, we will give better outcomes to the first group. This might lead to amplify social gaps between the two groups, because the first one could result in have more disposable income, which would be transferred to their children, who will reinforce the population of their parents' group as adults. This is a sort of *echo chamber* that may lead to amplify social discrepancies.

4.2.4.3 | Sufficiency

This principle is achieved if the target variable is independent from the sensitive attribute, conditionally to the prediction $Y \perp A | R$. In particular, when we work with a binary classification, we satisfy this condition by requiring a of positive/negative predictive value parity across all subgroups.

Even sufficiency, as separation, may not help reducing the gap between two different groups, for similar reasons.

In a perfect situation, the best goal would be reaching all the three criteria at the same time, but they imposes complex constraints on the distribution D , therefore they are mutual exclusive, as we show below.

But if all the criteria cannot be satisfied at the same time, how can we get fairness? The proper answer will be reaching an optimal trade-off between these three principles.

We analyse pairwise the relationships between the criteria:

4.2.4.4 | Independence \longleftrightarrow Sufficiency

These two criteria cannot both hold, under the assumption that A and Y are not independent, hence $A \not\perp\!\!\!\perp Y$. We have to bear in mind that this is not a restrictive assumption, but the most common situation in real cases: usually the sensitive attribute is related to the target, that means having higher positive outcomes in a subgroup than in another.

By this assumption, it is trivial the implication:

$$A \not\perp\!\!\!\perp Y \implies A \not\perp\!\!\!\perp R \mid Y \text{ or } A \not\perp\!\!\!\perp R.$$

That means: if A depends on the target Y , then either A depends on the classifier conditionally to the target or it does not depend on the classifier at all (target and classifier are obviously always dependant).

proof: we can demonstrate it by reducing to absurd. Let assume that both sufficiency and independence holds, then

$$A \perp\!\!\!\perp Y \mid R \text{ and } A \perp\!\!\!\perp R \implies A \perp\!\!\!\perp (Y, R) \implies A \perp\!\!\!\perp Y$$

which is absurd due to our original hypothesis.

4.2.4.5 | Independence \longleftrightarrow Separation

Also in this case, Independence and Separation cannot both hold, but besides the previous assumption of $A \not\perp\!\!\!\perp Y$ we have to add that Y is binary to prove the mutual exclusion. As before, it is also taken for granted that $R \not\perp\!\!\!\perp Y$, like in realistic classification problems.

proof: let assume instead that both independence and separation holds

$$A \perp\!\!\!\perp R \text{ and } A \perp\!\!\!\perp R \mid Y$$

For the theorem of total probability,

$$\mathbb{P}(R = r \mid A = a) = \sum_y \mathbb{P}(R = r \mid A = a, Y = y) \cdot \mathbb{P}(Y = y \mid A = a)$$

then by the assumption of independence and separation, it becomes

$$\mathbb{P}(R = r) = \sum_y \mathbb{P}(R = r \mid Y = y) \cdot \mathbb{P}(Y = y \mid A = a)$$

As well for the theorem of total probabilities we can say

$$\mathbb{P}(R = r) = \sum_y \mathbb{P}(R = r \mid Y = y) \cdot \mathbb{P}(Y = y)$$

Hence, we can equalize the summations

$$\sum_y \mathbb{P}(R = r \mid Y = y) \cdot \mathbb{P}(Y = y \mid A = a) = \sum_y \mathbb{P}(R = r \mid Y = y) \cdot \mathbb{P}(Y = y) \quad (4.1)$$

but in case of a binary target, this equation is satisfied if $A \perp\!\!\!\perp Y$ or $R \perp\!\!\!\perp Y$ (this is not an exclusive *or*).

Indeed, on the one hand,

$$\mathbb{P}(Y = y | A = a) = \frac{\mathbb{P}(A = a | Y = y) \cdot \mathbb{P}(Y = y)}{\mathbb{P}(A = a)}$$

then the equation

$$\sum_y \mathbb{P}(R = r | Y = y) \cdot \frac{\mathbb{P}(A = a | Y = y) \cdot \mathbb{P}(Y = y)}{\mathbb{P}(A = a)} = \sum_y \mathbb{P}(R = r | Y = y) \cdot \mathbb{P}(Y = y)$$

holds if and only if

$$\frac{\mathbb{P}(A = a | Y = y)}{\mathbb{P}(A = a)} = 1$$

that means

$$\mathbb{P}(A = a | Y = y) = \mathbb{P}(A = a) \cdot \mathbb{P}(Y = y)$$

namely $A \perp\!\!\!\perp Y$.

On the other hand, since we considered Y as binary, we can develop the summation for $y = 0$ and $y = 1$. The left term will be

$$\mathbb{P}(R = r | Y = 1) \cdot \mathbb{P}(Y = 1 | A = a) + \mathbb{P}(R = r | Y = 0) \cdot (1 - \mathbb{P}(Y = 1 | A = a))$$

and the right

$$\mathbb{P}(R = r | Y = 1) \cdot \mathbb{P}(Y = 1) + \mathbb{P}(R = r | Y = 1) \cdot (1 - \mathbb{P}(Y = 1))$$

Then if we equalize these two terms as in 4.1, we get

$$\mathbb{P}(Y = 1) \cdot k = \mathbb{P}(Y = 1 | A = a) \cdot k$$

where $k = \mathbb{P}(R = r | Y = 1) - \mathbb{P}(R = r | Y = 0)$.

This equation can be satisfied if $A \perp\!\!\!\perp Y$ (as we already said), because $\mathbb{P}(Y = 1 | A = a) = \mathbb{P}(Y = 1)$, or if $k = 0$, that is $R \perp\!\!\!\perp Y$.

In both case, this result is absurd, since it denies the hypothesis, then it is not possible to assume both $A \perp\!\!\!\perp R$ and $A \perp\!\!\!\perp R | Y$.

4.2.4.6 | Separation \longleftrightarrow Sufficiency

Both Separation and sufficiency implies non-trivial restrictions for A , R and Y . If all the restrictions hold, we end up in a degenerate solution space.

proof: we already proved that, if either sufficiency or separation holds, then independence cannot hold, therefore $A \not\perp\!\!\!\perp R$.

When both sufficiency and separation hold, the following property of conditional independence should be valid

$$A \perp\!\!\!\perp R \text{ and } A \perp\!\!\!\perp Y \mid R \implies A \perp\!\!\!\perp (R, Y)$$

under the assumption that all events have positive probabilities, as it is showed in [52]. It is then trivial the implication

$$A \perp (R, Y) \implies A \perp R \text{ and } A \perp Y$$

but it cannot be possible due to implausible independence principle.

4.3 | Bias mitigation techniques

We have already analyzed in previous chapters the significance of biases, their types and how can they affect fairness in data science. In this section we will focus on some bias mitigation techniques: in particular, these are usually divided in three main groups, based on where in the machine learning pipeline they are exploited. Then we have **pre-processing** techniques when we apply the bias mitigation algorithm on the training data; **in-processing** techniques if we want to modify the model during its training and get a fairer machine learning algorithm; **post-processing** techniques to mitigate bias on the already predicted labels.

Another way to classify the bias mitigation methods could be based on which fairness criterion or fairness metric they are trying to satisfy, but this might result in an categorization overlay.

In addition, we will not depict all the existing state-of-the-art techniques, so the following list will not be exhaustive, but we will mainly focus on **balancing data** and other methods that are implemented and available in IBM AIF360 R package.

There is a principle in information theory called **data processing inequality** that declares any data process as a possible source of loss of information. In no cases there is the possibility to get new information from a data processing. Moreover, in the best situation, you can only equalize the amount of information throughout the process. Thus, for all the following techniques we cannot expect not to lose *information*, by meaning with this term all the data characteristics that they had before the technique application.

On account of the high evolution rate of new algorithm and resources for big data, and due to the already remarkable quantity of possible techniques combinations, how can we optimally chose a bias mitigation strategy? First of all, we have to focus on the step of ML process where we can take action, since most of the time we can only modify our data in post-processing. However, if we have *carte blanche*, it might be a good strategy to combine different types of processing algorithms together, as for the Fairway method [13], which proposes a combination of pre-processing and in-processing bias mitigation.

Although we may think that the more strategies we put together, the highest fairness we get, this is a deceptive conjecture. Indeed, we have to bear in mind that some algorithms focus on different fairness criteria (e.g. *reweighing* algorithm is based on the independence criterion, whereas *disparate impact remover* on separation) and we already showed how independence, sufficiency and separation are mutually incompatible. When we used multiple algorithms, we

should check their cumulative performance, compared to the individual one, because it could be possible they interfere one another.

As a rule of thumb, we should deal with post-processing algorithms only in case we are not allowed to interfere with the ML process in advance (the earliest intervention, the best results).

In conclusion, there are no unique effective strategies to be applied in terms of fairness: for each dataset we have to make specific and different considerations in order to take into account all the possible case limitations.

4.3.1 | Pre-processing

4.3.1.1 | Balancing

Providing unbalanced data to a classifier can make them biased towards the majority class, basically because the classifier does not have enough data to learn more about the minority. Since it is quite frequent to have an imbalanced dataset, we can many research in literature about this problem. Specifically, some studies [40] [41] have shown that for some classifiers a balanced dataset provides an enhancement in classification performance, rather than the imbalanced one. However, some other studies [9] [33] on the contrary, studies have also proved that classifiers modelled on some imbalanced datasets do not show performance differences (not even relevant performance decrease) to classifiers modelled on the same balanced data. Consequently, previous research suggests always to rebalance data when they are highly skewed, by using one of the several available resampling techniques, such as random sampling methods, cost-sensitive methods, kernel-based methods, and active learning methods [28]. In fact, these methods can be divided into two groups: (1) based on re-sampling or combinations and (2) based on algorithmic strategies, as cost-sensitive and boosting.

In particular, in our experiment we used two basic sampling methods, with no algorithmic approach: random sampling and **synthetic minority oversampling technique** (SMOTE), which include both undersampling and oversampling.

- **Random sampling:** it consists in the random removal or addition of instances of the majority or minority class to achieve the exact number of individual for each group. alternatively, it is possible as well to decide a specific and different proportion of subjects for each group, so that data will be amplified or reduced accordingly.

Basing on add or subtract instances, we call it upsampling/oversampling or downsampling/undersampling respectively.

This technique might have relevant side effects: downsampling can potentially lead to loss of information, while oversampling can result in overfitting, due to the fact that it exactly replicates the minority samples and it introduces redundancy.

- **SMOTE:** it is an re-sampling approach in which the minority class is oversampled by

creating synthetic examples instead of random over-sampling with replacement. The oversampling rests on generating synthetic examples by using k minority class nearest neighbors, in the following way:

1. Take the current feature vector (sample) and evaluate the difference between it and its k nearest neighbors.
2. Multiply these differences by a random number between 0 and 1, and add it to the current feature vector. This step can be interpreted as a selection of a random point along the line segment between two specific features.

Figure 4.1 offers a visualisation of how it works.

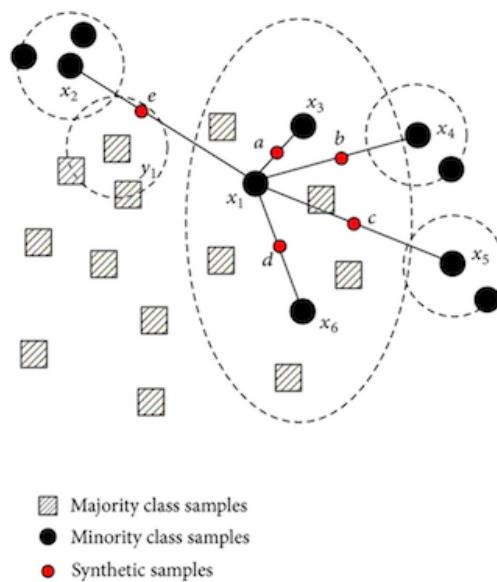


Figure 4.1: SMOTE visualisation - extract from [30]

See algorithm 1 for a SMOTE pseudo-implementation.

Algorithm 1 SMOTE algorithm

Require: n = number of minority class samples
 $perc$ = amount of SMOTE oversampling/undersampling % between 0 and 100
 k = number of nearest neighbours

Ensure: $(perc/100) \cdot n$ synthetic minority class samples

▷ If $perc$ is less than 1, select randomly only a percentage of samples in the minority class

```

1: if  $n < 100$  then
2:    $n = (N/100) \cdot n$ 
3:    $perc = 100$ 
4: else
5:    $numAttributes$  = number of attributes
6:    $Sample[ ][ ]$ : array for original minority class samples
7:    $index$ : index for counting the number of new synthetic samples, initialized to 0
8:    $Synthetic[ ][ ]$ : array for new synthetic samples
  ▷ Compute  $k$  nearest neighbors for each minority class sample
9:   for  $i = 1$  to  $n$  do    ▷ Compute  $k$  nearest neighbors for  $i$ ; save the indices in  $nnArray$ 
10:     $Populate(perc, i, nnArray)$                                 ▷ See procedure below
11:   end for
12: end if
  ▷ Procedure to generate the synthetic samples:
13: procedure  $Populate(perc, i, nnArray)$ 
14:   while  $n \neq 0$  do
  ▷ Choose a random number between 1 and  $k$ , call it  $nn$ , to randomly select one of the  $k$  nearest neighbors of  $i$ 
15:     for  $j = 1$  to  $numAttributes$  do
16:        $diff = Sample[nnArray[nn]][j] - Sample[i][j]$ 
17:        $gap$  = random number between 0 and 1
18:        $Synthetic[index][j] = Sample[i][j] + gap \cdot diff$ 
19:     end for
20:      $index +=$ 
21:      $perc = perc - 1$ 
22:   end while
  return
23: end procedure
```

Case studies

The experiment is based on two datasets and its aim is to research on bias and bias mitigation. For each dataset we proceed in the following parallel way:

1. **Data exploration:** we firstly made data exploration for each dataset, then we used logistic regression to model the prediction on the binary attributes we were interested into (crime recidivism or drug consumption).
2. **Fairness metrics:** after classification, we analysed and compared the fairness metrics available in R `fairness` package. By means of this evaluation, we identified the sensitive attribute that got the mean worst performance in terms of fairness.
3. **Bias mitigation:** we performed on the dataset one or more bias mitigation techniques described in chapter 4 and we evaluate again the fairness metrics, focusing on the identified attributed at step 2. Then we compared them to the ones of the original dataset.
4. **Comparisons:** we drew our comparative conclusions between the different types of mitigation and the different datasets.

5.1 | Datasets

5.1.1 | COMPAS

5.1.1.1 | Context description

The COMPAS case is a scandalous case of software discrimination, which owns its notoriety to ProPublica, an independent no-profit organization that aims to produce investigative journalism in the public interest.

In 2016 ProPublica collected the risk scores assigned to more than 7,000 people arrested in Broward County, Florida, between 2013 and 2014. They conducted an accurate study [34] in order to prove that the software used to evaluate risk assessment by the law enforcement

was biased against black people. Their analysis showed that scores for white court defendants were skewed toward lower-risk categories, while Scores for black defendants were not. Risk assessments are quite common yardsticks in U.S.A. courtrooms: they can have a relevant role in decisions about whether or not a criminal (or alleged criminal) can be set free in the criminal justice system, by giving the assessments results to judges during sentences. It is also frequent in many jurisdictions to adopt a risk assessment software before rigorously testing whether it works.

The risk assessment tool was developed by a for-profit company, Northpointe, which never made available to the public the algorithm source code or the calculations used to get defendants' risk scores, hence it was impossible for the people involved in the case to assess what might have drawn the disparity. It assigns risk scores between 1 and 10, which indicate their likelihood of committing a violent crime based on more than 100 factors, including age, gender, and criminal history. In particular, ProPublica revealed that blacks were almost twice as likely as whites to be labeled as higher risk subjects, without in fact re-offend. On the other hand their research stand also that whites were much more likely than blacks to be labeled lower risk, with actual recidivism.

This tool had then a high impact on several people's lives: some judges preferred to evaluate the cases without relying on it, but some other totally reckon on this automated tool, sentencing people for longer prison detention or releasing other in advance, regardless of fairness.

5.1.1.2 | Data exploration and classification:

We took an already cleaned COMPAS dataset, available in the package `fairness` for R.

The dataset has 6172 entries and the following 9 variables:

- **Two_yr_Recidivism:** yes/no for recidivism or no recidivism. This is the outcome or target in this dataset.
- **Number_of_Priors:** number of priors, normalized to mean 0 and standard deviation 1.
- **Age_Above_FourtyFive:** yes/no for age above 45 years or not.
- **Age_Below_TwentyFive:** yes/no for age below 25 years or not.
- **Female:** female/male for gender.
- **Misdemeanor:** yes/no for having recorded misdemeanor(s) or not.
- **ethnicity:** Caucasian, African American, Asian, Hispanic, Native American or Other.
- **probability:** predicted probabilities for recidivism, results from a classification made by the authors of the `fairness` package, ranges from 0 to 1.
- **predicted** binary predicted values for recidivism, 0/1 for no/yes.

Since the aim of this research is not to study Northpointe software fairness, nor any other already-existing algorithms fairness, we firstly deleted the columns 'predicted' and 'probability', because we want to use our own classification results.

Then, we merge the two age variables into one single column made of three options: under25/over45/between25_45.

We also substituted 'yes/no' in 'Two_yr_Recidivism' with 'Yes_recidivism' and 'No_recidivism' to be more clear: indeed, 'Yes_recidivism' will be our *unfavorable* outcome, and 'No_recidivism' the *favorable* one.

Now the dataset appears as in table 5.1.

Table 5.1: COMPAS dataset first rows

| | Two_yr_Recidivism | Number_of_Priors | Female | Misdemeanor | ethnicity | Age |
|----|-------------------|------------------|--------|-------------|------------------|--------------|
| 4 | No_recidivism | -0.68 | Male | yes | Other | between25_45 |
| 5 | Yes_recidivism | 2.27 | Male | no | Caucasian | between25_45 |
| 7 | No_recidivism | -0.68 | Female | yes | Caucasian | between25_45 |
| 11 | No_recidivism | -0.68 | Male | no | African_American | between25_45 |
| 14 | No_recidivism | -0.68 | Male | yes | Hispanic | between25_45 |
| 24 | No_recidivism | -0.68 | Male | yes | Other | between25_45 |

There are three sensitive attributes to work on: Female, ethnicity and Age.

In addition, we can visualise the subgroups distributions with the following pie charts (figure 5.1) we see how these attributes are balanced or imbalanced in the dataset. As we can notice, the dataset is imbalanced: there are evidently more male than female, more subjects between 25 and 45 years old and some ethnicity categories are underrepresented, such as Asian, compared to African American or Caucasian.

However, our target variable seems quite balanced: we have nearly the same amount of recidivists and non-recidivists in our data.

We would like to find the most common combination of the three attributes, both in terms of recidivism and non-recidivism. Hence, we prepared an ordered table in figure 5.2 for combinations on recidivists and another one 5.3 for combination on non-recidivists.

In these two tables we can read an absolute count of the rows with that specific sensitive attribute combination, but we still do not know which is the relative weight that the combination has on the entire dataset, or among all the recidivists/non-recidivists, yet the percentage of incidence of yes (respectively no) recidivism on that specific combination.

Therefore, we plotted, for each combination of these attributes, the percentages of recidivists and non-recidivists on the whole dataset (figure 5.2a and 5.2b), then on the total number of

Table 5.2: COMPAS dataset Yes_recidivism frequent combinations (first rows)

| | Age | ethnicity | Female | count |
|----|--------------|------------------|--------|-------|
| 24 | between25_45 | African_American | Male | 855 |
| 14 | below25 | African_American | Male | 427 |
| 27 | between25_45 | Caucasian | Male | 371 |
| 2 | above45 | African_American | Male | 176 |
| 17 | below25 | Caucasian | Male | 146 |
| 6 | above45 | Caucasian | Male | 135 |
| 23 | between25_45 | African_American | Female | 123 |
| 26 | between25_45 | Caucasian | Female | 112 |
| 29 | between25_45 | Hispanic | Male | 91 |
| 13 | below25 | African_American | Female | 64 |

Table 5.3: COMPAS dataset No_recidivism frequent combinations (first rows)

| | Age | ethnicity | Female | count |
|----|--------------|------------------|--------|-------|
| 20 | between25_45 | African_American | Male | 708 |
| 24 | between25_45 | Caucasian | Male | 490 |
| 5 | above45 | Caucasian | Male | 351 |
| 11 | below25 | African_American | Male | 237 |
| 2 | above45 | African_American | Male | 223 |
| 19 | between25_45 | African_American | Female | 212 |
| 23 | between25_45 | Caucasian | Female | 155 |
| 26 | between25_45 | Hispanic | Male | 148 |
| 14 | below25 | Caucasian | Male | 128 |
| 4 | above45 | Caucasian | Female | 107 |

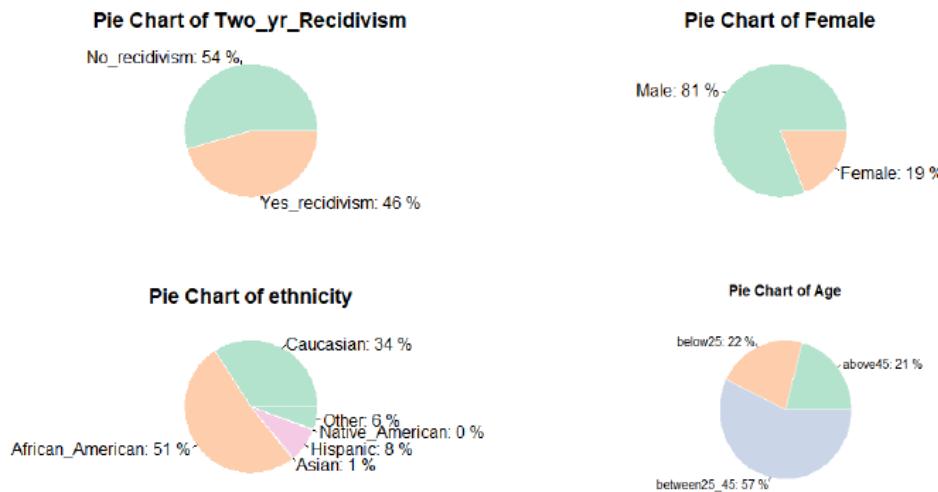


Figure 5.1: COMPAS categorical variables distributions

recidivists or non-recidivists of the dataset (figure 5.2c and 5.2d) and on the total amount of subjects for each combination(figure 5.2e and 5.2f). In the graphs we used always red color for the recidivists and green color for non-recidivists. We can notice that the first couple and the second couple of graphs show almost the same trend: this is due to the fact that the first couple of graphs plots the percentage on the whole data, while the second couple plots the percentage on all the recidivists 5.2c and on all the non-recidivists 5.2d, but we have already noticed that there is nearly the same number of recidivists and non-recidivists in the dataset. Moreover, as confirmation of the unbalancing, in the first two couple of graphs we see higher percentages for males, in particular Caucasian and African-American.

The most interesting visualisation is given by the last couple of graphs: here we see, for instance, that the Caucasian males under 25 are more commonly recidivists than non-recidivists, but it is also evident that the African-American males under 25 have far more tendency to recidivism than no-recidivism. Indeed, it is quite remarkable the overall higher percentage of recidivism for the below 25 age category, while, concerning to males and females in-within each graph, we do not have any considerable difference.

Anyway, the differences between recidivism and no-recidivism on the ethnicity lines are what drew our attention the most: if we compare the two graphs, all in all we have truly different balloon sizes for male African-Americans, for male and female Native-Americans or for all Asians. On the contrary, all the Caucasians seem to have approximately the same percentages for each of them combinations (both for recidivism and no-recidivism).

Accordingly to these reports, we may expect some degrees of unfairness for our classifier, whatever model we will choose. Indeed, if we train our model on skewed sensitive data, it is highly probable to get unfairness.



Figure 5.2: IMAGE RECIDIVISM ON DIFFERENT COMBINATIONS

Since we have only categorical data, except for the number of priors, we will focus our interest on the possible relationships among the sensitive categorical attributes. We performed the Pearson's Chi-square test and we plotted the residuals in a corrplot, for different attribute combinations. We reported here in figure 5.3 the resulting Pearson's Chi-square test graphs, to understand how the three sensitive attributes are related to the target.

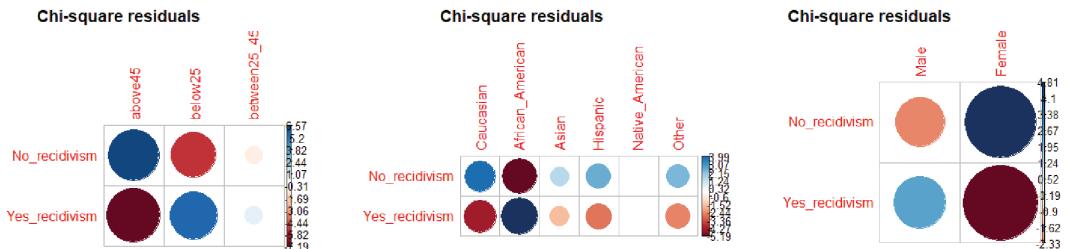


Figure 5.3: COMPAS Pearson's Chi-square test

Positive residuals are in blue: these values in cells specify an attraction (positive association) between the corresponding row and column variables. Negative residuals are in red: this implies a repulsion (negative association) between the corresponding row and column variables.

As we already noticed previously, there is a strong association between Age below 25 and recidivism and between African-American and recidivism. Now, we can also observe that there is a relevant positive correlation also for females and no-recidivism as for other age or ethnic categories.

In our test we got $p-value < 2.2 \cdot 10^{-16}$ for both Age category and ethnicity; $p-value = 2.898 \cdot 10^{-15}$ for gender attribute, namely all significantly low values, so that we could reject the null hypothesis of independence among variables.

5.1.1.3 | Classification

As we already mentioned in chapter 4, we used logistic regression to make our own predictions. We randomly sampled our data: 70% as training test and 30% as test set. We used R `seed()` function to make the experiment repeatable, so that we have the same results at each run time.

We chose to use an optimal cutoff for the predicted probabilities: locating the cut-off point requires a compromise between sensitivity (TNR) and specificity (TPR) and we found it by equalizing their two curves, as in figure 5.4. In fact, there are several approaches to this problem, but this is one of the most frequently used criterion for determination of the test cutoff value: this point of the curve is where the product of these two indices (*Sensitivity · Specificity*) is maximum (see [25]).

Once we found the optimal cutoff we defined the predictions and the resulting confusion matrix was the one in figure 5.5.

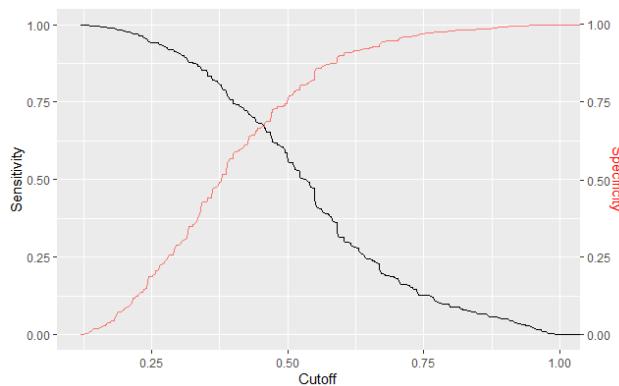


Figure 5.4: COMPAS sensitivity and specificity curves

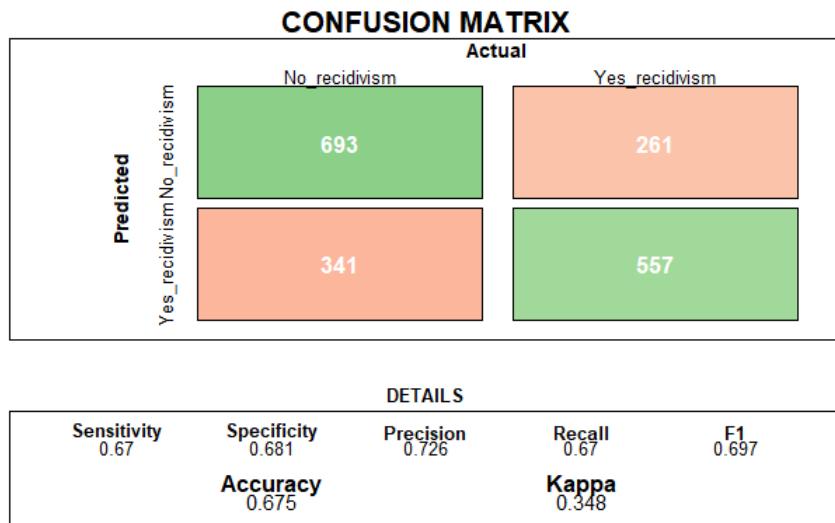


Figure 5.5: COMPAS linear regression confusion matrix

Choen's Kappa is quite good, we can consider it as 'fair' regarding Choen's suggestions. The accuracy is the ratio of correct predictions to total predictions made and it is close to 70%, then it is acceptable. Among the other measures, we see F1 score, which is more explicating than the others, but also not so useful, since our target was quite balanced from the beginning. An F1 score reaches its best value at 1 and worst value at 0, then in this case is similar to accuracy and we can take it as acceptable.

Lastly, we plotted the density of the predicted probabilities for each sensitive attribute (figure 5.6).

The vertical line represents the optimized cutoff: we can clearly notice that some subcategories are more likely classified as recidivists, such as age below 25, while some others are predominantly classified as non-recidivists, like Asians.

In the following section we will try to understand which is the most mistreated sensitive attribute, hence the one for which we have highest unsatisfied fairness metrics parity.

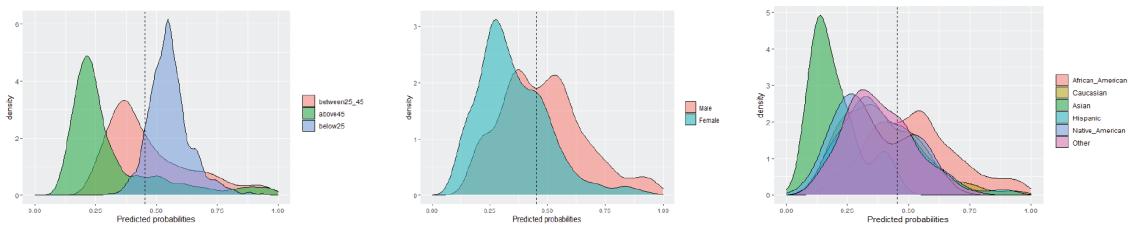


Figure 5.6: COMPAS predicted probabilities' densities

5.1.1.4 | Fairness metrics

For fairness analysis we took advantage of the `fairness` package in CRAN for R language. This library gives the possibility to have absolute and relative values for each of the fairness metrics described in chapter ???. The relatives values are compared to a base subcategory, that can be chosen indistinctly, since it will be given the value to this base subcategory and a comparative value to all the others. The objective of the parity metrics will be having approximately the same values for each subcategory (figures 5.7, 5.8, 5.9), therefore when we plot the relative values for each subcategory, for each metric, we desire to have all of them as close as possible to value 1. However, this goal is evidently not reached and there are subcategories for which barplots are even missing in some metrics, probably due to under-representation in the dataset. In fact, later we will see that when we balance these sensitive attributes, we will not get this kind of issue.

All the fairness metrics numerical values are available in appendix B.

Come this far, how can we detect the sensitive attribute in which we have the most mistreated subcategories in terms of fairness? We compared the delta-percentages for each fairness metric pairwise with respect to the sensitive attributes, then the worst results are the ones we obtained by comparing ethnicity to the other two sensitive attributes, as in figures 5.10a and 5.10b. In order to make this comparisons we attributed a value for each fairness metric plot in figures 5.7-5.8-5.9 by calculating the mean signed deviation for each subcategory to the value 1. Then we compared these values of one attribute to the ones of the others.

In table 5.4 there are the values of ethnicity compared to Female and in table 5.5 there are the ones of ethnicity compared to Age. We reported here only these two, as the most significant ones.

In both cases there are few ethnicity fairness values that are better than the ones for the other sensitive attributes. We can observe in particular that the NPV parity is extreamly lower for ethnicity than the other attributes (even nearly 1500% less). How is it possible? If we look at the single fairness metrics plots for ethnicity, we can see that Native Americans have an almost doubled value than Caucasian and Asians are considered as 0 value, even though, to be more precise, there were not enough Asians to get this metric value for this subcategory: indeed, the actual calculated value is 'NaN' (see numerical values in appendix B, that means



Figure 5.7: COMPAS fairness metrics Age

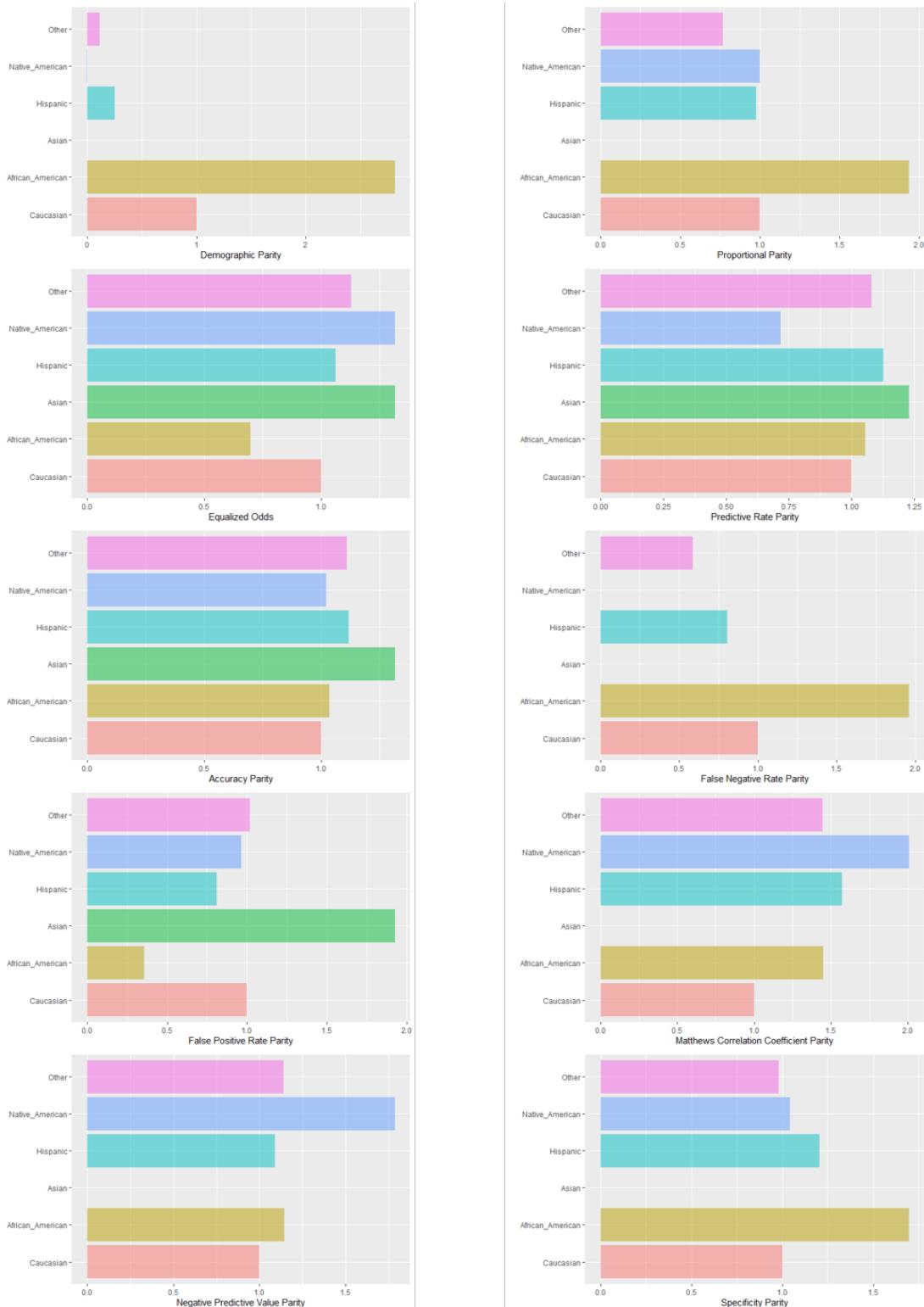


Figure 5.8: COMPAS fairness metrics ethnicity



Figure 5.9: COMPAS fairness metrics Female

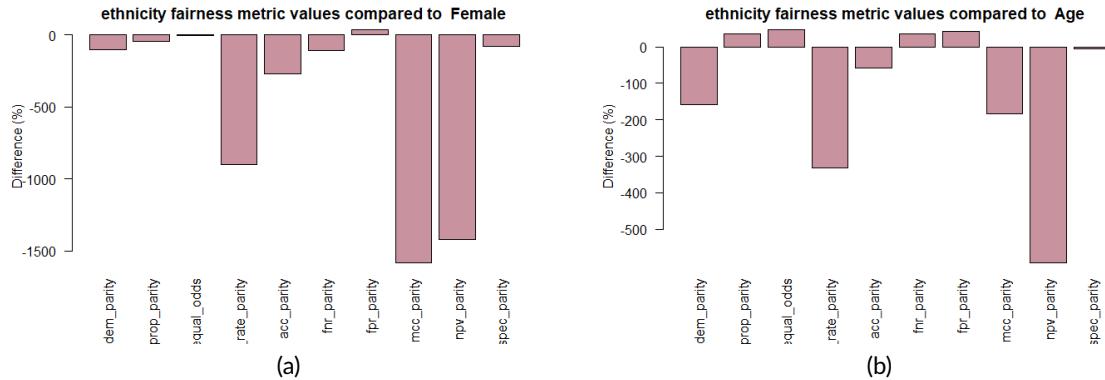


Figure 5.10: Fairness metric values of ethnicity compared to: (a) Female and (b) Age.

Table 5.4: COMPAS ethnicity VS Female fairness metrics mean signed deviations (% values)

| dem_parity | prop_parity | equal_odds | pred_rate_parity | acc_parity |
|------------|-------------|------------|------------------|-------------|
| -105.38 | -44.48 | -7.79 | -902.96 | -271.60 |
| fnr_parity | fpr_parity | mcc_parity | npv_parity | spec_parity |
| -107.37 | 35.93 | -1580.76 | -1418.87 | -78.52 |

Table 5.5: COMPAS ethnicity VS Age fairness metrics mean signed deviations (% values)

| dem_parity | prop_parity | equal_odds | pred_rate_parity | acc_parity |
|------------|-------------|------------|------------------|-------------|
| -158.48 | 35.96 | 47.52 | -332.90 | -58.14 |
| fnr_parity | fpr_parity | mcc_parity | npv_parity | spec_parity |
| 36.50 | 41.76 | -184.33 | -591.03 | -5.64 |

there were no Asians subjects predicted recidivists (negative class) that were truly recidivists. Overall, this comparison for NPV parities means that there is much more disparity in predicted recidivists that are actual recidivists among ethnic groups than among age/gender categories. Namely if we take randomly an Asian and a Native-American, is much more likely to have an Asian both predicted and actual recidivist and, at the same time, a Native-American predicted recidivist but actually non-recidivist, than a randomly selected male-female couple with the respective same predicted characteristics of the two ethnic subcategories.

Another sensibly low metric difference, especially regarding the comparison to Female, is the mcc-parity: we know that for imbalanced dataset this metric might be not reliable, but our dataset is quite balanced for the recidivism target class, hence we cannot consider MCC as not indicative from the outset. In addition, MCC takes into account NPV for its calculation, then we could expect it reflects the same trend of NPV parity. This low percentage difference

in MCC parity explains that the classifier (in terms of all TP, FP, TN and FN) could work in much more different ways for the ethnicity subgroups than for the gender subgroups.

The last visibly low metric comparison is the predictive rate parity, that we analysed better under the name of sufficiency: compared to Female in particular, it reaches the 903% less. No improvements for equalized-odds (separation) or for demographic/proportional parity (independence) goes along with this comparative sufficiency decrease. Sufficiency constrains having equal PPV (or equivalently FDR). This relates to the idea of *calibration*, the notion that, between two subgroups, the rate of actual recidivists subjects in the predicted-recidivist-portion (all the predicted recidivists in the dataset) are equal across any two sensitive groups. However, in both graphs, we see a slightly better FPR parity for ethnicity than for the other two attributes: it might reflects as well in a mildly better performance for equalized-odds in ethnicity-Age graph, since it requires both FPR parity and TPR parity. Whereas despite of this fact, in ethnicity-Female graph we have anyway a comparative reduction of equalized-odds metric, probably due to a worse satisfaction of TPR parity for ethnicity than for Female.

In conclusion, for COMPAS dataset we chose to focus on the ethnicity sensitive attributes for further fairness analysis.

5.1.1.5 | Bias mitigation

The first bias mitigation technique that we apply for every dataset is resampling, for rebalancing. We used Shannon index as a metric for detecting changes among our data, bearing in mind that it increases as both the richness and the evenness of the community increase. As we are not modifying the number of species (subcategories), all the Shannon index changes are based on subgroups evenness. Of course, when we balance the data towards an attribute, we will see Shannon index changes within that attribute, but we performed this analysis specifically to observe if these shifts may influence on the others as well.

For COMPAS we created 7 fully balanced datasets and we report for each of them the Shannon index comparison to the original data (see the corresponding references):

- balanced_data0: balanced for Two_yr_recidivism. No relevant improvements in diversity [5.11a](#), as we expected, since the target was already almost fully balanced.
- balanced_data1: balanced for Female. Of course higher Shannon index value for Female, namely less diversity in this category and higher evenness [5.11b](#).
- balanced_data2: balanced for ethnicity. Great increase of Shannon index for ethnicity, concurrent with an exceptional raising for Number_of_Priors [5.11c](#), that reflects in a change of distribution, with slightly lower standard deviation (0.8469448). In this case, we would accept the compromise to lose Number_of_Priors diversity in favor of ethnicity balancing.

- balanced_data3: balanced for Age. Soft increment for Age Shannon index, that goes along with an even higher increment for Number_of_Priors [5.11d](#).
- balanced_data4: balanced for Two_yr_recidivism within each fully balanced subcategory of ethnicity [*Two_yr_recidivism|ethnicity*]. We tried this combination to see if we could get better improvements compared to balanced_data2. In fact, the Shannon index results are pretty much the same, except for a very light improvement for Two_yr_Recidivism, but, for instance, no other evident changes for Number_of_Priors or others [5.11e](#).
- balanced_data5: balanced for Two_yr_recidivism within each unbalanced subcategory of ethnicity. In this case we wanted to assure that the results were pretty much the same as balanced_data0 and indeed it is what we got in terms of Shannon index [5.11f](#).
- balanced_data6: balanced for ethnicity within each balanced subcategory of Female [*ethnicity|Female*]. Since apart from ethnicity, the second most unbalanced sensitive attribute was Female (as we can notice from balanced_data1), we balanced both female and ethnicity [5.11g](#). We could have chosen to invert the balancing orders: firstly balancing for ethnicity and then for Female within each already balanced subcategory of ethnicity and we did this in balanced_data7. We decided not to modify the original data with more than two balancing at the same time, because it could have strayed too far from the characteristics of the initial population.
- balanced_data7: balanced for Female within each balanced subcategory of ethnicity [*Female|ethnicity*]. In this case, there is practically no changing for Shannon index [5.11h](#).

How can we select the balanced dataset for which we get the best fairness performance? For each of them, we performed again classification with logistic regression and then we compared the fairness metrics between the original and balanced datasets. This is not a standard work: actually, we can never say in a deterministic way if a dataset is fairer than another. As we already stressed in the previous chapters, we have to accept compromises, because there is no way to get each kind of fairness equally satisfied at the same time, not even for a single sensitive attribute, not mention all of them. Hence, we decided to focus on ethnicity and we should consider which fairness metric is more relevant for us, depending on the study case we are dealing with. For recidivism classification is it better to lend more weight to independence, sufficiency or separation? This is the question we have to make when we face a data fairness analysis. However, since we have no legal expertise in this sector and this is not the aim of this research, we have addressed our approach to not let prevaricate one fairness criterion on the others. We simply analysed the different fairness metrics and tried to select the balanced dataset which gave us the overall better outcomes.

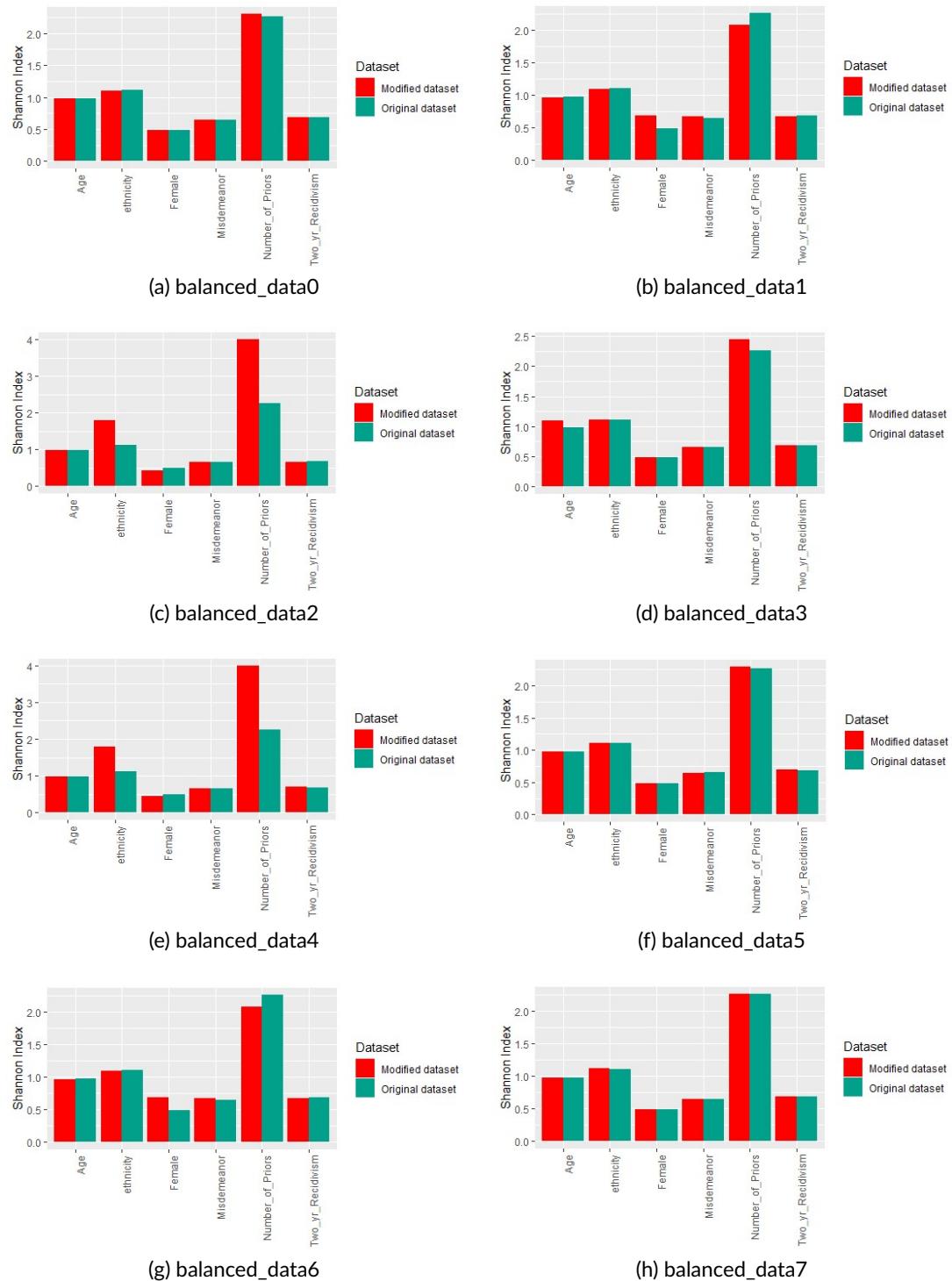


Figure 5.11: Shannon index comparison between original dataset and balances datasets.

5.1.1.6 | Comparisons

■ Observations on balanced datasets VS original data:

In figures [5.12a](#)÷[5.12h](#) are reported the most relevant comparisons between different classified datasets, only regarding the ethnicity variable, which is the one we chose as most mistreated.

As a matter of fact, we get much better results for almost every fairness metric when there is ethnicity fully balanced (figures [5.12c](#) and [5.12e](#)). If we look further, we notice that for [5.12e](#) we lose accuracy parity in favor of higher proportional parity. In this case, both Two_yr_Recidivism and ethnicity are balanced. Keep in mind that these two are both comparisons to the original classified dataset: we get much better values for demographic parity in [5.12e](#) than in [5.12c](#), but this is not as indicative as proportional parity. Indeed, it is much better to get higher proportional parity values than demographic parity values in terms of fairness, as we explained in [4](#).

However, it is noticeable the fact that when Two_yr_Recidivism is fully balanced, the new dataset gives an overall worse fairness performance [5.12a](#) than when Female is fully balanced [5.12b](#). That is another reason why a balancing combination of Female and ethnicity [5.12g](#) or [5.12h](#) might be more interesting than a combination of ethnicity and recidivism [5.12e](#).

The plausible assumption of having similar fairness performance when balancing an attribute such as Two_yr_Recidivism on its own [5.12a](#) and within the subcategories of ethnicity [5.12f](#) is evidently deceptive: in spite of having worse results for some metrics, we got widely different metric values in two cases. Another proof of evidence for this is the comparison between the last two plots [5.12g](#) and [5.12h](#), both balanced for ethnicity and Female, but in different ways.

■ Observations on selected balanced datasets:

After our previous observations, we retrieve indeed that when we focus on ethnicity fairness, the best fairness metrics performances are obtained of course on ethnicity balanced datasets. Hence, we compared the four ethnicity balanced datasets and we report in figure [5.13a](#)÷[5.13b](#) the most relevant comparative graphs.

The immediate deduction is that balanced_data2 does not satisfy Independence for ethnicity groups as much as balanced_data4 does and balanced_data7 does not satisfy Separation as much as balanced_data6 does.

Lastly, in figure [5.14](#) we have a useful overview of all balanced datasets metric values, split in different graphs for each metric. In this way, it could be immediate the choice of one dataset over another, depending on which metric we want to satisfy the most. For instance, if we need to have false negative rates as similar as possible for each subcategory, it is better to use balanced_data4.

In these graphs we have plotted the mean deviations from perfect fairness for each fairness

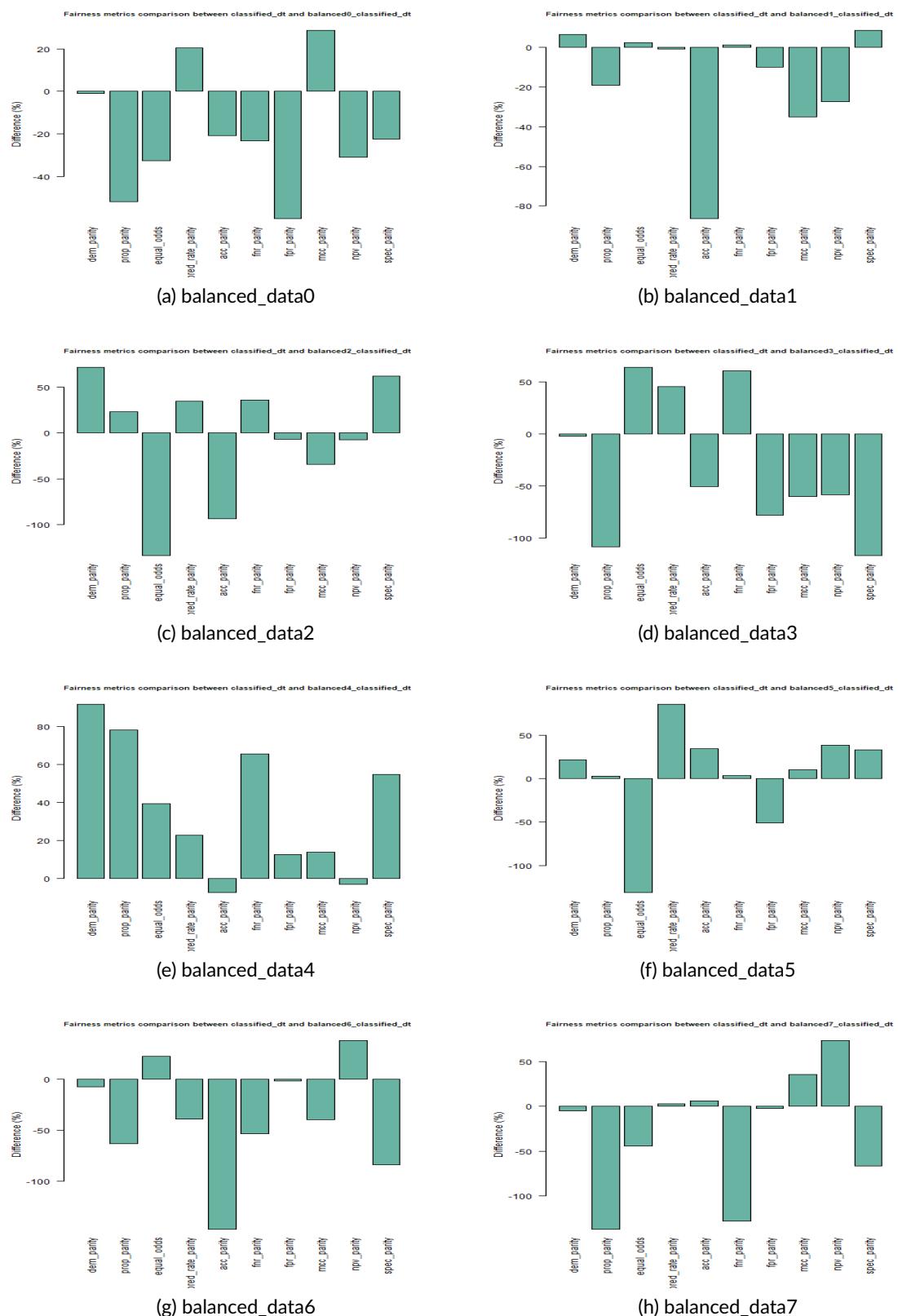


Figure 5.12: COMPAS fairness metric comparison between balanced datasets and original data.

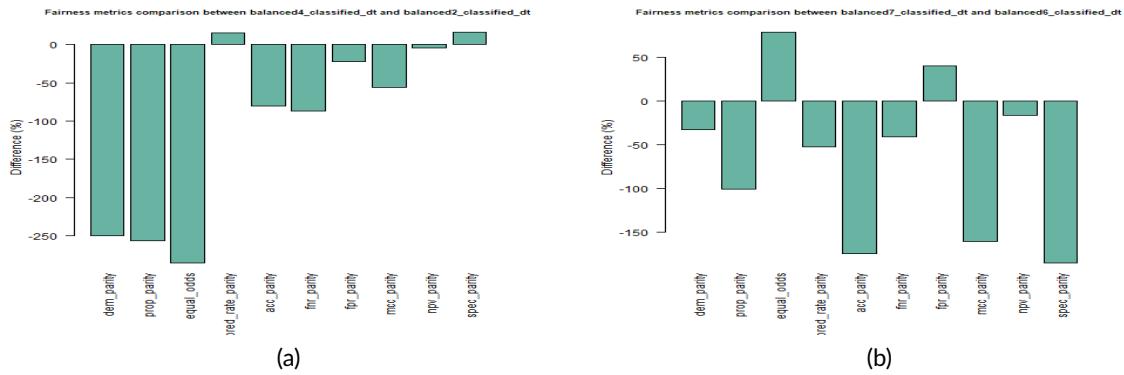


Figure 5.13: COMPAS fairness metric comparison between: (a) balanced_data2-balanced_data4 and (b) balanced_data6-balanced_data7.

metric (all subgroups with value 1), for balanced classified datasets and original classified dataset (Classified_dt). Notice that we do not have comparative values here and the optimal value is 0 (no deviation, namely all subgroups with the same fairness metric value).



Figure 5.14: Mean deviations from complete fairness, for each fairness metric, for balanced classified datasets and original classified dataset

5.1.2 | Drugs

For Drugs dataset we will not explain all the analysis steps as it was done for COMPAS data. Instead, we will report the significant results that will be used for the last comparative analysis among the different datasets. For any further detail, the R code for all datasets analysis is available in appendix A.

5.1.2.1 | Context description

The database was collected by Elaine Fehrman between March 2011 and March 2012 for research proposal, by means of an online survey tool from Survey Gizmo was employed to gather data.

The study recruited 2051 participants over a 12-month recruitment period and the data were collected maximising anonymity. Every participant has been categorized in a different level of 'drug user' based on the recency of use, for each drug. The drugs evaluation regards In the original research [22], two isolated categories ('Never used' and 'Used over a decade ago') are set into the class of non-users and all other categories form the class of users; whereas, we decided to categorize only 'Never used' individuals as non-users. The aim of the original study was to evaluate the individual drug consumption risk separately, for each drug (18 different) and group of drugs. However, our main interest in this case lies in the fact that usually, for clinical cohorts, data are biased when compared with the general population and in fact [22] research showed that this dataset truly is biased when compared to the data published by Egan, et al [19] and Costa Jr & McCrae [43]. The 5 scores, the impulsiveness and the sensation seeing are the results of the so-called NEO Five-Factor Inventory (NEO-FFI-R) questionnaire, which is a highly reliable measure of basic personality traits.

5.1.2.2 | Data exploration

This is a cleaned data frame with 1885 rows and 12 variables, available on UCI repository. Among the variables there are sensitive attributes and some continuous scores: we will not analyse the scores because they brings information we are not interested to the aim of our research. However, we will not perform any feature selection for classification and we will involve them in the logistic regression model. Here is the list of variables:

- **Age:** is age of participant.
- **Gender:** is gender of participant.
- **Education:** is level of education of participant.
- **Country:** is country of current residence of participant.
- **Ethnicity:** is ethnicity of participant.

- **Nscore:** is NEO-FFI-R Neuroticism.
- **Escore:** is NEO-FFI-R Extraversion.
- **Oscore:** is NEO-FFI-R Openness to experience.
- **Ascore:** is NEO-FFI-R Agreeableness.
- **Cscore:** is NEO-FFI-R Conscientiousness.
- **Impulsive:** is impulsiveness measured by BIS-11.
- **SS:** is sensation seeing measured by ImpSS.

Other columns list several types of drugs. We'll select only two of them for which predicting the consumption: **Coke** and **Crack**.

At first, we only had real values, then we mutated them to categorical, to get more data readability. After that, the data are presented as in table 5.6.

Data are highly skewed (except for Gender, which is fully balanced), as we can observe in figure 5.15.

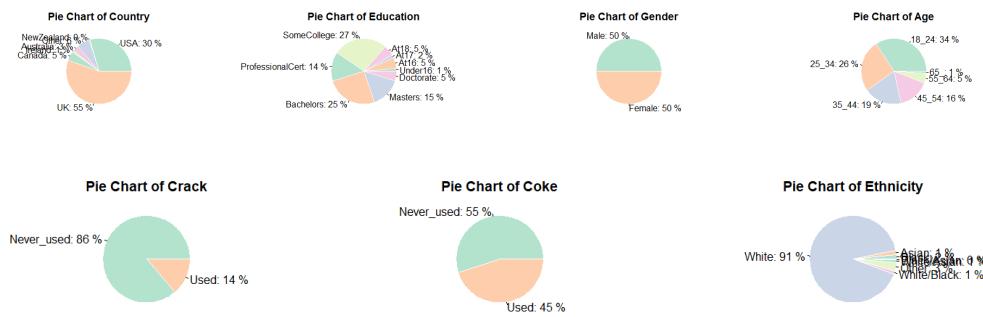


Figure 5.15: Drug dataset categorical variable distributions

Likewise COMPAS case, there are three sensitive attributes in our data (Gender, Ethnicity and Age) and a bunch of other variables that might be even proxies for the sensitive ones.

Since we are interested into two different drug consumption predictions, in the following analysis we will proceed in parallel for Coke and Crack.

In figure 5.16 is shown the Coke percentage consumption distribution on the whole dataset and on each single combination of sensitive attributes. Whereas, in figure 5.17 there are the same plots for Crack consumption. These data regards only subjects who used the corresponding drug. Moreover, we did not perform any cross evaluation for the two drugs consumption: we limited the research on an individual separated consumption analysis.

In figure 5.18 and 5.19 are reported the Chi-square residuals of Pearson's Chi-square test (respectively for Coke and Crack variables), in order to see any possible relationship or causal

Table 5.6: Drug dataset first rows

| Age | Gender | Education | Country | Ethnicity | Nscore | Escore |
|--------|--------|------------------|-----------|-----------|------------|------------|
| 25_34 | Male | Doctorate | UK | White | -0.68 | 1.94 |
| 35_44 | Male | ProfessionalCert | UK | White | -0.47 | 0.81 |
| 18_24 | Female | Masters | UK | White | -0.15 | -0.81 |
| 35_44 | Female | Doctorate | UK | White | 0.74 | -1.63 |
| 65_ | Female | At18 | Canada | White | -0.68 | -0.30 |
| 45_54 | Male | Masters | USA | White | -0.47 | -1.09 |
| Oscore | Ascore | Cscore | Impulsive | SS | Coke | Crack |
| 1.44 | 0.76 | -0.14 | -0.71 | -0.22 | Used | Never_used |
| -0.85 | -1.62 | -1.01 | -1.38 | 0.40 | Never_used | Never_used |
| -0.02 | 0.59 | 0.58 | -1.38 | -1.18 | Used | Never_used |
| -0.45 | -0.30 | 1.31 | -0.22 | -0.22 | Never_used | Never_used |
| -1.56 | 2.04 | 1.63 | -1.38 | -1.55 | Never_used | Never_used |
| -0.45 | -0.30 | 0.94 | -0.22 | 0.08 | Never_used | Never_used |

inference between consumption and sensitive attributes. Regarding both Coke and Crack, Chi-squared approximation may be incorrect, since one Chi-squared test assumption is broken: there are indeed less than 5 subjects over 65 years old who have used the drug. We have also a similar borderline situation for Ethnicity variable. However, we can rely on the Gender Chi-square test: there is a strong association between male and both drugs consumption.

5.1.2.3 | Data classification

We modelled again a logistic regression, which involved all features as predictors. The positive class is "Never_used" and the negative is "Used". The confusion matrix is quite different for Coke (figure 5.20) and Crack (figure 5.21): we can notice a really high TPR for Crack (actual non-consumers predicted as non-consumers) that is not so surprising, since 86% of subjects are crack non-consumers. In figure 5.22 are represented the density plots for Coke consumption probability, with the relative optimal cutoff. It is relevant that in Crack case the optimized cutoff is surprisingly low (0.1829116), probably due to high unbalanced data for Crack consumption variable (only 14% of subjects used this drug), which also leads to issues for fairness

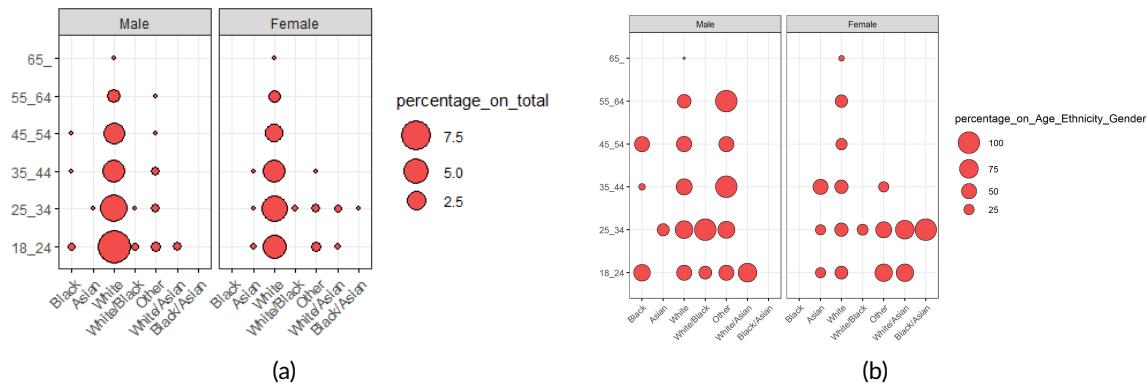


Figure 5.16: % of coke consumption on: (a) total and (b) each sensitive attribute combination.

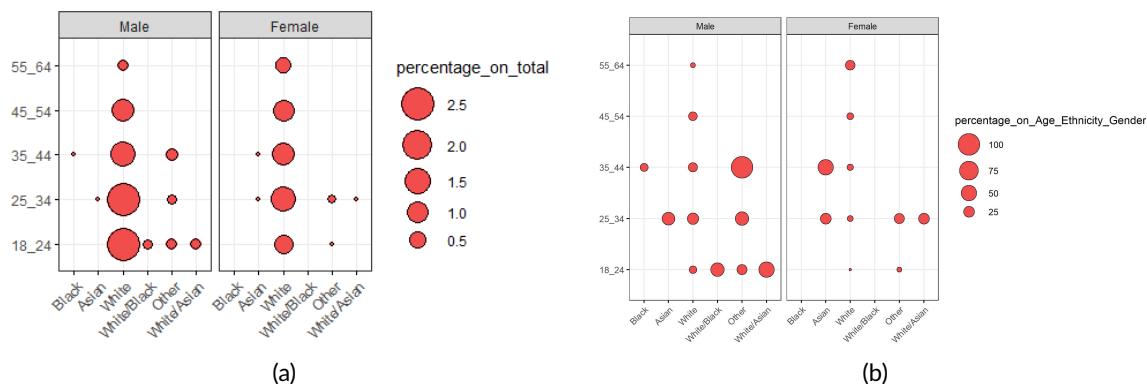


Figure 5.17: % of crack consumption on: (a) total and (b) each sensitive attribute combination.

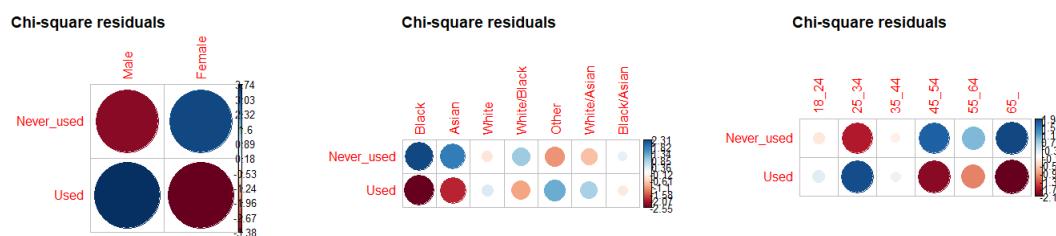


Figure 5.18: Drug dataset Pearson's Chi-square test for Coke

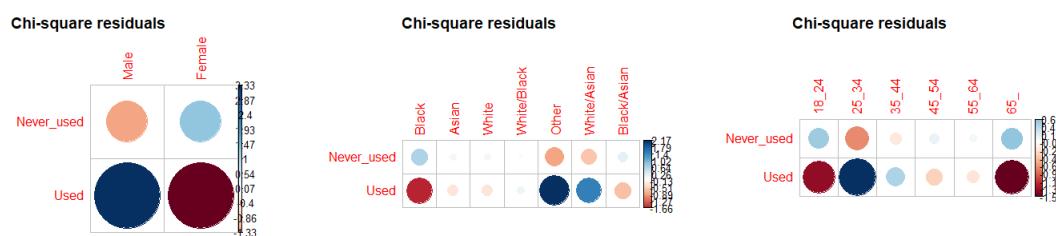


Figure 5.19: Drug dataset Pearson's Chi-square test for Crack

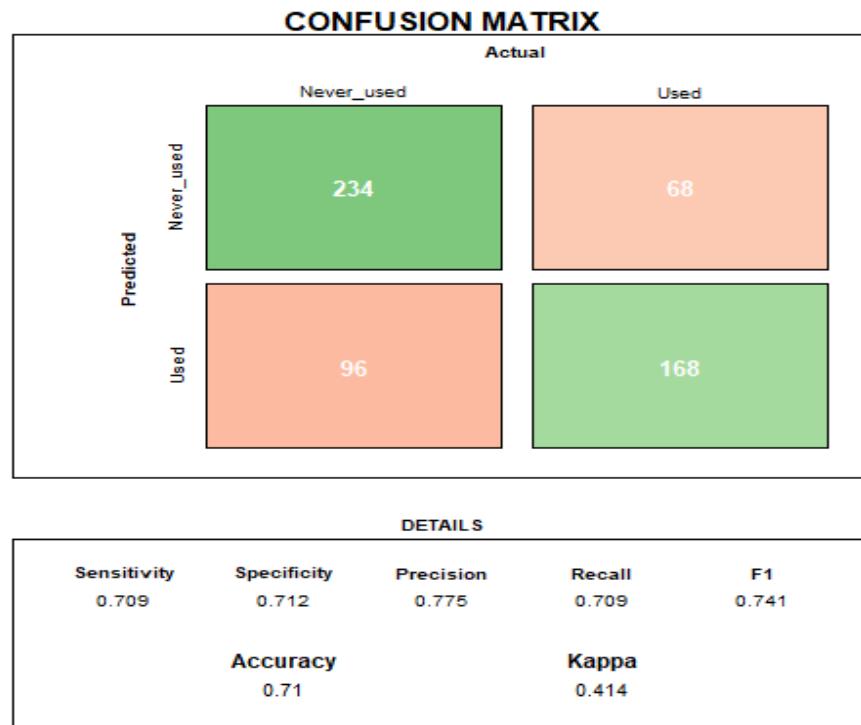


Figure 5.20: Drug dataset Coke classification confusion matrix

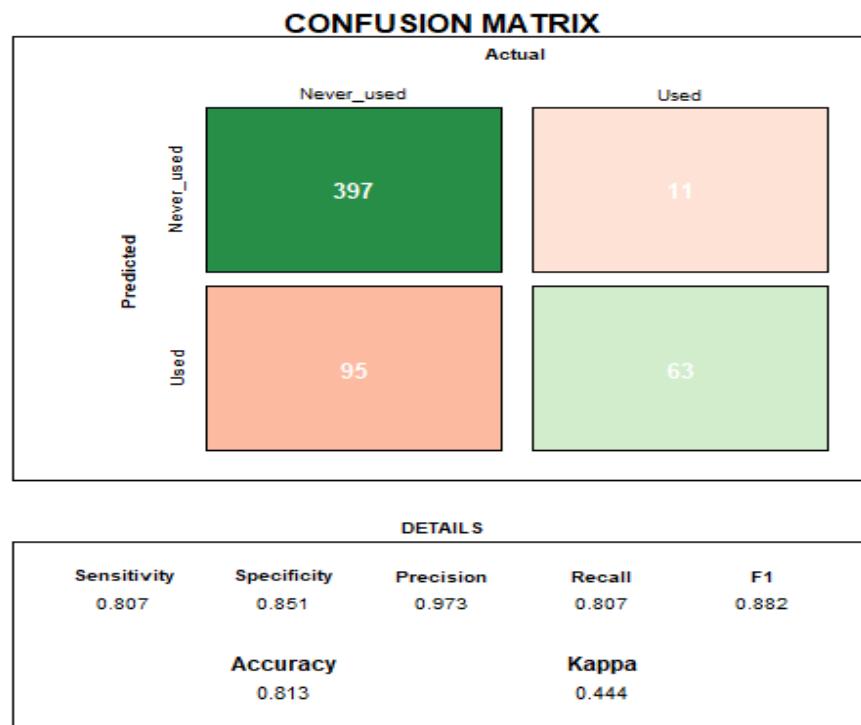


Figure 5.21: Drug dataset Crack classification confusion matrix

metrics and density probabilities plots (hence, we will not report them here for Crack case).

5.1.2.4 | Fairness metrics

We tried several sensitive attributes fairness metrics comparisons and, as well as for COMPAS, it results that for both Coke (figures 5.23a and 5.23b) and Crack (figures 5.24a and 5.24b) the most critic sensitive attribute is Ethnicity. For all the fairness metrics plots and values, please refer to the attached code in appendix B.

5.1.2.5 | Bias mitigation

Taking into account that Gender is already balanced in the original dataset, we created 3 balanced datasets for Coke as follow:

- balanced_data0: balanced for Ethnicity.
- balanced_data1: balanced for Country. In this way, we wanted to see if there could be any improvement in Ethnicity fairness metrics, since Country may be a proxy for Ethnicity.
- balanced_data2: balanced for Ethnicity and Age. For each already balanced ethnic subcategory, we balanced on Age. Here we got some issues in using SMOTE: since there are too few examples in some subcategories, replicas are introduced by the algorithm. It was indeed impossible to balance Ethnicity for each balanced Age subcategory (opposite order), but this combination was still acceptable for metrics evaluation.

And 3 balanced datasets for Crack:

- balanced_data0: balanced for Crack.
- balanced_data1: balanced for Crack and Ethnicity.
- balanced_data2: balanced for Crack and Age. In this case, SMOTE could not perform Age balancing within each balanced subcategory of Crack, due to few examples in subcategories, then we simply run SMOTE for Age on the whole balanced_data0.

For this dataset we faced dimension problems: when we classify the datasets we get a little more than 500 rows and the fairness metric computation does not work properly on such small datasets, especially when there are underrepresented subcategories in some attributes. In particular in Crack case, we had too few samples for Crack consumers and we had to create balanced datasets always balanced towards Crack in order to get fairness metrics values. Anyway, even if all these balanced datasets are balanced towards Crack, it is not surprising to see that some metrics have better mean deviations in case we *only* balance Crack. Indeed,

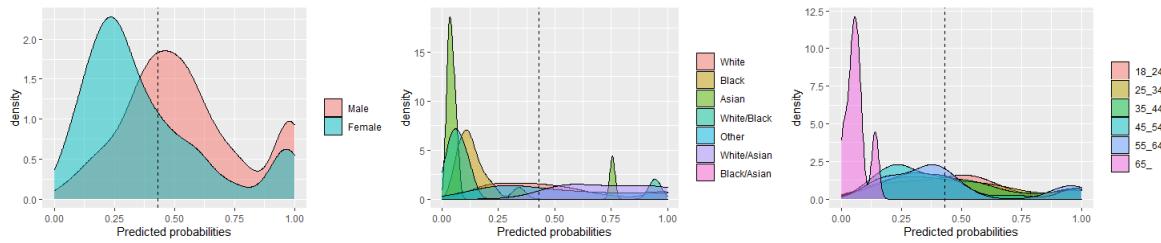


Figure 5.22: Drug dataset Coke density probabilities

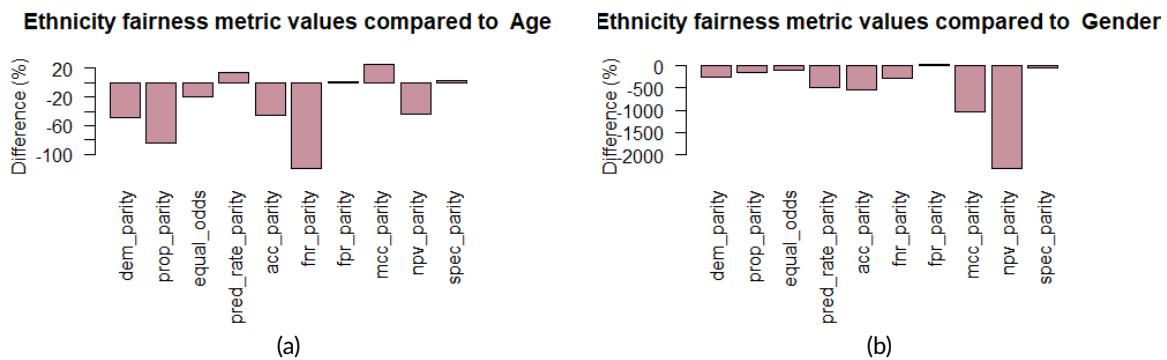


Figure 5.23: Drug dataset Coke fairness metrics comparisons between Ethnicity VS (a) Age and (b) Gender.

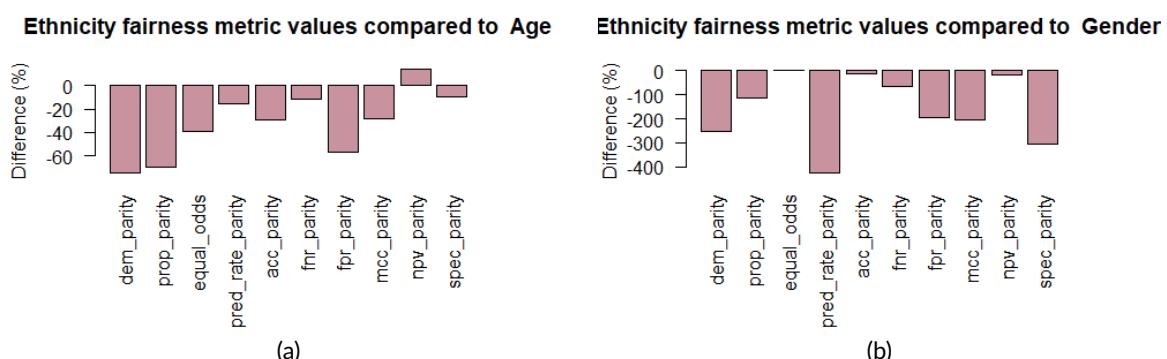


Figure 5.24: Drug dataset Crack fairness metrics comparisons between Ethnicity VS (a) Age and (b) Gender.

when we try to balance a couple of attributes with SMOTE or ROSE, it does not guarantees to reach a fully balancing for both attributes if we cannot perform the algorithm within each balanced subcategory of Crack. Then, for balanced_data1 and balanced_data2 we do not get a 50-50 proportions for Crack subcategories and this can influence the fairness metric outcomes.

We reported here in figure [5.25](#) and [5.26](#) the Mean deviations from complete fairness of balanced classified datasets and original classified dataset (Classified_dt).



Figure 5.25: Mean deviations from complete fairness, for each fairness metric in Coke case, for balanced classified datasets and original classified datasets



Figure 5.26: Mean deviations from complete fairness, for each fairness metric in Crack case, for balanced classified datasets and original classified datasets

Conclusions

Automated decision systems are likely to be used more and more in every process of our ordinary everyday life. Our first initial analysis - concerning how these systems are being used in the past and the kind of bias and discrimination they may lead to - brings evidence of different and crucial influence of the decisions issued by such systems on different groups of population. In addition, lack of transparency and accountability caused the developing organizations to be criticized and exposed to further investigation. Discriminating behaviors may arise for many reasons, hence before deployment of such tools it is necessary a risks assessment and evaluation of their impact on the society. Thus, it has an ethical importance to investigate whether such algorithms provide outcomes that can declass, demerit, or exclude individuals belonging to disadvantaged groups.

6.1 | Achieved Aims and Objectives

With this research we intended to carry out a comparative analysis of the various known fairness metrics, for the evaluation of classification systems, which are actually a small part of all the automatic decision systems that we can deal with today. In particular, we observed how the metrics varied based on the balance of the datasets, which is reflected in an alteration of the bias present in the data.

We also made our contribution to research on fairness in machine learning by focusing in particular on the balancing of sensitive attributes, which leads to the reduction of selection and population bias.

6.2 | Critique and Limitations

Surely a great limitation in the field of fairness in machine learning lies in the fact that every situation that interfaces with human and / or social variables requires an *ad hoc* study, dedicated only to particular circumstances and situations. It is not possible to recreate automated

tools for the removal of absolute discrimination, in any context. Furthermore, even the fact that each sensitive attribute must be analyzed in its own right, did not allow us to carry out a cross or aggregate analysis of all the attributes, for an overall assessment of discrimination. Another limitation is certainly due to the innumerable presence of types of bias (here we have created a taxonomy of the best known, but it is constantly evolving), which therefore could not be analyzed all, specifically for our datasets.

A last one also concerns the difficulty in actually being able to quantize a data such as fairness: numerous researches have now been conducted, but it is not possible - and it would certainly not be ethically correct - to establish (only) numerically whether a process is more or less discriminating than a other.

6.3 | Future Work

This research can be deepened with various analyzes of other bias mitigation techniques, such as those provided by the IBM AIF360 tool. Besides, it is possible to create a tool for comparing fairness metrics that takes into account other factors, such as the type of attribute being analyzed or parameters relating to the context of application (e.g. legal context, recruiting systems etc.), perhaps through benchmarks obtained from previous analyzes.

A further point of study is the combination and comparison of the various techniques of de-bias, to study whether there could be a link between certain social contexts and more marked improvements by means of certain specific techniques.

R code

R code for COMPAS dataset:

```
1 ## -----
2 #prendo il dataset dalla libreria fairness di CRAN
3 library(fairness)
4 library(knitr)
5 library(recipes)
6 library(skimr)
7 library(modeldata)
8 library(RColorBrewer)
9 library(tangram)
10 library(dplyr)
11 library(vegan)
12 library(diverse)
13 library(randomForest)
14 library(caret)
15 library(varhandle)
16 library(MASS)
17 library(dostats) #per %contains%
18 library(graphics) #per mosaicplot
19 library(corrplot)
20 library(wesanderson)
21 library(Amelia)
22 library(ROCR) #per ROC function
23 library(ROSE)#binary balancing
24 library("UBL")#multiclass balancing
25 library(ggplot2)
26 library(ggpubr)
27 library(xtable)
28 library(tidyr)
29
30 #carico il file util con le altre funzioni
31 source("util.R")
32
```

```

33
34 ## -----
35 knitr::purl('COMPAS.Rmd')
36
37
38 ## -----
39 #load data
40 raw_data = fairness::compas
41 write.csv(raw_data,'COMPAS.csv')
42 #in questo dataset gi stato fatto del lavoro rispetto al dataset originale. Tra l'altro: nel
43   ds originale c'era anche il marital status, che qui viene eliminato (probabilmente perch
44   si tratta di un dato personale e non sensibile?), poi lo score nel ds originale
45   presentato in modo diverso (raw score, decile score, text score...)
46 h = head(raw_data)
47 #print(xtable(h, type = "latex"), file = "COMPAShead.tex")
48 #probability e predicted per ora non ci interessano, perci eliminiamo le due colonne
49 raw_data$predicted = NULL
50 raw_data$probability = NULL
51
52
53 #sostituisci yes\no di 2yrRecidivism con Recidivism\No recidivism
54 levels(raw_data$Two_yr_Rcidivism)[levels(raw_data$Two_yr_Rcidivism)=="yes"] <- "Yes_
      recidivism"
55 levels(raw_data$Two_yr_Rcidivism)[levels(raw_data$Two_yr_Rcidivism)=="no"] <- "No_
      recidivism"
56
57 #crea una sola colonna di age, con under25/over45/between25_45
58 raw_data$Age <- mapply(create_Age, raw_data$Age_Below_TwentyFive, raw_data$Age_Above_
      FourtyFive)
59 raw_data$Age = as.factor(raw_data$Age)
60
61 h = head(raw_data)
62 #print(xtable(h, type = "latex"), file = "COMPAShead.tex")
63 summary(raw_data)
64
65 #data visualization CATEGORICAL
66
67 #Filter out categorical columns
68 cat_data <- raw_data[, sapply(raw_data, is.categorical)]
69 cat_names <- names(cat_data)
70
71 #pie print per ogni colonna categorica
72 for ( i in seq(1,length( cat_data ),1) ){
73   pie_print(cat_data,cat_names[i])
74 }
75
76 a = table(raw_data)
77 a = as.data.frame(a)
78 #a$Number_of_Priors = NULL

```

```
75 ggballoonplot(a, x = "ethnicity", y = "Age", size = "Freq",
76             color = "#8591C2", fill = "#8591C2", facet.by = "Female",
77             ggtheme = theme_bw()) #+
78 #scale_fill_gradientn(colors = wes_palette(2, name="GrandBudapest"))
79
80 #seleziono yes_recidivism e plotto la stessa cosa
81 raw_data_copy = raw_data
82 a = subset(raw_data_copy, Two_yr_Recidivism == "Yes_recidivism")
83 a = table(a)
84 a = as.data.frame(a)
85 ggballoonplot(a, x = "ethnicity", y = "Age", size = "Freq",
86             color = "#F24D4D", fill = "#F24D4D", facet.by = "Female",
87             ggtheme = theme_bw())
88
89 #seleziono no_recidivism e plotto la stessa cosa
90 a = subset(raw_data_copy, Two_yr_Recidivism == "No_recidivism")
91 a = table(a)
92 a = as.data.frame(a)
93 ggballoonplot(a, x = "ethnicity", y = "Age", size = "Freq",
94             color = "#50C395", fill = "#50C395", facet.by = "Female",
95             ggtheme = theme_bw())
96
97 ## -----
98 #missing values
99 missmap(raw_data, main = "Missing values vs observed")
100 #no missing values
101
102 # Create a boxplot using base R
103 boxplot(Number_of_Priors~Female,data=raw_data)
104 boxplot(Number_of_Priors~Age,data=raw_data)
105 boxplot(Number_of_Priors~ethnicity,data=raw_data)
106
107 #sembra che ci siano parecchi outlier del numero dei precedenti in funzione degli attributi
108     sensibili, che potrebbero quindi non essere outlier o alcuni essere dovuti ad errori di
109     registrazione
110
111 boxplot(Number_of_Priors~Misdemeanor,data=raw_data)
112 #ci sono anche degli outlier del numero dei precedenti in funzione del fatto che il crimine
113     sia stato (o meno) registrato
114
115 ## -----
116 #plotto lo Two_yr_recidivism al variare di Female
117 ##########
118 d = group_by_feature_sensitive(raw_data, "Two_yr_Recidivism", "Female")
119 bar(dv = count_val,
120      factors = c("Two_yr_Recidivism", "Female"), #non riesco a risolvere il problema di
121      passare in questa lista una variabile contenente la stringa, invece della stringa stessa.
122      dataframe = as.data.frame(d),
```

```

119     errbar = FALSE,
120     ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4])),
121     col = c("#50C395", "#F24D4D")
122   ) #I increased the upper y-limit to accommodate the legend.
123
124 #####ethnicity
125 d = group_by_feature_sensitive(raw_data, "Two_yr_Recidivism", "ethnicity")
126 bar(dv = count_val,
127   factors = c("Two_yr_Recidivism", "ethnicity"), #non riesco a risolvere il problema di
128   passare in questa lista una variabile contenente la stringa, invece della stringa stessa.
129   dataframe = as.data.frame(d),
130   errbar = FALSE,
131   ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4])),
132   col = c("#50C395", "#F24D4D")
133 ) #I increased the upper y-limit to accommodate the legend.
134 #####Misdemeanor
135 d = group_by_feature_sensitive(raw_data, "Two_yr_Recidivism", "Misdemeanor")
136 bar(dv = count_val,
137   factors = c("Two_yr_Recidivism", "Misdemeanor"), #non riesco a risolvere il problema di
138   passare in questa lista una variabile contenente la stringa, invece della stringa stessa.
139   dataframe = as.data.frame(d),
140   errbar = FALSE,
141   ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4])),
142   col = c("#50C395", "#F24D4D")
143 ) #I increased the upper y-limit to accommodate the legend.
144 #####Age
145 d = group_by_feature_sensitive(raw_data, "Two_yr_Recidivism", "Age")
146 bar(dv = count_val,
147   factors = c("Two_yr_Recidivism", "Age"), #non riesco a risolvere il problema di passare
148   in questa lista una variabile contenente la stringa, invece della stringa stessa.
149   dataframe = as.data.frame(d),
150   errbar = FALSE,
151   ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4])),
152   col = c("#50C395", "#F24D4D")
153 ) #I increased the upper y-limit to accommodate the legend.
154
155 ## -----
156 #NUMERICAL COLUMN
157 #Number_of_Priors come lo visualizzo/interpreto? essendo normalized to mean = 0 and standard
158 deviation = 1
159
160 ## -----
161 #con frquent_comb1 non devo dividere "manualmente" tra i due valori di Two_yr_Recidivism (cio
162 yes_recidivism e no_recidivism), ma lo fa in automatico, restituendomi poi due tabelle.

```

Per capire quale sia riferita a yes e quale a no, occorre guardare le colonne: in una tabella presente la colonna "percentage_on_yes_recidivism" e nell'altra "percentage_on_no_recidivism" e le due tabelle sono quindi riferite rispettivamente a "yes_recidivism" e "no_recidivism"

```

162 frquent_comb1(raw_data, filter = "Two_yr_Recidivism", c("Age", "ethnicity"))
163 frquent_comb1(raw_data, filter = "Two_yr_Recidivism", c("Age", "ethnicity", "Female"))
164
165 #plotto la distribuzione delle combinazioni (in proporzione )
166
167 frquent_comb1(raw_data, filter = "Two_yr_Recidivism", c("Age", "Female"))
168 frquent_comb1(raw_data, filter = "Two_yr_Recidivism", c("ethnicity", "Female"))
169 frquent_comb1(raw_data, filter = "Two_yr_Recidivism", c("ethnicity"))
170 ##
171 #COME SI LEGGE LA TABELLA?
172 #esempio con frquent_comb(raw_data, data_good, c("Marital", "ageR"))
173 #dataset originale: raw_data
174 #dataset filtrato: data_good (filtro su status=good)
175 #attributi che mi interessano: c("Marital", "ageR") -> pu essere una lista di lunghezza
#      variabile
176 #-Percentage_on_partition rappresenta la probabilit condizionata: es. P(marital=x,ageR=y|
#      status=good).
177 #-percentage_on_total la percentuale della coppia selezionata (qui ad esempio data dalle
#      combinazioni di marital e ageR) sul totale delle righe del dataset originale. p = #
#      (status=good, marital=x, ageR=y)/(numero righe dataset originale)
178 #-numrows_filter_on_col un parametro che non mi interessa in output, ma non riesco ad
#      eliminarlo dallpa funzione
179 #-percentage_on_Marital_ageR la probabilit condizionata p(good| marital=x, ageR=y) = #
#      (status=good, marital=x, ageR=y)/(marital=x, ageR=y)
180 #####
181 #esempio: dato Status="good" probabile al 25,81% cheo si tratti di una persona sposata, tra
#      i 30 e 40 anni.
182 #esempio: dato Status="bad" probabile al 14.49% che si tratti di un single sotto i 25 anni.
183 #####
184
185
186
187 ## -----
188 #####
189 #plotto la distribuzione delle combinazioni (conteggio casi)
190 a = frquent_comb1(raw_data, filter = "Two_yr_Recidivism", c("Age", "ethnicity", "Female"))
191
192 a1 = as.data.frame(a[1]) #la tabella 1 ha sempre no_recidivism
193 a2 = as.data.frame(a[2]) #la tabella 2 ha sempre yes_recidivism
194
195 ggballoonplot(a1, x = "ethnicity", y = "Age", size = "percentage_on_No_recidivism",
#                  color = "black", fill = "#50C395", facet.by = "Female", ggtheme = theme_bw())
196 #percentage_on_Age_ethnicity_Female
197
198

```

```

199 ggballoonplot(a2, x = "ethnicity", y = "Age", size = "percentage_on_Yes_recidivism",
200             color = "black", fill = "#F24D4D", facet.by = "Female", ggtheme = theme_bw())
201
202 ggballoonplot(a1, x = "ethnicity", y = "Age", size = "percentage_on_total",
203             color = "black", fill = "#50C395", facet.by = "Female", ggtheme = theme_bw())
204 #percentage_on_Age_ethnicity_Female
205
206 ggballoonplot(a2, x = "ethnicity", y = "Age", size = "percentage_on_Age_ethnicity_Female",
207             color = "black", fill = "#F24D4D", facet.by = "Female", ggtheme = theme_bw())
208
209 ggballoonplot(a1, x = "ethnicity", y = "Age", size = "percentage_on_Age_ethnicity_Female",
210             color = "black", fill = "#50C395", facet.by = "Female", ggtheme = theme_bw())
211 #percentage_on_Age_ethnicity_Female
212
213 ggballoonplot(a2, x = "ethnicity", y = "Age", size = "percentage_on_total",
214             color = "black", fill = "#F24D4D", facet.by = "Female", ggtheme = theme_bw())
215
216 a1$percentage_on_No_recidivism = NULL
217 a1$percentage_on_total = NULL
218 a1$percentage_on_Age_ethnicity_Female = NULL
219 a2$percentage_on_Yes_recidivism = NULL
220 a2$percentage_on_total = NULL
221 a2$percentage_on_Age_ethnicity_Female = NULL
222 print(xtable(a1, type = "latex"), file = "No_recFreqComb.tex")
223 print(xtable(a2, type = "latex"), file = "Yes_recFreqComb.tex")
224 #####
225
226 ## -----
227 ###OCCHIO: qui il bad e good del valore target (in questo caso recidivism) devono avere un
228     certo ordine
229 #source("util.R")
230 mosaicp(raw_data, c("Two_yr_Recidivism", "Age"))
231 mosaicp(raw_data, c("Two_yr_Recidivism", "ethnicity"))
232 mosaicp(raw_data, c("Two_yr_Recidivism", "Female"))
233
234 #Il blu indica che il valore osservato pi alto del valore atteso se i dati fossero random
235 #Il rosso indica che il valore osservato pi basso del valore atteso se i dati fossero random
236
237 #gi da questo mosaicplot possiamo capire cosa aspettarci per le metriche di fairness. es.
238     attributo Female: evidente che per gli uomini ci sar pi disparit, perch verranno
239     categorizzati Yes_recid pi delle donne. Anche per gli african-american o per quelli
240     sottoi 25 anni.
241
242 ## -----
243 #chi squared test (consideriamo le colonne come categoriche)
244 source("util.R")
245 chisq_fun(raw_data, "Two_yr_Recidivism", "Age")
246 chisq_fun(raw_data, "Two_yr_Recidivism", "ethnicity")

```

```
243 chisq_fun(raw_data, "Two_yr_Recidivism", "Female")
244 #In console possiamo osservare la differenza tra i valori osservati (prima tabella) e i
#      valori attesi (seconda tabella).
245
246
247 ## -----
248 #SHANNON INDEX
249 #uso funzioni create nella libreria
250 source("util.R")
251 shannon_diversity(raw_data)
252 print_shannon(raw_data)
253 #print_cmp_shannon(raw_data, raw_data)
254
255 ## -----
256 #sostituisco i livelli con valori numerici
257 levels(raw_data$Two_yr_Recidivism)[levels(raw_data$Two_yr_Recidivism)=="yes"] <- 1
258 levels(raw_data$Two_yr_Recidivism)[levels(raw_data$Two_yr_Recidivism)=="no"] <- 0
259 #rimuovo da raw_data le colonne age_above e age_below
260 cols_to_drop <- names(raw_data) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
261 raw_data <- raw_data[, !cols_to_drop, drop = FALSE]
262
263 #split into train and test set
264 sample_size = floor(0.7*nrow(raw_data))
265 #set a random seed per rendere l'esperimento ripetibile
266 set.seed(42)
267
268 # randomly split data
269 picked = sample(seq_len(nrow(raw_data)), size = sample_size)
270 training_set = raw_data[picked,]
271 test_set = raw_data[-picked,]
272
273 #as.factor mi serve per fare una classificazione binaria
274 raw_data$Two_yr_Recidivism = as.factor(raw_data$Two_yr_Recidivism)
275 #devo anche eliminare recidivismo dal training_set
276 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
277 #training e test set senza variabile target
278 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
279 #anche sui dati di test
280 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
281
282 ## -----
283 #REGRESSIONE LOGISTICA
284 #in questo caso ho bisogno del dataset numerico
285 #training_set = as.data.frame(lapply(unclass(training_set), as.numeric))
286 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_"
#      recidivism"] <- 1
287 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism"
#"] <- 0
```

```

288 levels(training_set$Two_yr_Rcidivism)
289
290 levels(test_set$Two_yr_Rcidivism)[levels(test_set$Two_yr_Rcidivism)=="Yes_recidivism"] <- 1
291 levels(test_set$Two_yr_Rcidivism)[levels(test_set$Two_yr_Rcidivism)=="No_recidivism"] <- 0
292 levels(test_set$Two_yr_Rcidivism)
293 #####
294 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
295 logitMod <- glm(Two_yr_Rcidivism ~ ., data = training_set, family=binomial(link="logit"))
296 logitMod #capire cosa significano questi dati
297
298 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
299 classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria fairness
300 classified_dt2 = cbind(test_set_data, pred)
301
302
303 ## -----
304 #ROC
305 pred1 <- prediction(pred, test_set$Two_yr_Rcidivism) #pred1 un oggetto pred.obj di ROCR
306 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
307 plot(perf)
308 #posso anche fare una ROC comparison per le categorie di un attributo sensibile, tipo Female,
# con la libreria fairness:
309 roc_fairness <- roc_parity(data      = classified_dt,
310                               outcome   = 'Two_yr_Rcidivism',
311                               group     = 'Female',
312                               probs     = 'pred',
313                               preds_levels = c('No_recidivism', 'Yes_recidivism'),
314                               base      = 'Male')
315 roc_fairness$Metric
316 roc_fairness$ROCAUC_plot
317
318 #AUC
319 auc.perf = performance(pred1, measure = "auc")
320 auc.perf@y.values #stampa a video il valore di AUC
321 #####
322 #plotto sensitivity e specificity in modo da individuare il cutoff ottimizzato
323 sens <- data.frame(x=unlist(performance(pred1, "sens")@x.values),
324                      y=unlist(performance(pred1, "sens")@y.values))
325 spec <- data.frame(x=unlist(performance(pred1, "spec")@x.values),
326                      y=unlist(performance(pred1, "spec")@y.values))
327
328 sens %>% ggplot(aes(x,y)) +
329   geom_line() +
330   geom_line(data=spec, aes(x,y,col="red")) +

```

```
331 scale_y_continuous(sec.axis = sec_axis(~., name = "Specificity")) +
332   labs(x='Cutoff', y="Sensitivity") +
333   theme(axis.title.y.right = element_text(colour = "red"), legend.position="none")
334 #####
335
336
337 ## -----
338 #troviamo l'optimal cutoff, cio l'optimal score che minimizza il misclassification error del
#      modello di regressione logistica
339 #The best threshold (or cutoff) point to be used in glm models is the point which maximises
#      the specificity and the sensitivity. This threshold point might not give the highest
#      prediction in your model, but it wouldn't be biased towards positives or negatives.
340 optimized_cutoff = opt.cut(perf, pred1)#il cutoff ottimale leggermente piu basso di quello
#      standard, che sarebbe invece 0.5
341
342 #convert la colonna pred di classified_dt2 in yes/no recidivism usando il cutoff ottimizzato
343 #yes era 1, quindi ho recidivismo se maggiore del cutoff
344 classified_dt2$pred = ifelse(classified_dt2$pred<optimized_cutoff,"No_recidivism","Yes_
#      recidivism")
345 #sostituisco i valori dei livelli per poter pottare la matrice di confusione
346 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)==1] <- "Yes_recidivism"
347 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)==0] <- "No_recidivism"
348 levels(test_set$Two_yr_Recidivism)
349 #anche per classified_dt2
350 classified_dt2$pred <- factor(classified_dt2$pred, levels = c("Yes_recidivism", "No_
#      recidivism"))
351
352
353 ## -----
354 #matrice di confusione
355 cm <- confusionMatrix(data = as.factor(classified_dt2$pred), reference = as.factor(test_set$Two_yr_Recidivism))
356 #plotto la matrice di confusione
357 draw_confusion_matrix(cm)
358
359 ## -----
360 #ci occupiamo di group fairness: indipendenza, separazione e sufficienza
361
362 #lo facciamo per tutti i sensitive attribute
363
364 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
365 levels(classified_dt$Two_yr_Recidivism)[levels(classified_dt$Two_yr_Recidivism)==1] <- "Yes_
#      recidivism"
366 levels(classified_dt$Two_yr_Recidivism)[levels(classified_dt$Two_yr_Recidivism)==0] <- "No_
#      recidivism"
367
368 #per scegliere la categoria "base" per cui confrontare tutte le altre, per ogni attributo
#      sensibile, ci conviene guardare le tabelle delle frequenze plottate in precedenza:
```

```

369
370
371 ## -----
372 fair_metrics(data      = classified_dt,
373                 outcome     = 'Two_yr_Recidivism',
374                 group       = 'Age',
375                 probs      = 'pred',
376                 preds_levels = c('No_recidivism', 'Yes_recidivism'),
377                 cutoff      = optimized_cutoff,
378                 base        = 'between25_45')
379
380 fair_metrics(data      = classified_dt,
381                 outcome     = 'Two_yr_Recidivism',
382                 group       = 'ethnicity',
383                 probs      = 'pred',
384                 preds_levels = c('No_recidivism', 'Yes_recidivism'),
385                 cutoff      = optimized_cutoff,
386                 base        = 'Caucasian')
387
388 fair_metrics(data      = classified_dt,
389                 outcome     = 'Two_yr_Recidivism',
390                 group       = 'Female',
391                 probs      = 'pred',
392                 preds_levels = c('No_recidivism', 'Yes_recidivism'),
393                 cutoff      = optimized_cutoff,
394                 base        = 'Male')
395 source("util.R")
396 fair_metrics_cmp_graph2(data      = classified_dt,
397                           outcome     = 'Two_yr_Recidivism',
398                           group1     = 'Female',
399                           group2     = 'ethnicity',
400                           probs      = 'pred',
401                           preds_levels = c('No_recidivism', 'Yes_recidivism'),
402                           cutoff      = optimized_cutoff,
403                           base1      = 'Male',
404                           base2      = 'Caucasian')
405 fair_metrics_cmp_graph2(data      = classified_dt,
406                           outcome     = 'Two_yr_Recidivism',
407                           group1     = 'Age',
408                           group2     = 'ethnicity',
409                           probs      = 'pred',
410                           preds_levels = c('No_recidivism', 'Yes_recidivism'),
411                           cutoff      = optimized_cutoff,
412                           base1      = 'between25_45',
413                           base2      = 'Caucasian')
414
415
416 ## -----

```

```
417 #la libreria ROSE si pu usare soltanto per bilanciare attributi binari (nasce per bilanciare  
418 # classificazione binaria)  
419 library("ROSE")  
420 balanced_data = ovun.sample(Female~, data = raw_data, method="both", N = nrow(raw_data), p  
421 =0.5, seed = 42)$data  
422 print_cmp_shannon(raw_data, balanced_data)  
423  
424 #library(imbalance)  
425 #plotComparison(balanced_data, raw_data, attrs = c("ethnicity", "Number_of_Priors"), cols =  
426 # 2, classAttr = "Two_yr_Recidivism")  
427  
428 #Filter out categorical columns  
429 cat_data <- balanced_data[, sapply(balanced_data, is.categorical)]  
430 cat_names <- names(cat_data)  
431 #pie print per ogni colonna categorica  
432 for ( i in seq(1,length( cat_data ),1) ){  
433 pie_print(cat_data,cat_names[i])  
434 }  
435 library("UBL")  
436 #questo per il multiclass con la libreria UBL  
437 source("util.R")  
438 sample_size = floor(nrow(raw_data)/3)  
439 #set a random seed per rendere l'esperimento ripetibile  
440 set.seed(10)  
441 data_reduced = raw_data[sample(nrow(raw_data), sample_size), ]  
442 balanced_data = AdasynClassif(ethnicity~, data_reduced, beta = 1, dist = "HEOM", k=1)  
443 C.perc = list(Caucasian = 0.1, Asian = 0.1, Other=0.1, African_American=0.1, Hispanic=0.1,  
444 Native_American=0.1)  
445  
446 #balanced_data = adasyn_mod(ethnicity~, data_reduced, beta = 1, dist = "HEOM", k=1) #NON  
# POSSO FARLO COS: Error in class.freq(dat, tgt) : non trovo la funzione "class.freq"  
447 #Filter out categorical columns  
448 cat_data <- balanced_data[, sapply(balanced_data, is.categorical)]  
449 cat_names <- names(cat_data)  
450 #pie print per ogni colonna categorica  
451 for ( i in seq(1,length( cat_data ),1) ){  
452 pie_print(cat_data,cat_names[i])  
453 }  
454 #print_cmp_shannon(raw_data, balanced_data1)  
455  
456 ## -----  
457 #ora provo a riplottare le metriche di fairness per ethnicity  
458 #quindi prima faccio di nuovo sampling, regressione e trovo dataset classificato  
459 source("util.R")  
460 #sostituisco i livelli con valori numerici  
461 levels(balanced_data$Two_yr_Recidivism)[levels(balanced_data$Two_yr_Recidivism)=="yes"] <- 1
```

```
460 levels(balanced_data$Two_yr_Recidivism)[levels(balanced_data$Two_yr_Recidivism)=="no"] <- 0
461 #rimuovo da balanced_data le colonne age_above e age_below
462 cols_to_drop <- names(balanced_data) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
463 balanced_data <- balanced_data[, !cols_to_drop, drop = FALSE]
464
465 #split into train and test set
466 sample_size = floor(0.7*nrow(balanced_data))
467 #set a random seed per rendere l'esperimento ripetibile
468 set.seed(42)
469
470 # randomly split data
471 picked = sample(seq_len(nrow(balanced_data)), size = sample_size)
472 training_set = balanced_data[picked,]
473 test_set = balanced_data[-picked,]
474
475 #as.factor mi serve per fare una classificazione binaria
476 balanced_data$Two_yr_Recidivism = as.factor(balanced_data$Two_yr_Recidivism)
477 #devo anche eliminare recidivismo dal training_set
478 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
479 #training e test set senza variabile target
480 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
481 #anche sui dati di test
482 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
483
484
485
486
487 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
488 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
489 levels(training_set$Two_yr_Recidivism)
490
491 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
492 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
493 levels(test_set$Two_yr_Recidivism)
494 #####
495 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
496 logitMod <- glm(Two_yr_Recidivism ~ ., data = training_set, family=binomial(link="logit"))
497 logitMod #capire cosa significano questi dati
498
499 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
500 balanced_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
# fairness
```

```

501 classified_dt2 = cbind(test_set_data, pred)
502 #classified_dt = rbind(classified_dt1, classified_dt2)
503
504 pred1 <- prediction(pred, test_set$Two_yr_Recidivism) #pred1 un oggetto pred.obj di ROCR
505 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
506 optimized_cutoff_prova = opt.cut(perf, pred1)#il cutoff ottimale leggermente piu basso di
      quello standard, che sarebbe invece 0.5
507
508 #ripristino di nuovo i livelli yes/no per renderlo piu leggibile
509 levels(balanced_classified_dt$Two_yr_Recidivism)[levels(balanced_classified_dt$Two_yr_
      Recidivism)==1] <- "Yes_recidivism"
510 levels(balanced_classified_dt$Two_yr_Recidivism)[levels(balanced_classified_dt$Two_yr_
      Recidivism)==0] <- "No_recidivism"
511
512 #plotto shannon compare per vedere come cambia l'indice rispetto al dataset originale per l'
      attributo ethnicity
513
514 #voglio provare a calcolare il valore medio delle predizioni per tutti i caucasian: PER
      PROPORTIONAL PARITY FA PROPRIO QUESTO!!!!
515 cdt_caucasian = classified_dt
516 cdt_caucasian = classified_dt[classified_dt$ethnicity=="Caucasian",]
517 cdt_caucasian$pred=ifelse(cdt_caucasian$pred<optimized_cutoff,0,1)
518 m = mean(cdt_caucasian$pred)
519 m #questa m quella che io avrei chiamato percentage on ethnicity nella mia funzione frquent_
      comb1, cio percentage_on_ethnicity la probabilit condizionata p(good| ethnicity=x) = #((
      status=good, ethnicity=x)/(ethnicity=x)
520
521 cdt_caucasian = classified_dt
522 cdt_caucasian$pred=ifelse(cdt_caucasian$pred<optimized_cutoff,0,1)
523 frquent_comb1(cdt_caucasian, filter = "pred", c("ethnicity"))
524 #####
525 levels(classified_dt$Two_yr_Recidivism)
526
527 #####
528
529 fair_metrics_cmp(data1          = classified_dt,
530                   data2 = balanced_classified_dt,
531                   outcome       = 'Two_yr_Recidivism',
532                   outcome_base = 'No_recidivism',
533                   group         = 'ethnicity',
534                   probs        = 'pred',
535                   preds_levels = c('No_recidivism','Yes_recidivism'),
536                   cutoff1       = optimized_cutoff,
537                   cutoff2       = optimized_cutoff_prova,
538                   base          = 'Caucasian')
539
540 # fair_metrics(data          = classified_dt,
541 #                  outcome       = 'Two_yr_Recidivism',

```

```
542 #           group      = 'ethnicity',
543 #           probs     = 'pred',
544 #           preds_levels = c('No_recidivism', 'Yes_recidivism'),
545 #           cutoff     = optimized_cutoff,
546 #           base       = 'Caucasian')
547
548
549
550 ## -----
551 #CREO I SEGUENTI DATASET BILANCIATI:
552 #balanced_data0: bilanciato per Two_yr_recidivism
553 #balanced_data1: bilanciato per Female
554 #balanced_data2: bilanciato per ethnicity
555 #balanced_data3: bilanciato per Age
556 #balanced_data4: bilanciato per Two_yr_recidivism|ethnicity
557
558 #i due successivi li scarterei:
559 #balanced_data5: bilanciato per Female|ethnicity
560 #balanced_data6: bilanciato per Age|ethnicity
561
562 #balanced_data0: bilanciato per Two_yr_recidivism
563 balanced_data0=NULL
564 balanced_data0 = ovun.sample(Two_yr_Recidivism~, data = raw_data, method="both", N = nrow(
  raw_data), p=0.5, seed = 42)$data
565 print_cmp_shannon(raw_data, balanced_data0)
566 #shannon index praticamente uguale per Two_yr, perci questo potrebbe essere inutile (non ho
  cambiato il grado diversit in questo attributo, perci dovrei avere la stessa proporzione
  di individui yes/no recidivism)
567
568 #####
569 #balanced_data1: bilanciato per Female
570 #la libreria ROSE si pu usare soltanto per bilanciare attributi binari (nasce per bilanciare
  classificazione binaria)
571 balanced_data1 = NULL
572 balanced_data1 = ovun.sample(Female~, data = raw_data, method="both", N = nrow(raw_data), p
  =0.5, seed = 42)$data
573 print_cmp_shannon(raw_data, balanced_data1)
574 #shannon index migliorato per Female (cio riduzione di diversit)
575
576 #####
577 #balanced_data2: bilanciato per ethnicity con un random balance che ho creato io
578 #balanced_data2 = rebalance_multiclass(raw_data, "ethnicity")
579
580 #questo invece ha ethnicity bilanciato con smote ed meglio usare questo!
581 balanced_data2 = SmoteClassif(ethnicity ~ ., raw_data, dist = "HEOM")
582
583 print_cmp_shannon(raw_data, balanced_data2)
584 #shannon index aumentato per ethnicity
```

```
585  
586 #####  
587 #balanced_data3: bilanciato per Age  
588 balanced_data3 = rebalance_multiclass(raw_data, "Age")  
589  
590 #questo invece ha ethnicity bilanciato con smote ed meglio usare questo!  
591 balanced_data3 = SmoteClassif(Age ~ ., raw_data, dist = "HEOM")  
592  
593 print_cmp_shannon(raw_data, balanced_data3)  
594 #shannon index leggermente aumentato per Age  
595  
596  
597 #####  
598 #balanced_data4: bilanciato per Two_yr_recidivism|ethnicity (cio voglio ogni sottocategoria  
      di ethnicity, gi precedentemente bilanciato (balanced_data2), bilanciata anche rispetto  
      al recidivismo )  
599 balanced_data4 = NULL  
600 for(i in levels(balanced_data2$ethnicity)){  
601   #faccio il rebalancing di Two_yr_Recidivism per ogni categoria di ethnicity  
602   temp_dt = balanced_data2[balanced_data2$ethnicity==i, ]  
603   s = ovun.sample(Two_yr_Recidivism~, data = temp_dt, method="both", N = nrow(temp_dt), p  
      =0.5, seed = 42)$data  
604   #metto poi insieme i vari piccoli dataset bilanciati  
605   balanced_data4 = rbind(balanced_data4, s)  
606 }  
607  
608 #plotto entrambi i confronti dello shannon index  
609 print_cmp_shannon(raw_data, balanced_data4) #ethnicity aumentato di molto (ma c'era da  
      aspettarsel, visto che il confronto con il dataset originale)  
610 print_cmp_shannon(balanced_data2, balanced_data4)#non cambiato quasi nulla, quindi sembra  
      sia sufficiente fare un rebalancing anche solo su ethnicity per migliorare la situazione  
611  
612 #####  
613 #balanced_data5: bilanciato per Two_yr_recidivism|ethnicity (cio voglio ogni sottocategoria  
      di ethnicity bilanciata rispetto a Two_yr_recidivism)  
614 #rispetto al dataset precedente, qui non abbiamo entrambi gli attributi contemporaneamente  
      bilanciati vicendevolmente: ethnicity non affatto bilanciato qui!  
615 balanced_data5 = NULL  
616 for(i in levels(raw_data$ethnicity)){  
617   #faccio il rebalancing di Two_yr_Recidivism per ogni categoria di ethnicity  
618   temp_dt = raw_data[raw_data$ethnicity==i, ]  
619   s = ovun.sample(Two_yr_Recidivism~, data = temp_dt, method="both", N = nrow(temp_dt), p  
      =0.5, seed = 42)$data  
620   #metto poi insieme i vari piccoli dataset bilanciati  
621   balanced_data5 = rbind(balanced_data5, s)  
622 }  
623 print_cmp_shannon(raw_data, balanced_data5) #shannon praticamente uguali  
624 print_cmp_shannon(balanced_data4, balanced_data5)
```

```

625
626 #####
627 #bilancio ethnicity in ogni sottocategoria bilanciata di Female
628 balanced_data6 = NULL
629 for(i in levels(balanced_data1$ethnicity)){
630   #faccio il rebalancing di Two_yr_Recidivism per ogni categoria di ethnicity
631   temp_dt = balanced_data1[balanced_data1$ethnicity==i, ]
632   s = SmoteClassif(ethnicity ~ ., temp_dt, dist = "HEOM")
633   #metto poi insieme i vari piccoli dataset bilanciati
634   balanced_data6 = rbind(balanced_data6, s)
635 }
636 print_cmp_shannon(raw_data, balanced_data6) #molto meglio per ethnicity e un po' meglio per
637   female
638
639 #confronto con bilanciato solo su female
640 print_cmp_shannon(balanced_data1, balanced_data6) #meglio per ethnicity
641
642 #confronto con bilanciato solo su ethnicity
643 print_cmp_shannon(balanced_data2, balanced_data6) #meglio per Female
644 #####
645 balanced_data7 = NULL
646 #bilancio il gender in ogni sottocategoria bilanciata di ethnicity
647 for(i in levels(balanced_data2$Female)){
648   temp_dt = raw_data[raw_data$Female==i, ]
649   s = ovun.sample(Two_yr_Recidivism~, data = temp_dt, method="both", N = nrow(temp_dt), p
650     =0.5, seed = 42)$data
651   #metto poi insieme i vari piccoli dataset bilanciati
652   balanced_data7 = rbind(balanced_data7, s)
653 }
654 print_cmp_shannon(raw_data, balanced_data7) #molto meglio per ethnicity e un po' meglio per
655   female
656
657 summary(raw_data$Number_of_Priors)
658 summary(balanced_data2$Number_of_Priors)
659 sd(raw_data$Number_of_Priors)
660
661
662
663 ## -----
664 #balanced_data0: bilanciato per Two_yr_recidivism
665 #balanced_data1: bilanciato per Female
666 #balanced_data2: bilanciato per ethnicity
667 #balanced_data3: bilanciato per Age
668 #balanced_data4: bilanciato per Two_yr_recidivism|ethnicity
669

```

```
670 #otteniamo tutti i balanced dataset classificati:  
671 source("util.R")  
672  
673 #balanced_data0  
674  
675 #sostituisco i livelli con valori numerici  
676 levels(balanced_data0$Two_yr_Recidivism)[levels(balanced_data0$Two_yr_Recidivism)=="yes"] <-  
    1  
677 levels(balanced_data0$Two_yr_Recidivism)[levels(balanced_data0$Two_yr_Recidivism)=="no"] <- 0  
678 #rimuovo da balanced_data0 le colonne age_above e age_below  
679 cols_to_drop <- names(balanced_data0) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")  
680 balanced_data0 <- balanced_data0[, !cols_to_drop, drop = FALSE]  
681  
682 #split into train and test set  
683 sample_size = floor(0.7*nrow(balanced_data0))  
684 #set a random seed per rendere l'esperimento ripetibile  
685 set.seed(42)  
686  
687 # randomly split data  
688 picked = sample(seq_len(nrow(balanced_data0)), size = sample_size)  
689 training_set = balanced_data0[picked, ]  
690 test_set = balanced_data0[-picked, ]  
691  
692 #as.factor mi serve per fare una classificazione binaria  
693 balanced_data0$Two_yr_Recidivism = as.factor(balanced_data0$Two_yr_Recidivism)  
694 #devo anche eliminare recidivismo dal training_set  
695 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")  
696 #training e test set senza variabile target  
697 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]  
698 #anche sui dati di test  
699 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]  
700  
701 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_  
    recidivism"] <- 1  
702 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism"  
    "] <- 0  
703 levels(training_set$Two_yr_Recidivism)  
704  
705 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1  
706 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="No_recidivism"] <- 0  
707 levels(test_set$Two_yr_Recidivism)  
708 #####  
709 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (   
    #sufficiente salvarlo e ricaricarlo)  
710 logitMod <- glm(Two_yr_Recidivism ~ ., data = training_set, family=binomial(link="logit"))  
711 logitMod #capire cosa significano questi dati  
712
```

```

713 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
    mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
    tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
    fare la predizione
714 balanced0_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
    fairness
715 classified_dt2 = cbind(test_set_data, pred)
716 #classified_dt = rbind(classified_dt1, classified_dt2)
717 pred1 <- prediction(pred, test_set$Two_yr_Recidivism) #pred1 un oggetto pred.obj di ROC
718 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
719 optimized_cutoff0 = opt.cut(perf, pred1)#il cutoff ottimale leggermente piii basso di quello
    standard, che sarebbe invece 0.5
720
721 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
722 levels(balanced0_classified_dt$Two_yr_Recidivism)[levels(balanced0_classified_dt$Two_yr_
    Recidivism)==1] <- "Yes_recidivism"
723 levels(balanced0_classified_dt$Two_yr_Recidivism)[levels(balanced0_classified_dt$Two_yr_
    Recidivism)==0] <- "No_recidivism"
724
725 #####
726
727
728 ## -----
729 #balanced_data1
730
731 #sostituisco i livelli con valori numerici
732 levels(balanced_data1$Two_yr_Recidivism)[levels(balanced_data1$Two_yr_Recidivism)=="yes"] <-
    1
733 levels(balanced_data1$Two_yr_Recidivism)[levels(balanced_data1$Two_yr_Recidivism)=="no"] <- 0
734 #rimuovo da balanced_data1 le colonne age_above e age_below
735 cols_to_drop <- names(balanced_data1) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
736 balanced_data1 <- balanced_data1[, !cols_to_drop, drop = FALSE]
737
738 #split into train and test set
739 sample_size = floor(0.7*nrow(balanced_data1))
740 #set a random seed per rendere l'esperimento ripetibile
741 set.seed(42)
742
743 # randomly split data
744 picked = sample(seq_len(nrow(balanced_data1)), size = sample_size)
745 training_set = balanced_data1[picked,]
746 test_set = balanced_data1[-picked,]
747
748 #as.factor mi serve per fare una classificazione binaria
749 balanced_data1$Two_yr_Recidivism = as.factor(balanced_data1$Two_yr_Recidivism)
750 #devo anche eliminare recidivismo dal training_set
751 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
752 #training e test set senza variabile target

```

```
753 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
754 #anche sui dati di test
755 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
756
757 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_
    recidivism"] <- 1
758 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism"
    ] <- 0
759 levels(training_set$Two_yr_Recidivism)
760
761 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
762 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
763 levels(test_set$Two_yr_Recidivism)
764 #####
765 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
    sufficiente salvarlo e ricaricarlo)
766 logitMod <- glm(Two_yr_Recidivism ~ ., data = training_set, family=binomial(link="logit"))
767 logitMod #capire cosa significano questi dati
768
769 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
    mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
    tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
    fare la predizione
770 balanced1_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
    fairness
771 classified_dt2 = cbind(test_set_data, pred)
772 #classified_dt = rbind(classified_dt1, classified_dt2)
773 pred1 <- prediction(pred, test_set$Two_yr_Recidivism) #pred1 un oggetto pred.obj di ROCR
774 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
775 optimized_cutoff1 = opt.cut(perf, pred1)#il cutoff ottimale leggermente ppi basso di quello
    standard, che sarebbe invece 0.5
776
777 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
778 levels(balanced1_classified_dt$Two_yr_Recidivism)[levels(balanced1_classified_dt$Two_yr_
    Recidivism)==1] <- "Yes_recidivism"
779 levels(balanced1_classified_dt$Two_yr_Recidivism)[levels(balanced1_classified_dt$Two_yr_
    Recidivism)==0] <- "No_recidivism"
780
781 #####
782
783
784 ## -----
785 #balanced_data2
786
787 #sostituisco i livelli con valori numerici
788 levels(balanced_data2$Two_yr_Recidivism)[levels(balanced_data2$Two_yr_Recidivism)=="yes"] <-
    1
789 levels(balanced_data2$Two_yr_Recidivism)[levels(balanced_data2$Two_yr_Recidivism)=="no"] <- 0
```

```
790 #rimuovo da balanced_data2 le colonne age_above e age_below
791 cols_to_drop <- names(balanced_data2) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
792 balanced_data2 <- balanced_data2[, !cols_to_drop, drop = FALSE]
793
794 #split into train and test set
795 sample_size = floor(0.7*nrow(balanced_data2))
796 #set a random seed per rendere l'esperimento ripetibile
797 set.seed(42)
798
799 # randomly split data
800 picked = sample(seq_len(nrow(balanced_data2)), size = sample_size)
801 training_set = balanced_data2[picked,]
802 test_set = balanced_data2[-picked,]
803
804 #as.factor mi serve per fare una classificazione binaria
805 balanced_data2$Two_yr_Recidivism = as.factor(balanced_data2$Two_yr_Recidivism)
806 #devo anche eliminare recidivismo dal training_set
807 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
808 #training e test set senza variabile target
809 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
810 #anche sui dati di test
811 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
812
813 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
814 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
815 levels(training_set$Two_yr_Recidivism)
816
817 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
818 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
819 levels(test_set$Two_yr_Recidivism)
820 #####
821 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (sufficiente salvarlo e ricaricarlo)
822 logitMod <- glm(Two_yr_Recidivism ~ ., data = training_set, family=binomial(link="logit"))
823 logitMod #capire cosa significano questi dati
824
825 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response, mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile fare la predizione
826 balanced2_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria fairness
827 classified_dt2 = cbind(test_set_data, pred)
828 #classified_dt = rbind(classified_dt1, classified_dt2)
829 pred1 <- prediction(pred, test_set$Two_yr_Recidivism) #pred1 un oggetto pred.obj di ROCR
830 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
```

```
831 optimized_cutoff2 = opt.cut(perf, pred1)#il cutoff ottimale leggermente piu basso di quello
standard, che sarebbe invece 0.5
832
833 #ripristino di nuovo i livelli yes/no per renderlo piu leggibile
834 levels(balanced2_classified_dt$Two_yr_Recidivism)[levels(balanced2_classified_dt$Two_yr_
Recidivism)==1] <- "Yes_recidivism"
835 levels(balanced2_classified_dt$Two_yr_Recidivism)[levels(balanced2_classified_dt$Two_yr_
Recidivism)==0] <- "No_recidivism"
836
837 #####
838
839
840 ## -----
841 #balanced_data3
842
843 #sostituisco i livelli con valori numerici
844 levels(balanced_data3$Two_yr_Recidivism)[levels(balanced_data3$Two_yr_Recidivism)=="yes"] <-
     1
845 levels(balanced_data3$Two_yr_Recidivism)[levels(balanced_data3$Two_yr_Recidivism)=="no"] <- 0
846 #rimuovo da balanced_data3 le colonne age_above e age_below
847 cols_to_drop <- names(balanced_data3) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
848 balanced_data3 <- balanced_data3[, !cols_to_drop, drop = FALSE]
849
850 #split into train and test set
851 sample_size = floor(0.7*nrow(balanced_data3))
852 #set a random seed per rendere l'esperimento ripetibile
853 set.seed(42)
854
855 # randomly split data
856 picked = sample(seq_len(nrow(balanced_data3)), size = sample_size)
857 training_set = balanced_data3[picked,]
858 test_set = balanced_data3[-picked,]
859
860 #as.factor mi serve per fare una classificazione binaria
861 balanced_data3$Two_yr_Recidivism = as.factor(balanced_data3$Two_yr_Recidivism)
862 #devo anche eliminare recidivismo dal training_set
863 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
864 #training e test set senza variabile target
865 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
866 #anche sui dati di test
867 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
868
869 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_
recidivism"] <- 1
870 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism
     "] <- 0
871 levels(training_set$Two_yr_Recidivism)
```

```

873 levels(test_set$Two_yr_Rcidivism)[levels(test_set$Two_yr_Rcidivism)=="Yes_recidivism"] <- 1
874 levels(test_set$Two_yr_Rcidivism)[levels(test_set$Two_yr_Rcidivism)=="No_recidivism"] <- 0
875 levels(test_set$Two_yr_Rcidivism)
876 #####
877 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
878 logitMod <- glm(Two_yr_Rcidivism ~ ., data = training_set, family=binomial(link="logit"))
879 logitMod #capire cosa significano questi dati
880
881 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
882 balanced3_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
# fairness
883 classified_dt2 = cbind(test_set_data, pred)
884 #classified_dt = rbind(classified_dt1, classified_dt2)
885 pred1 <- prediction(pred, test_set$Two_yr_Rcidivism) #pred1 un oggetto pred.obj di ROCR
886 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
887 optimized_cutoff3 = opt.cut(perf, pred1)#il cutoff ottimale leggermente pi basso di quello
# standard, che sarebbe invece 0.5
888
889 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
890 levels(balanced3_classified_dt$Two_yr_Rcidivism)[levels(balanced3_classified_dt$Two_yr_
# Recidivism)==1] <- "Yes_recidivism"
891 levels(balanced3_classified_dt$Two_yr_Rcidivism)[levels(balanced3_classified_dt$Two_yr_
# Recidivism)==0] <- "No_recidivism"
892 #####
893
894
895 ## -----
896 #balanced_data4
897
898 #sostituisco i livelli con valori numerici
899 levels(balanced_data4$Two_yr_Rcidivism)[levels(balanced_data4$Two_yr_Rcidivism)=="yes"] <-
# 1
900 levels(balanced_data4$Two_yr_Rcidivism)[levels(balanced_data4$Two_yr_Rcidivism)=="no"] <- 0
901 #rimuovo da balanced_data4 le colonne age_above e age_below
902 cols_to_drop <- names(balanced_data4) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
903 balanced_data4 <- balanced_data4[, !cols_to_drop, drop = FALSE]
904
905 #split into train and test set
906 sample_size = floor(0.7*nrow(balanced_data4))
907 #set a random seed per rendere l'esperimento ripetibile
908 set.seed(42)
909
910 # randomly split data
911 picked = sample(seq_len(nrow(balanced_data4)), size = sample_size)

```

```

912 training_set =balanced_data4[picked,]
913 test_set =balanced_data4[-picked,]
914
915 #as.factor mi serve per fare una classificazione binaria
916 balanced_data4$Two_yr_Recidivism = as.factor(balanced_data4$Two_yr_Recidivism)
917 #devo anche eliminare recidivismo dal training_set
918 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
919 #training e test set senza variabile target
920 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
921 #anche sui dati di test
922 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
923
924 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)== "Yes_
    recidivism"] <- 1
925 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)== "No_recidivism
    "] <- 0
926 levels(training_set$Two_yr_Recidivism)
927
928 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)== "Yes_recidivism"] <- 1
929 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)== "No_recidivism"] <- 0
930 levels(test_set$Two_yr_Recidivism)
931 ##########
932 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
    sufficiente salvarlo e ricaricarlo)
933 logitMod <- glm(Two_yr_Recidivism ~ ., data = training_set, family=binomial(link="logit"))
934 logitMod #capire cosa significano questi dati
935
936 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
    mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
    tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
    fare la predizione
937 balanced4_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
    fairness
938 classified_dt2 = cbind(test_set_data, pred)
939 #classified_dt = rbind(classified_dt1, classified_dt2)
940 pred1 <- prediction(pred, test_set$Two_yr_Recidivism) #pred1 un oggetto pred.obj di ROCR
941 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
942 optimized_cutoff4 = opt.cut(perf, pred1)#il cutoff ottimale leggermente pii basso di quello
    standard, che sarebbe invece 0.5
943
944 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
945 levels(balanced4_classified_dt$Two_yr_Recidivism)[levels(balanced4_classified_dt$Two_yr_
    Recidivism)==1] <- "Yes_recidivism"
946 levels(balanced4_classified_dt$Two_yr_Recidivism)[levels(balanced4_classified_dt$Two_yr_
    Recidivism)==0] <- "No_recidivism"
947
948
949 ## -----

```

```

950 #####
951
952 #balanced_data5
953
954 #sostituisco i livelli con valori numerici
955 levels(balanced_data5$Two_yr_Recidivism)[levels(balanced_data5$Two_yr_Recidivism)=="yes"] <-
956   1
956 levels(balanced_data5$Two_yr_Recidivism)[levels(balanced_data5$Two_yr_Recidivism)=="no"] <- 0
957 #rimuovo da balanced_data5 le colonne age_above e age_below
958 cols_to_drop <- names(balanced_data5) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
959 balanced_data5 <- balanced_data5[, !cols_to_drop, drop = FALSE]
960
961 #split into train and test set
962 sample_size = floor(0.7*nrow(balanced_data5))
963 #set a random seed per rendere l'esperimento ripetibile
964 set.seed(42)
965
966 # randomly split data
967 picked = sample(seq_len(nrow(balanced_data5)), size = sample_size)
968 training_set = balanced_data5[picked,]
969 test_set = balanced_data5[-picked,]
970
971 #as.factor mi serve per fare una classificazione binaria
972 balanced_data5$Two_yr_Recidivism = as.factor(balanced_data5$Two_yr_Recidivism)
973 #devo anche eliminare recidivismo dal training_set
974 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
975 #training e test set senza variabile target
976 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
977 #anche sui dati di test
978 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
979
980 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_"
981   recidivism"] <- 1
981 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism"
982   "] <- 0
982 levels(training_set$Two_yr_Recidivism)
983
984 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
985 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
986 levels(test_set$Two_yr_Recidivism)
987 #####
988 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
989   #sufficiente salvarlo e ricaricarlo)
989 logitMod <- glm(Two_yr_Recidivism ~ ., data = training_set, family=binomial(link="logit"))
990 logitMod #capire cosa significano questi dati
991
992 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
992   #mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo

```

```

    tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
    fare la predizione
993 balanced5_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
    fairness
994 classified_dt2 = cbind(test_set_data, pred)
995 #classified_dt = rbind(classified_dt1, classified_dt2)
996 pred1 <- prediction(pred, test_set$Two_yr_Recidivism) #pred1 un oggetto pred.obj di ROCR
997 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
998 optimized_cutoff5 = opt.cut(perf, pred1)#il cutoff ottimale leggermente piii basso di quello
    standard, che sarebbe invece 0.5
999
1000 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
1001 levels(balanced5_classified_dt$Two_yr_Recidivism)[levels(balanced5_classified_dt$Two_yr_
    Recidivism)==1] <- "Yes_recidivism"
1002 levels(balanced5_classified_dt$Two_yr_Recidivism)[levels(balanced5_classified_dt$Two_yr_
    Recidivism)==0] <- "No_recidivism"
1003
1004 ## -----
1005 ##########
1006
1007 #balanced_data6
1008
1009 #sostituisco i livelli con valori numerici
1010 levels(balanced_data6$Two_yr_Recidivism)[levels(balanced_data6$Two_yr_Recidivism)=="yes"] <-
    1
1011 levels(balanced_data6$Two_yr_Recidivism)[levels(balanced_data6$Two_yr_Recidivism)=="no"] <- 0
1012 #rimuovo da balanced_data6 le colonne age_above e age_below
1013 cols_to_drop <- names(balanced_data6) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
1014 balanced_data6 <- balanced_data6[, !cols_to_drop, drop = FALSE]
1015
1016 #split into train and test set
1017 sample_size = floor(0.7*nrow(balanced_data6))
1018 #set a random seed per rendere l'esperimento ripetibile
1019 set.seed(42)
1020
1021 # randomly split data
1022 picked = sample(seq_len(nrow(balanced_data6)), size = sample_size)
1023 training_set = balanced_data6[picked,]
1024 test_set = balanced_data6[-picked,]
1025
1026 #as.factor mi serve per fare una classificazione binaria
1027 balanced_data6$Two_yr_Recidivism = as.factor(balanced_data6$Two_yr_Recidivism)
1028 #devo anche eliminare recidivismo dal training_set
1029 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
1030 #training e test set senza variabile target
1031 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
1032 #anche sui dati di test
1033 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]

```

```

1034
1035 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_
    recidivism"] <- 1
1036 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism
    "] <- 0
1037 levels(training_set$Two_yr_Recidivism)
1038
1039 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
1040 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
1041 levels(test_set$Two_yr_Recidivism)
1042 #####
1043 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
    sufficiente salvarlo e ricaricarlo)
1044 logitMod <- glm(Two_yr_Recidivism ~ ., data = training_set, family=binomial(link="logit"))
1045 logitMod #capire cosa significano questi dati
1046
1047 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
    mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
    tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
    fare la predizione
1048 balanced6_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
    fairness
1049 classified_dt2 = cbind(test_set_data, pred)
1050 #classified_dt = rbind(classified_dt1, classified_dt2)
1051 pred1 <- prediction(pred, test_set$Two_yr_Recidivism) #pred1 un oggetto pred.obj di ROCRR
1052 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
1053 optimized_cutoff6 = opt.cut(perf, pred1)#il cutoff ottimale leggermente ppi basso di quello
    standard, che sarebbe invece 0.5
1054
1055 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
1056 levels(balanced6_classified_dt$Two_yr_Recidivism)[levels(balanced6_classified_dt$Two_yr_
    Recidivism)==1] <- "Yes_recidivism"
1057 levels(balanced6_classified_dt$Two_yr_Recidivism)[levels(balanced6_classified_dt$Two_yr_
    Recidivism)==0] <- "No_recidivism"
1058
1059
1060
1061 ## -----
1062 #####
1063
1064 #balanced_data7
1065
1066 #sostituisco i livelli con valori numerici
1067 levels(balanced_data7$Two_yr_Recidivism)[levels(balanced_data7$Two_yr_Recidivism)=="yes"] <-
    1
1068 levels(balanced_data7$Two_yr_Recidivism)[levels(balanced_data7$Two_yr_Recidivism)=="no"] <- 0
1069 #rimuovo da balanced_data7 le colonne age_above e age_below
1070 cols_to_drop <- names(balanced_data7) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")

```

```

1071 balanced_data7 <- balanced_data7[, !cols_to_drop, drop = FALSE]
1072
1073 #split into train and test set
1074 sample_size = floor(0.7*nrow(balanced_data7))
1075 #set a random seed per rendere l'esperimento ripetibile
1076 set.seed(42)
1077
1078 # randomly split data
1079 picked = sample(seq_len(nrow(balanced_data7)), size = sample_size)
1080 training_set = balanced_data7[picked,]
1081 test_set = balanced_data7[-picked,]
1082
1083 #as.factor mi serve per fare una classificazione binaria
1084 balanced_data7$Two_yr_Recidivism = as.factor(balanced_data7$Two_yr_Recidivism)
1085 #devo anche eliminare recidivismo dal training_set
1086 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
1087 #training e test set senza variabile target
1088 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
1089 #anche sui dati di test
1090 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
1091
1092 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
1093 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
1094 levels(training_set$Two_yr_Recidivism)
1095
1096 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
1097 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
1098 levels(test_set$Two_yr_Recidivism)
1099 ######
1100 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
1101 logitMod <- glm(Two_yr_Recidivism ~ ., data = training_set, family=binomial(link="logit"))
1102 logitMod #capire cosa significano questi dati
1103
1104 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
1105 balanced7_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
# fairness
1106 classified_dt2 = cbind(test_set_data, pred)
1107 #classified_dt = rbind(classified_dt1, classified_dt2)
1108 pred1 <- prediction(pred, test_set$Two_yr_Recidivism) #pred1 un oggetto pred.obj di ROCR
1109 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
1110 optimized_cutoff7 = opt.cut(perf, pred1)#il cutoff ottimale leggermente pi basso di quello
# standard, che sarebbe invece 0.5

```

```

1111
1112 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
1113 levels(balanced7_classified_dt$Two_yr_Recidivism)[levels(balanced7_classified_dt$Two_yr_
1114 Recidivism)==1] <- "Yes_recidivism"
1115 levels(balanced7_classified_dt$Two_yr_Recidivism)[levels(balanced7_classified_dt$Two_yr_
1116 Recidivism)==0] <- "No_recidivism"
1117
1118 ## -----
1119 #ora confronto ciascun dataset bilanciato con quello orignale
1120 #poi seleziono quelli che sembrano effettivamente migliori del dataset originale
1121 #poi confronto tra loro questi dataset bilanciati rimanenti e scelgo il migliore
1122
1123 #balanced_data0: bilanciato per Two_yr_recidivism
1124 #balanced_data1: bilanciato per Female
1125 #balanced_data2: bilanciato per ethnicity
1126 #balanced_data3: bilanciato per Age
1127 #balanced_data4: bilanciato per Two_yr_recidivism|ethnicity
1128
1129 #creo anche un dataset con i valori delle mean deviation per ogni dataset bilanciato, per
1130 #ogni metrica
1131 msd_balanced=NULL
1132
1133 source("util.R")
1134 png("COMPASb0rdComp.png")
1135 p = fair_metrics_cmp_graph(data1      = classified_dt,
1136                             data2 = balanced0_classified_dt,
1137                             outcome     = 'Two_yr_Recidivism',
1138                             outcome_base = 'No_recidivism',
1139                             group       = 'ethnicity',
1140                             probs1     = 'pred',
1141                             probs2     = 'pred',
1142                             preds_levels = c('No_recidivism', 'Yes_recidivism'),
1143                             cutoff1    = optimized_cutoff,
1144                             cutoff2    = optimized_cutoff0,
1145                             base       = 'Caucasian')
1146 dev.off()
1147 #praticamente solo peggioramenti
1148 msd_balanced=p[[1]]
1149 msd_balanced= rbind(msd_balanced, p[[2]])
1150
1151 png("COMPASbirdComp.png")
1152 p = fair_metrics_cmp_graph(data1      = classified_dt,
1153                             data2 = balanced1_classified_dt,
1154                             outcome     = 'Two_yr_Recidivism',
1155                             outcome_base = 'No_recidivism',
1156                             group       = 'ethnicity',

```

```

1156         probs1 =      'pred',
1157         probs2 =      'pred',
1158         preds_levels = c('No_recidivism','Yes_recidivism'),
1159         cutoff1       = optimized_cutoff,
1160         cutoff2       = optimized_cutoff,
1161         base          = 'Caucasian')

1162 dev.off()
1163 msd_balanced= rbind(msd_balanced, p[[2]])
1164 #in questo caso abbiamo dei miglioramenti abbastanza notevoli: dal momento che questo sembra
    il miglioramento pi notevole per le metriche di fairness, proviamo a migliorarlo
    ulteriormente mettendo lo stesso numero di individui per ogni etnia tra i maschi e tra le
    femmine, quindi avremo un doppio bilanciamento (lo chiamo balanced6_classified_dt)
1165
1166 png("COMPASb2rdComp.png")
1167 p = fair_metrics_cmp_graph(data1      = classified_dt,
1168                             data2 = balanced2_classified_dt,
1169                             outcome      = 'Two_yr_Recidivism',
1170                             outcome_base = 'No_recidivism',
1171                             group        = 'ethnicity',
1172                             probs1 =      'pred',
1173                             probs2 =      'pred',
1174                             preds_levels = c('No_recidivism','Yes_recidivism'),
1175                             cutoff1       = optimized_cutoff,
1176                             cutoff2       = optimized_cutoff2,
1177                             base          = 'Caucasian')
1178 dev.off()
1179 msd_balanced= rbind(msd_balanced, p[[2]])
1180 #questo era il dataset bilanciato per ethnicity: abbiamo qualche miglioramento, ma anche
    qualche peggioramento
1181
1182 png("COMPASb3rdComp.png")
1183 p = fair_metrics_cmp_graph(data1      = classified_dt,
1184                             data2 = balanced3_classified_dt,
1185                             outcome      = 'Two_yr_Recidivism',
1186                             outcome_base = 'No_recidivism',
1187                             group        = 'ethnicity',
1188                             probs1 =      'pred',
1189                             probs2 =      'pred',
1190                             preds_levels = c('No_recidivism','Yes_recidivism'),
1191                             cutoff1       = optimized_cutoff,
1192                             cutoff2       = optimized_cutoff3,
1193                             base          = 'Caucasian')
1194 dev.off()
1195 msd_balanced= rbind(msd_balanced, p[[2]])
1196 #qualche miglioramento e peggioramento
1197
1198 png("COMPASb4rdComp.png")
1199 p = fair_metrics_cmp_graph(data1      = classified_dt,

```

```

1200      data2 = balanced4_classified_dt,
1201      outcome      = 'Two_yr_Recidivism',
1202      outcome_base = 'No_recidivism',
1203      group        = 'ethnicity',
1204      probs1       = 'pred',
1205      probs2       = 'pred',
1206      preds_levels = c('No_recidivism', 'Yes_recidivism'),
1207      cutoff1      = optimized_cutoff,
1208      cutoff2      = optimized_cutoff4,
1209      base         = 'Caucasian')

1210 dev.off()
1211 msd_balanced= rbind(msd_balanced, p[[2]])
1212 #in questo caso abbiamo ponderato sia l'etnia sia il recidivismo: stesso numero di individui
# per ciascuna etnia e, all'interno della singola etnia, stesso numero di recidivi e non
# recidivi
1213
1214 #####
1215 #balanced balanced5_classified_dt
1216 #re avessimo invece ponderato il recidivismo in funzione dell'etnia, cio ugual numero di
# recidivi per ogni etnia, ma con l'attributo etnia non equilibrato? proviamo (vedi chunck
# precedente e poi qui sotto)
1217
1218 png("COMPASb5rdComp.png")
1219 p = fair_metrics_cmp_graph(data1      = classified_dt,
1220                             data2 = balanced5_classified_dt,
1221                             outcome      = 'Two_yr_Recidivism',
1222                             outcome_base = 'No_recidivism',
1223                             group        = 'ethnicity',
1224                             probs1       = 'pred',
1225                             probs2       = 'pred',
1226                             preds_levels = c('No_recidivism', 'Yes_recidivism'),
1227                             cutoff1      = optimized_cutoff,
1228                             cutoff2      = optimized_cutoff5,
1229                             base         = 'Caucasian')
1230 dev.off()
1231 msd_balanced= rbind(msd_balanced, p[[2]])
1232 #alcune metriche sono migliori, ma altre decisamente peggiori
1233
1234 #####
1235 #balanced6_classified_dt
1236
1237 png("COMPASb6b1Comp.png")
1238 #NOTARE CHE QUESTA NON UNA COMPARISON RISPETTO AL DATASET ORIGINALE
1239 fair_metrics_cmp_graph(data1      = balanced1_classified_dt,
1240                           data2 = balanced6_classified_dt,
1241                           outcome      = 'Two_yr_Recidivism',
1242                           outcome_base = 'No_recidivism',
1243                           group        = 'ethnicity',

```

```
1244     probs1 =      'pred',
1245     probs2 =      'pred',
1246     preds_levels = c('No_recidivism', 'Yes_recidivism'),
1247     cutoff1       = optimized_cutoff1,
1248     cutoff2       = optimized_cutoff6,
1249     base          = 'Caucasian')
1250 dev.off()
1251
1252 png("COMPASb6rdComp.png")
1253 p = fair_metrics_cmp_graph(data1      = classified_dt,
1254                             data2 = balanced6_classified_dt,
1255                             outcome      = 'Two_yr_Recidivism',
1256                             outcome_base = 'No_recidivism',
1257                             group        = 'ethnicity',
1258                             probs1 =      'pred',
1259                             probs2 =      'pred',
1260                             preds_levels = c('No_recidivism', 'Yes_recidivism'),
1261                             cutoff1       = optimized_cutoff1,
1262                             cutoff2       = optimized_cutoff6,
1263                             base          = 'Caucasian')
1264 dev.off()
1265 msd_balanced= rbind(msd_balanced, p[[2]])
1266 #a quanto pare il balanced6_classified_dt NON meglio del #1
1267
1268 png("COMPASb7rdComp.png")
1269 p = fair_metrics_cmp_graph(data1      = classified_dt,
1270                             data2 = balanced7_classified_dt,
1271                             outcome      = 'Two_yr_Recidivism',
1272                             outcome_base = 'No_recidivism',
1273                             group        = 'ethnicity',
1274                             probs1 =      'pred',
1275                             probs2 =      'pred',
1276                             preds_levels = c('No_recidivism', 'Yes_recidivism'),
1277                             cutoff1       = optimized_cutoff1,
1278                             cutoff2       = optimized_cutoff7,
1279                             base          = 'Caucasian')
1280 dev.off()
1281 msd_balanced= rbind(msd_balanced, p[[2]])
1282
1283 #CONFRONTO TRA LORO BALANCED DATASETS
1284 #####
1285 png("COMPASb2b4Comp.png")
1286 fair_metrics_cmp_graph(data1      = balanced4_classified_dt,
1287                         data2 = balanced2_classified_dt,
1288                         outcome      = 'Two_yr_Recidivism',
1289                         outcome_base = 'No_recidivism',
1290                         group        = 'ethnicity',
1291                         probs1 =      'pred',
```

```

1292         probs2 =      'pred',
1293         preds_levels = c('No_recidivism', 'Yes_recidivism'),
1294         cutoff1      = optimized_cutoff4,
1295         cutoff2      = optimized_cutoff2,
1296         base        = 'Caucasian')
1297 dev.off()
1298
1299 png("COMPASb7b6Comp.png")
1300 fair_metrics_cmp_graph(data1      = balanced7_classified_dt,
1301                         data2 = balanced6_classified_dt,
1302                         outcome      = 'Two_yr_Recidivism',
1303                         outcome_base = 'No_recidivism',
1304                         group       = 'ethnicity',
1305                         probs1 =      'pred',
1306                         probs2 =      'pred',
1307                         preds_levels = c('No_recidivism', 'Yes_recidivism'),
1308                         cutoff1      = optimized_cutoff6,
1309                         cutoff2      = optimized_cutoff7,
1310                         base        = 'Caucasian')
1311 dev.off()
1312 fair_metrics_cmp_graph(data1      = balanced7_classified_dt,
1313                         data2 = balanced6_classified_dt,
1314                         outcome      = 'Two_yr_Recidivism',
1315                         outcome_base = 'No_recidivism',
1316                         group       = 'ethnicity',
1317                         probs1 =      'pred',
1318                         probs2 =      'pred',
1319                         preds_levels = c('No_recidivism', 'Yes_recidivism'),
1320                         cutoff1      = optimized_cutoff7,
1321                         cutoff2      = optimized_cutoff6,
1322                         base        = 'Caucasian')
1323 #dev.off()
1324
1325 msd_balanced
1326 long_msd_balanced <- msd_balanced %>% gather(metrics, values, dem_parity:spec_parity)
1327 xlab("")
1328
1329 g = ggplot(long_msd_balanced, aes(fill=metrics, y=values, x=datasets)) +
1330   geom_bar(position="dodge", stat="identity") +
1331   #ggtitle("Studying 4 species..") +
1332   facet_wrap(~metrics, ncol=2) +
1333   #facet_grid(metrics ~ datasets)
1334   theme(legend.position="none",
1335         strip.text = element_text(face="bold", size=15, lineheight=5.0),
1336         strip.background = element_rect(fill="lightblue", colour="black",
1337                                         size=1,), axis.text.x = element_text(angle = 90),
1338         axis.text=element_text(size=20, face="bold"),
1339         axis.title=element_text(size=20, face="bold")) +

```

```

1340     xlab(""))
1341 ggsave("MetricsValuesForbd.jpg", plot = g, width = 24, height = 33, units = "in")
1342
1343
1344 ## -----
1345 #USO DI AIF360
1346 library(aif360)
1347 #load_aif360_lib()
1348
1349
1350 ## -----
1351 ######
1352 #converto dataset in binario
1353 #qua devo usare il balanced non ancora classificato, NON quello classificato!!!
1354 #OCCHIO: fai il run di queste righe di conversione 0-1 solo se il dataset non ancora stato
    convertito!!! ad esempio fai prima un check is.numeric
1355
1356 levels(balanced_data1$Two_yr_Rcidivism)[levels(balanced_data1$Two_yr_Rcidivism)=="Yes_"
    recidivism"] <- 1
1357 levels(balanced_data1$Two_yr_Rcidivism)[levels(balanced_data1$Two_yr_Rcidivism)=="No_"
    recidivism"] <- 0
1358 #prendo caucasian come riferimento perch il confronto tra le metrich di fairness lo far poi
    con caucasian come base
1359 #I caucasian sono codificati con 1 (privilegiati)
1360 levels(balanced_data1$ethnicity)[levels(balanced_data1$ethnicity)!="Caucasian"] <- 0
1361 levels(balanced_data1$ethnicity)[levels(balanced_data1$ethnicity)=="Caucasian"] <- 1
1362 levels(balanced_data1$Female)[levels(balanced_data1$Female)=="Female"] <- 0
1363 levels(balanced_data1$Female)[levels(balanced_data1$Female)=="Male"] <- 1
1364 levels(balanced_data1$Misdemeanor)[levels(balanced_data1$Misdemeanor)=="yes"] <- 0
1365 levels(balanced_data1$Misdemeanor)[levels(balanced_data1$Misdemeanor)=="no"] <- 1
1366 levels(balanced_data1$Age)[levels(balanced_data1$Age)!="between25_45"] <- 0
1367 levels(balanced_data1$Age)[levels(balanced_data1$Age)=="between25_45"] <- 1
1368 # #trasformo colonne categoriche in numeriche
1369 # must_convert<-sapply(balanced_data1,is.factor)      # logical vector telling if a variable
    needs to be displayed as numeric
1370 # balanced_data1<-sapply(balanced_data1[,must_convert],unclass)    # data.frame of all
    categorical variables now displayed as numeric
1371
1372
1373 ## -----
1374 #i favoriti sono gli adulti-1
1375 formatted_dataset <- aif360::aif_dataset(data_path = balanced_data1, favor_label = 0,
1376 unfavor_label = 1, #label favorita per la variabile target: 0 corrisponde a good
1377 unprivileged_protected_attribute = 0, # 0-young-unprivileged
1378 privileged_protected_attribute = 1, #se non lo scrivo, per il valore privilegiato viene
    considerato il valore pi alto
1379 target_column = "Two_yr_Rcidivism", #qua prima avevo pred.rescaled
1380 protected_attribute = "ethnicity")

```

```
1381 formatted_dataset
1382
1383 #tecnica di debias preprocessing
1384 di <- disparate_impact_remover(repair_level = 0.8, sensitive_attribute = "ethnicity")
1385 rp <- di$fit_transform(formatted_dataset)
1386
1387 #reweighing (tecnica preprocessing)
1388 pri <- list("ethnicity", 1)
1389 unpri <- list("ethnicity", 0)
1390 rw <- reweighing(unpri,pri)
1391 #rw$fit(formatted_dataset)
1392 #ad_transformed <- rw$transform(formatted_dataset)
1393 ad_fit_transformed <- rw$fit_transform(rp) #questo lo faccio su rp, cos analizzo entrambe le
      tecniche in una volta sola (altrimenti dovrei fare mille classificazioni e mille casi)
1394
1395
1396 ## -----
1397 #ora torniamo a un dataset in formato data frame (non aif360)
1398 col_names <- c(ad_fit_transformed$feature_names, "Two_yr_Recidivism")
1399 col_names
1400 new_data <- data.frame(ad_fit_transformed$features, ad_fit_transformed$labels)
1401 colnames(new_data) <- col_names
1402
1403
1404
1405 #quindi new_data un dataset su cui ho fatto rebalancing per Female, disparate impact remover
      e reweighing
1406
1407
1408 ## -----
1409 #sostituisco i livelli con valori numerici
1410 levels(new_data$Two_yr_Recidivism)[levels(new_data$Two_yr_Recidivism)=="yes"] <- 1
1411 levels(new_data$Two_yr_Recidivism)[levels(new_data$Two_yr_Recidivism)=="no"] <- 0
1412 #rimuovo da new_data le colonne age_above e age_below
1413 cols_to_drop <- names(new_data) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
1414 new_data <- new_data[, !cols_to_drop, drop = FALSE]
1415
1416 #split into train and test set
1417 sample_size = floor(0.7*nrow(new_data))
1418 #set a random seed per rendere l'esperimento ripetibile
1419 set.seed(42)
1420
1421 # randomly split data
1422 picked = sample(seq_len(nrow(new_data)), size = sample_size)
1423 training_set = new_data[picked,]
1424 test_set = new_data[-picked,]
1425
1426 #as.factor mi serve per fare una classificazione binaria
```

```

1427 new_data$Two_yr_Rcidivism = as.factor(new_data$Two_yr_Rcidivism)
1428 #devo anche eliminare recidivismo dal training_set
1429 cols_to_drop <- names(training_set) %in% c("Two_yr_Rcidivism")
1430 #training e test set senza variabile target
1431 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
1432 #anche sui dati di test
1433 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
1434
1435 levels(training_set$Two_yr_Rcidivism)[levels(training_set$Two_yr_Rcidivism)=="Yes_recidivism"] <- 1
1436 levels(training_set$Two_yr_Rcidivism)[levels(training_set$Two_yr_Rcidivism)=="No_recidivism"] <- 0
1437 levels(training_set$Two_yr_Rcidivism)
1438
1439 levels(test_set$Two_yr_Rcidivism)[levels(test_set$Two_yr_Rcidivism)=="Yes_recidivism"] <- 1
1440 levels(test_set$Two_yr_Rcidivism)[levels(test_set$Two_yr_Rcidivism)=="No_recidivism"] <- 0
1441 levels(test_set$Two_yr_Rcidivism)
1442 ##########
1443 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
1444 logitMod <- glm(Two_yr_Rcidivism ~ ., data = training_set, family=binomial(link="logit"))
1445 logitMod #capire cosa significano questi dati
1446
1447 pred = predict(logitMod, test_set_data, type = "response")
1448 new_data_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
# fairness
1449 classified_dt2 = cbind(test_set_data, pred)
1450 #classified_dt = rbind(classified_dt1, classified_dt2)
1451 optimized_cutoff_new = opt.cut(perf, pred1)#il cutoff ottimale leggermente pi basso di
# quello standard, che sarebbe invece 0.5
1452
1453 ##
# -----
1454 #ripristino di nuovo TUTTI i livelli categorici per renderlo pi leggibile
1455 new_data_classified_dt$Two_yr_Rcidivism = ifelse(new_data_classified_dt$Two_yr_Rcidivism
==1,"Yes_recidivism", "No_recidivism")
1456
1457 new_data_classified_dt$ethnicity = ifelse(new_data_classified_dt$ethnicity==1,"Caucasian", "Not_caucasian")
1458
1459
1460 new_data_classified_dt$Female = ifelse(new_data_classified_dt$Female==1,"Male", "Female")
1461
1462
1463 new_data_classified_dt$Misdemeanor = ifelse(new_data_classified_dt$Misdemeanor==1,"no", "yes")
1464
1465
1466

```

```

1467 new_data_classified_dt$Age = ifelse(new_data_classified_dt$Age==1,"between25_45", "Not_
1468 between25_45")
1469
1470
1471 ## -----
1472 #confronto con il dataset bilanciato su female
1473 fair_metrics_cmp_graph(data1      = balanced1_classified_dt,
1474                         data2 = new_data_classified_dt,
1475                         outcome      = 'Two_yr_Recidivism',
1476                         outcome_base = 'No_recidivism',
1477                         group        = 'ethnicity',
1478                         probs1       = 'pred',
1479                         probs2       = 'pred',
1480                         preds_levels = c('No_recidivism','Yes_recidivism'),
1481                         cutoff1      = optimized_cutoff1,
1482                         cutoff2      = optimized_cutoff_new,
1483                         base         = 'Caucasian')
1484
1485 #confronto con il dataset originale
1486 fair_metrics_cmp_graph(data1      = classified_dt,
1487                         data2 = new_data_classified_dt,
1488                         outcome      = 'Two_yr_Recidivism',
1489                         outcome_base = 'No_recidivism',
1490                         group        = 'ethnicity',
1491                         probs1       = 'pred',
1492                         probs2       = 'pred',
1493                         preds_levels = c('No_recidivism','Yes_recidivism'),
1494                         cutoff1      = optimized_cutoff,
1495                         cutoff2      = optimized_cutoff_new,
1496                         base         = 'Caucasian')
1497
1498
1499 ## -----
1500 #convertendo il dataset in binario
1501 levels(new_data$Two_yr_Recidivism)[levels(new_data$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
1502 levels(new_data$Two_yr_Recidivism)[levels(new_data$Two_yr_Recidivism)=="No_recidivism"] <- 0
1503 #prendo caucasian come riferimento perch il confronto tra le metrich di fairness lo far poi
1504     con caucasian come base
1505 #I caucasian sono codificati con 1 (privilegiati)
1506 levels(new_data$ethnicity)[levels(new_data$ethnicity)!="Caucasian"] <- 0
1507 levels(new_data$ethnicity)[levels(new_data$ethnicity)=="Caucasian"] <- 1
1508 levels(new_data$Female)[levels(new_data$Female)=="Female"] <- 0
1509 levels(new_data$Female)[levels(new_data$Female)=="Male"] <- 1
1510 levels(new_data$Misdemeanor)[levels(new_data$Misdemeanor)=="yes"] <- 0
1511 levels(new_data$Misdemeanor)[levels(new_data$Misdemeanor)=="no"] <- 1
1512 levels(new_data$Age)[levels(new_data$Age)!="between25_45"] <- 0
1513 levels(new_data$Age)[levels(new_data$Age)=="between25_45"] <- 1

```

```

1513 # #trasformo colonne categoriche in numeriche
1514 # must_convert<-sapply(new_data,is.factor)           # logical vector telling if a variable needs
   to be displayed as numeric
1515 # new_data<-sapply(new_data[,must_convert],unclass)    # data.frame of all categorical
   variables now displayed as numeric
1516
1517
1518 ## -----
1519 #creo dataset formattato per aif360
1520 #NOTA BENE: formatted_dataset la versione di raw_data per aif360. formatted_dataset sar
   SEMPRE quello che uso per il fit dei metodi di debias e far la predizione sul formatted_
   dataset di new_data
1521 #creo quindi il formatted dataset di new_data e lo chiamo
1522 formatted_dataset_forPred <- aif360::aif_dataset(data_path = new_data, favor_label = 0,
1523 unfavor_label = 1, #label favorita per la variabile target: 0 corrisponde a good
1524 unprivileged_protected_attribute = 0, # 0-young-unprivileged
1525 privileged_protected_attribute = 1, #se non lo scrivo, per il valore privilegiato viene
   considerato il valore pi alto
1526 target_column = "Two_yr_Recidivism", #qua prima avevo pred.rescaled
1527 protected_attribute = "ethnicity")
1528
1529
1530 ## -----
1531 #applico tecniche di debias inprocessing: adversarial debiasing and prejudice remover
1532 #prejudice remover
1533 model <- prejudice_remover(class_attr = "Two_yr_Recidivism", sensitive_attr = "ethnicity")
1534 model$fit(formatted_dataset_forPred)
1535 formatted_dataset_pr <- model$predict(formatted_dataset_forPred) #FUNZIONA ma devi fare
   questo come prima tecnica in assoluto
1536 #in formatted_dataset_pr ho quindi IL NUOVO DATASET FORMATTATO con le predizioni del
   recidivismo fatto con prejudice remover
1537
1538
1539
1540 ## -----
1541 #ora torniamo a un dataset in formato data frame (non aif360)
1542 col_names <- c(formatted_dataset_pr$feature_names, "preds")#questa colonna contiene le
   predizioni sul recidivismo appena fatte dall'algoritmo
1543 col_names
1544 new_data_pr <- data.frame(formatted_dataset_pr$features, formatted_dataset_pr$labels)
1545 colnames(new_data_pr) <- col_names
1546 #new_data_pr in questo caso gi classificato!!!
1547
1548
1549 ## -----
1550 new_data_pr$Two_yr_Recidivism = new_data$Two_yr_Recidivism
1551
1552

```

```

1553 ## -----
1554 #ripristino di nuovo TUTTI i livelli categorici per renderlo pi leggibile
1555 new_data_pr$Two_yr_Recidivism = ifelse(new_data_pr$Two_yr_Recidivism==1,"Yes_recidivism", "No_recidivism")
1556
1557 new_data_pr$ethnicity = ifelse(new_data_pr$ethnicity==1,"Caucasian", "Not_caucasian")
1558
1559
1560 new_data_pr$Female = ifelse(new_data_pr$Female==1,"Male", "Female")
1561
1562
1563 new_data_pr$Misdemeanor = ifelse(new_data_pr$Misdemeanor==1,"no", "yes")
1564
1565
1566 new_data_pr$Age = ifelse(new_data_pr$Age==1,"between25_45", "Not_between25_45")
1567
1568
1569
1570 ## -----
1571 #confronto le metriche con quelle di
1572 #new_data_pr
1573 #source("util.R")
1574
1575 #NOTA BENE: in questo caso abbiamo solo due sottocategorie (caucasian e non caucasian) per un
#dataset e ancora tutte le etnie nell'altro, quindi non possiamo usare la funzione di
#confronto delle metriche usata fino ad ora, perch teneva conto soltanto delle somme dei
#valori assoluti, allora la modifichiamo con una media.
1576 #confronto con il dataset bilanciato su female
1577 fair_metrics_cmp_graph(data1      = new_data_classified_dt,
1578                         data2 = new_data_pr,
1579                         outcome      = 'Two_yr_Recidivism',
1580                         outcome_base = 'No_recidivism',
1581                         group       = 'ethnicity',
1582                         probs1     = 'pred',
1583                         preds2     = 'preds', #per il secondo dataset non c' una colonna pred/prob, ma le
#previsioni sono gi messe in Two_yr_rec
1584                         preds_levels = c('No_recidivism','Yes_recidivism'),
1585                         cutoff1     = optimized_cutoff_new,
1586                         cutoff2     = 0.5,
1587                         base        = 'Caucasian')
1588
1589 #confronto con il dataset originale
1590 fair_metrics_cmp_graph(data1      = classified_dt,
1591                         data2 = new_data_pr,
1592                         outcome      = 'Two_yr_Recidivism',
1593                         outcome_base = 'No_recidivism',
1594                         group       = 'ethnicity',
1595                         probs1     = 'pred',

```

```

1596         probs2 =      'preds',
1597         preds_levels = c('No_recidivism', 'Yes_recidivism'),
1598         cutoff1       = optimized_cutoff,
1599         cutoff2       = 0.5,
1600         base          = 'Caucasian')
1601
1602
1603
1604
1605 ## -----
1606 #adversarial debiasing
1607 tf$reset_default_graph() #questo serve per non avere problemi con tensorflow
1608 sess = tf$compat$v1$Session()
1609 plain_model = adversarial_debiasing(privileged_groups = pri,
1610 unprivileged_groups = unpri,
1611 scope_name='plain_classifier',
1612 debias=TRUE, #CAPIRE BENE LA DIFFERENZA TRA QUESTO TRUE/FALSE
1613 sess=sess)
1614 plain_model$fit(formatted_dataset_forPred)
1615 formatted_dataset_ADVdebiasing <- plain_model$predict(formatted_dataset_forPred)
1616
1617
1618 ## -----
1619 #ora torniamo a un dataset in formato data frame (non aif360)
1620 col_names <- c(formatted_dataset_ADVdebiasing$feature_names, "preds")
1621 col_names
1622 new_data_ADVdebiasing <- data.frame(formatted_dataset_ADVdebiasing$features, formatted_
1623 dataset_ADVdebiasing$labels)
1624 colnames(new_data_ADVdebiasing) <- col_names
1625 #new_data_ADVdebiasing gi classificato
1626
1627 ## -----
1628 new_data_ADVdebiasing$Two_yr_Recidivism = new_data$Two_yr_Recidivism
1629
1630 ## -----
1631 #ripristino di nuovo TUTTI i livelli categorici per renderlo pi leggibile
1632 new_data_ADVdebiasing$Two_yr_Recidivism = ifelse(new_data_ADVdebiasing$Two_yr_Recidivism==1,
1633 "Yes_recidivism", "No_recidivism")
1634 new_data_ADVdebiasing$ethnicity = ifelse(new_data_ADVdebiasing$ethnicity==1,"Caucasian", "Not
1635 _caucasian")
1636
1637 new_data_ADVdebiasing$Female = ifelse(new_data_ADVdebiasing$Female==1,"Male", "Female")
1638
1639 new_data_ADVdebiasing$Misdemeanor = ifelse(new_data_ADVdebiasing$Misdemeanor==1,"no", "yes")

```

```

1641
1642
1643 new_data_ADVdebiasing$Age = ifelse(new_data_ADVdebiasing$Age==1, "between25_45", "Not_
1644   between25_45")
1645
1646
1647 ## -----
1648 #confronto le metriche di fairness: vedo se ci sono stati miglioramenti rispetto alle
1649   tecniche di preprocessing debias.
1650 #per questi due nuovi dataset non ho l'optimal cutoff quindi prendo 0.5 di default
1651 #confronto con il dataset bilanciato su female
1652 fair_metrics_cmp_graph(data1      = new_data_classified_dt,
1653                         data2 = new_data_ADVdebiasing,
1654                         outcome      = 'Two_yr_Recidivism',
1655                         outcome_base = 'No_recidivism',
1656                         group        = 'ethnicity',
1657                         probs1       = 'pred',
1658                         probs2       = 'preds',
1659                         preds_levels = c('No_recidivism', 'Yes_recidivism'),
1660                         cutoff1      = optimized_cutoff_new,
1661                         cutoff2      = 0.5,
1662                         base         = 'Caucasian')
1663
1664 #confronto con il dataset originale
1665 fair_metrics_cmp_graph(data1      = classified_dt,
1666                         data2 = new_data_ADVdebiasing,
1667                         outcome      = 'Two_yr_Recidivism',
1668                         outcome_base = 'No_recidivism',
1669                         group        = 'ethnicity',
1670                         probs1       = 'pred',
1671                         probs2       = 'preds',
1672                         preds_levels = c('No_recidivism', 'Yes_recidivism'),
1673                         cutoff1      = optimized_cutoff,
1674                         cutoff2      = 0.5,
1675                         base         = 'Caucasian')
1676
1677
1678 ## -----
1679 #formatted_dataset_ADVdebiasing quello predetto e new_data quello vero
1680 #la versione di new_data formattata per aif360 esiste gi (creata qualche chunck prima) e si
1681   chiama ad_fit_transformed, ma non posso usarlo qui perch contiene weights diversi dal
1682   dataset originale, allora uso formatted_dataset_forPred
1683 true_label_dt = formatted_dataset_forPred
1684 #il dataset che scelgo con le predizioni quello uscito dall'adversarial debiasing e ha le
1685   predizioni di tipo 0-1
1686 pred_label_dt = formatted_dataset_ADVdebiasing

```

```
1684  
1685  
1686  
1687 ## -----  
1688 roc <- reject_option_classification(unprivileged_groups = unpri,  
1689 privileged_groups = pri,  
1690 low_class_thresh = 0.01,  
1691 high_class_thresh = 0.99,  
1692 num_class_thresh = as.integer(100),  
1693 num_ROC_margin = as.integer(50),  
1694 metric_name = "Statistical parity difference",  
1695 metric_ub = 0.05,  
1696 metric_lb = -0.05)  
1697 roc <- roc$fit_predict(true_label_dt, pred_label_dt) #il primo il dataset contenente le  
1698 #label vere e il secondo contiene le predizioni  
1699  
1700  
1701  
1702 ## -----  
1703 #ora torniamo a un dataset in formato data frame (non aif360)  
1704 col_names <- c(roc$feature_names, "preds")  
1705 col_names  
1706 new_data_roc <- data.frame(roc$features, roc$labels)  
1707 colnames(new_data_roc) <- col_names  
1708 #new_data_ADVdebiasing gi classificato  
1709  
1710 ## -----  
1711 new_data_roc$Two_yr_Rcidivism = new_data$Two_yr_Rcidivism  
1712  
1713  
1714 ## -----  
1715 #ripristino di nuovo TUTTI i livelli categorici per renderlo pi leggibile  
1716 new_data_roc$Two_yr_Rcidivism = ifelse(new_data_roc$Two_yr_Rcidivism==1,"Yes_recidivism",  
1717 "No_recidivism")  
1718 new_data_roc$ethnicity = ifelse(new_data_roc$ethnicity==1,"Caucasian", "Not_caucasian")  
1719  
1720  
1721 new_data_roc$Female = ifelse(new_data_roc$Female==1,"Male", "Female")  
1722  
1723  
1724 new_data_roc$Misdemeanor = ifelse(new_data_roc$Misdemeanor==1,"no", "yes")  
1725  
1726  
1727 new_data_roc$Age = ifelse(new_data_roc$Age==1,"between25_45", "Not_between25_45")  
1728  
1729
```

```

1730
1731 ## -----
1732 #confronto le metriche di fairness: vedo se ci sono stati miglioramenti rispetto alle
1733 tecniche di preprocessing debias.
1734 #per questi due nuovi dataset non ho l'optimal cutoff quindi prendo 0.5 di default
1735 #confronto con il dataset classificato con adv debiasing
1736 fair_metrics_cmp_graph(data1      = new_data_ADVdebiasing,
1737                         data2 = new_data_roc,
1738                         outcome      = 'Two_yr_Recidivism',
1739                         outcome_base = 'No_recidivism',
1740                         group        = 'ethnicity',
1741                         probs1       = 'preds',
1742                         probs2       = 'preds',
1743                         preds_levels = c('No_recidivism', 'Yes_recidivism'),
1744                         cutoff1      = 0.5,
1745                         cutoff2      = 0.5,
1746                         base         = 'Caucasian')
1747
1748 #confronto con il dataset originale
1749 fair_metrics_cmp_graph(data1      = classified_dt,
1750                         data2 = new_data_roc,
1751                         outcome      = 'Two_yr_Recidivism',
1752                         outcome_base = 'No_recidivism',
1753                         group        = 'ethnicity',
1754                         probs1       = 'pred',
1755                         probs2       = 'preds',
1756                         preds_levels = c('No_recidivism', 'Yes_recidivism'),
1757                         cutoff1      = optimized_cutoff,
1758                         cutoff2      = 0.5,
1759                         base         = 'Caucasian')
1760
1761
1762 ## -----
1763 #TENTATIVO DI USO DI AIF360
1764 library(aif360)
1765 #load_aif360_lib()
1766
1767
1768 ## -----
1769 #####
1770 #convert dataset in binario
1771 #qua devo usare il balanced non ancora classificato, NON quello classificato!!!
1772 #OCCHIO: fai il run di queste righe di conversione 0-1 solo se il dataset non ancora stato
1773     convertito!!! ad esempio fai prima un check is.numeric
1774 levels(balanced_data1$Two_yr_Recidivism)[levels(balanced_data1$Two_yr_Recidivism)=="Yes_
1775     recidivism"] <- 1

```

```

1775 levels(balanced_data1$Two_yr_Recidivism)[levels(balanced_data1$Two_yr_Recidivism)=="No_"
    recidivism"] <- 0
1776 #prendo caucasian come riferiemnto perch il confronto tra le metrich di fairness lo far poi
    con caucasian come base
1777 #I caucasian sono codificati con 1 (privilegiati)
1778 levels(balanced_data1$ethnicity)[levels(balanced_data1$ethnicity)!=="Caucasian"] <- 0
1779 levels(balanced_data1$ethnicity)[levels(balanced_data1$ethnicity)=="Caucasian"] <- 1
1780 levels(balanced_data1$Female)[levels(balanced_data1$Female)=="Female"] <- 0
1781 levels(balanced_data1$Female)[levels(balanced_data1$Female)=="Male"] <- 1
1782 levels(balanced_data1$Misdemeanor)[levels(balanced_data1$Misdemeanor)=="yes"] <- 0
1783 levels(balanced_data1$Misdemeanor)[levels(balanced_data1$Misdemeanor)=="no"] <- 1
1784 levels(balanced_data1$Age)[levels(balanced_data1$Age)!=="between25_45"] <- 0
1785 levels(balanced_data1$Age)[levels(balanced_data1$Age)=="between25_45"] <- 1
1786 # #trasformo colonne categoriche in numeriche
1787 # must_convert<-sapply(balanced_data1,is.factor)           # logical vector telling if a variable
    needs to be displayed as numeric
1788 # balanced_data1<-sapply(balanced_data1[,must_convert],unclass)      # data.frame of all
    categorical variables now displayed as numeric
1789
1790
1791 ## -----
1792 #i favoriti sono gli adulti-1
1793 formatted_dataset <- aif360::aif_dataset(data_path = balanced_data1, favor_label = 0,
1794 unfavor_label = 1, #label favorita per la variabile target: 0 corrisponde a good
1795 unprivileged_protected_attribute = 0, # 0-young-unprivileged
1796 privileged_protected_attribute = 1, #se non lo scrivo, per il valore privilegiato viene
    considerato il valore pi alto
1797 target_column = "Two_yr_Recidivism", #qua prima avevo pred.rescaled
1798 protected_attribute = "ethnicity")
1799 formatted_dataset
1800
1801 #tecnica di debias preprocessing
1802 di <- disparate_impact_remover(repair_level = 0.8, sensitive_attribute = "ethnicity")
1803 rp <- di$fit_transform(formatted_dataset)
1804
1805 #reweighing (tecnica preprocessing)
1806 pri <- list("ethnicity", 1)
1807 unpri <- list("ethnicity", 0)
1808 rw <- reweighing(unpri,pri)
1809 #rw$fit(formatted_dataset)
1810 #ad_transformed <- rw$transform(formatted_dataset)
1811 ad_fit_transformed <- rw$fit_transform(rp) #questo lo faccio su rp, cos analizzo entrambe le
    tecniche in una volta sola (altrimenti dovrei fare mille classificazioni e mille casi)
1812
1813
1814 ## -----
1815 #ora torniamo a un dataset in formato data frame (non aif360)
1816 col_names <- c(ad_fit_transformed$feature_names, "Two_yr_Recidivism")

```

```

1817 col_names
1818 new_data <- data.frame(ad_fit_transformed$features, ad_fit_transformed$labels)
1819 colnames(new_data) <- col_names
1820
1821
1822
1823 #quindi new_data un dataset su cui ho fatto rebalancing per Female, disparate impact remover
  e reweighing
1824
1825
1826 ## -----
1827 #sostituisco i livelli con valori numerici
1828 levels(new_data$Two_yr_Recidivism)[levels(new_data$Two_yr_Recidivism)=="yes"] <- 1
1829 levels(new_data$Two_yr_Recidivism)[levels(new_data$Two_yr_Recidivism)=="no"] <- 0
1830 #rimuovo da new_data le colonne age_above e age_below
1831 cols_to_drop <- names(new_data) %in% c("Age_Above_FourtyFive", "Age_Below_TwentyFive")
1832 new_data <- new_data[, !cols_to_drop, drop = FALSE]
1833
1834 #split into train and test set
1835 sample_size = floor(0.7*nrow(new_data))
1836 #set a random seed per rendere l'esperimento ripetibile
1837 set.seed(42)
1838
1839 # randomly split data
1840 picked = sample(seq_len(nrow(new_data)), size = sample_size)
1841 training_set = new_data[picked,]
1842 test_set = new_data[-picked,]
1843
1844 #as.factor mi serve per fare una classificazione binaria
1845 new_data$Two_yr_Recidivism = as.factor(new_data$Two_yr_Recidivism)
1846 #devo anche eliminare recidivismo dal training_set
1847 cols_to_drop <- names(training_set) %in% c("Two_yr_Recidivism")
1848 #training e test set senza variabile target
1849 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
1850 #anche sui dati di test
1851 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
1852
1853 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="Yes_"
  recidivism"] <- 1
1854 levels(training_set$Two_yr_Recidivism)[levels(training_set$Two_yr_Recidivism)=="No_recidivism"
  "] <- 0
1855 levels(training_set$Two_yr_Recidivism)
1856
1857 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
1858 levels(test_set$Two_yr_Recidivism)[levels(test_set$Two_yr_Recidivism)=="No_recidivism"] <- 0
1859 levels(test_set$Two_yr_Recidivism)
1860 ##########

```

```

1861 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
1862   # sufficiente salvarlo e ricaricarlo)
1863 logitMod <- glm(Two_yr_Recidivism ~ ., data = training_set, family=binomial(link="logit"))
1864 logitMod #capire cosa significano questi dati
1865 pred = predict(logitMod, test_set, type = "response")
1866 new_data_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
1867   fairness
1868 classified_dt2 = cbind(test_set_data, pred)
1869 #classified_dt = rbind(classified_dt1, classified_dt2)
1870 optimized_cutoff_new = opt.cut(perf, pred1)#il cutoff ottimale leggermente pii basso di
1871   #quello standard, che sarebbe invece 0.5
1872
1873 ## -----
1874 #ripristino di nuovo TUTTI i livelli categorici per renderlo pi leggibile
1875 new_data_classified_dt$Two_yr_Recidivism = ifelse(new_data_classified_dt$Two_yr_Recidivism
1876   ==1, "Yes_recidivism", "No_recidivism")
1877
1878 new_data_classified_dt$ethnicity = ifelse(new_data_classified_dt$ethnicity==1, "Caucasian", "
1879   Not_caucasian")
1880
1881
1882 new_data_classified_dt$Female = ifelse(new_data_classified_dt$Female==1, "Male", "Female")
1883
1884
1885 new_data_classified_dt$Age = ifelse(new_data_classified_dt$Age==1, "between25_45", "Not_
1886   between25_45")
1887
1888
1889 ## -----
1890 #confronto con il dataset bilanciato su female
1891 fair_metrics_cmp_graph(data1      = balanced1_classified_dt,
1892   data2 = new_data_classified_dt,
1893   outcome      = 'Two_yr_Recidivism',
1894   outcome_base = 'No_recidivism',
1895   group        = 'ethnicity',
1896   probs1       = 'pred',
1897   probs2       = 'pred',
1898   preds_levels = c('No_recidivism', 'Yes_recidivism'),
1899   cutoff1      = optimized_cutoff1,
1900   cutoff2      = optimized_cutoff_new,
1901   base         = 'Caucasian')

```

```

1902
1903 #confronto con il dataset originale
1904 fair_metrics_cmp_graph(data1      = classified_dt,
1905                 data2 = new_data_classified_dt,
1906                 outcome      = 'Two_yr_Recidivism',
1907                 outcome_base = 'No_recidivism',
1908                 group       = 'ethnicity',
1909                 probs1     = 'pred',
1910                 probs2     = 'pred',
1911                 preds_levels = c('No_recidivism', 'Yes_recidivism'),
1912                 cutoff1    = optimized_cutoff,
1913                 cutoff2    = optimized_cutoff_new,
1914                 base       = 'Caucasian')
1915
1916
1917 ## -----
1918 #convertendo di nuovo il dataset in binario
1919 levels(new_data$Two_yr_Recidivism)[levels(new_data$Two_yr_Recidivism)=="Yes_recidivism"] <- 1
1920 levels(new_data$Two_yr_Recidivism)[levels(new_data$Two_yr_Recidivism)=="No_recidivism"] <- 0
1921 #prendo caucasian come riferimento perch il confronto tra le metrich di fairness lo far poi
#con caucasian come base
1922 #I caucasian sono codificati con 1 (privilegiati)
1923 levels(new_data$ethnicity)[levels(new_data$ethnicity)!="Caucasian"] <- 0
1924 levels(new_data$ethnicity)[levels(new_data$ethnicity)=="Caucasian"] <- 1
1925 levels(new_data$Female)[levels(new_data$Female)=="Female"] <- 0
1926 levels(new_data$Female)[levels(new_data$Female)=="Male"] <- 1
1927 levels(new_data$Misdemeanor)[levels(new_data$Misdemeanor)=="yes"] <- 0
1928 levels(new_data$Misdemeanor)[levels(new_data$Misdemeanor)=="no"] <- 1
1929 levels(new_data$Age)[levels(new_data$Age)!="between25_45"] <- 0
1930 levels(new_data$Age)[levels(new_data$Age)=="between25_45"] <- 1
1931 # #trasformo colonne categoriche in numeriche
1932 # must_convert<-sapply(new_data,is.factor)      # logical vector telling if a variable needs
# to be displayed as numeric
1933 # new_data<-sapply(new_data[,must_convert],unclass)   # data.frame of all categorical
# variables now displayed as numeric
1934
1935
1936 ## -----
1937 #creo dataset formattato per aif360
1938 #NOTA BENE: formatted_dataset la versione di raw_data per aif360. formatted_dataset sar
# SEMPRE quello che uso per il fit dei metodidi debias e far la predizione sul formatted_
# dataset di new_data
1939 #creo quindi il formatted dataset di new_data e lo chiamo
1940 formatted_dataset_forPred <- aif360::aif_dataset(data_path = new_data, favor_label = 0,
1941 unfavor_label = 1, #label favorita per la variabile target: 0 corrisponde a good
1942 unprivileged_protected_attribute = 0, # 0-young-unprivileged
1943 privileged_protected_attribute = 1, #se non lo scrivo, per il valore privilegiato viene
# considerato il valore pi alto

```

```
1944 target_column = "Two_yr_Rcidivism", #qua prima avevo pred.rescaled
1945 protected_attribute = "ethnicity")
1946
1947
1948 ## -----
1949 #applico tecniche di debias inprocessing: adversarial debiasing and prejudice remover
1950 #prejudice remover
1951 model <- prejudice_remover(class_attr = "Two_yr_Rcidivism", sensitive_attr = "ethnicity")
1952 model$fit(formatted_dataset_forPred)
1953 formatted_dataset_pr <- model$predict(formatted_dataset_forPred) #FUNZIONA ma devi fare
    questo come prima tecnica in assoluto
1954 #in formatted_dataset_pr ho quindi IL NUOVO DATASET FORMATTATO con le predizioni del
    recidivismo fatto con prejudice remover
1955
1956
1957
1958 ## -----
1959 #ora torniamo a un dataset in formato data frame (non aif360)
1960 col_names <- c(formatted_dataset_pr$feature_names, "preds")#questa colonna contiene le
    predizioni sul recidivismo appena fatte dall'algoritmo
1961 col_names
1962 new_data_pr <- data.frame(formatted_dataset_pr$features, formatted_dataset_pr$labels)
1963 colnames(new_data_pr) <- col_names
1964 #new_data_pr in questo caso gi classificato!!!
1965
1966
1967 ## -----
1968 new_data_pr$Two_yr_Rcidivism = new_data$Two_yr_Rcidivism
1969
1970
1971 ## -----
1972 #ripristino di nuovo TUTTI i livelli categorici per renderlo pi leggibile
1973 new_data_pr$Two_yr_Rcidivism = ifelse(new_data_pr$Two_yr_Rcidivism==1,"Yes_recidivism", "
    No_recidivism")
1974
1975 new_data_pr$ethnicity = ifelse(new_data_pr$ethnicity==1,"Caucasian", "Not_caucasian")
1976
1977
1978 new_data_pr$Female = ifelse(new_data_pr$Female==1,"Male", "Female")
1979
1980
1981 new_data_pr$Misdemeanor = ifelse(new_data_pr$Misdemeanor==1,"no", "yes")
1982
1983
1984 new_data_pr$Age = ifelse(new_data_pr$Age==1,"between25_45", "Not_between25_45")
1985
1986
1987
```

```

1988 ## -----
1989 #confronto le metriche con quelle di
1990 #new_data_pr
1991 #source("util.R")
1992
1993 #NOTA BENE: in questo caso abbiamo solo due sottocategorie (caucasian e non caucasian) per un
    dataset e ancora tutte le etnie nell'altro, quindi non possiamo usare la funzione di
    confronto delle metriche usata fino ad ora, perch teneva conto soltanto delle somme dei
    valori assoluti, allora la modifichiamo con una media.
1994 #confronto con il dataset bilanciato su female
1995 fair_metrics_cmp_graph(data1      = new_data_classified_dt,
1996             data2 = new_data_pr,
1997             outcome      = 'Two_yr_Recidivism',
1998             outcome_base = 'No_recidivism',
1999             group        = 'ethnicity',
2000             probs1       = 'pred',
2001             probs2       = 'preds', #per il secondo dataset non c' una colonna pred/prob, ma le
                previsioni sono gi messe in Two_yr_rec
2002             preds_levels = c('No_recidivism', 'Yes_recidivism'),
2003             cutoff1      = optimized_cutoff_new,
2004             cutoff2      = 0.5,
2005             base         = 'Caucasian')
2006
2007 #confronto con il dataset originale
2008 fair_metrics_cmp_graph(data1      = classified_dt,
2009             data2 = new_data_pr,
2010             outcome      = 'Two_yr_Recidivism',
2011             outcome_base = 'No_recidivism',
2012             group        = 'ethnicity',
2013             probs1       = 'pred',
2014             probs2       = 'preds',
2015             preds_levels = c('No_recidivism', 'Yes_recidivism'),
2016             cutoff1      = optimized_cutoff,
2017             cutoff2      = 0.5,
2018             base         = 'Caucasian')
2019
2020
2021
2022
2023 ## -----
2024 #adversarial debiasing
2025 tf$reset_default_graph() #questo serve per non avere problemi con tensorflow
2026 sess = tf$compat$v1$Session()
2027 plain_model = adversarial_debiasing(privileged_groups = pri,
2028 unprivileged_groups = unpri,
2029 scope_name='plain_classifier',
2030 debias=TRUE, #CAPIRE BENE LA DIFFERENZA TRA QUESTO TRUE/FALSE
2031 sess=sess)

```

```

2032 plain_model$fit(formatted_dataset_forPred)
2033 formatted_dataset_ADVdebiasing <- plain_model$predict(formatted_dataset_forPred)
2034
2035
2036 ## -----
2037 #ora torniamo a un dataset in formato data frame (non aif360)
2038 col_names <- c(formatted_dataset_ADVdebiasing$feature_names, "preds")
2039 col_names
2040 new_data_ADVdebiasing <- data.frame(formatted_dataset_ADVdebiasing$features, formatted_
2041   dataset_ADVdebiasing$labels)
2042 colnames(new_data_ADVdebiasing) <- col_names
2043 #new_data_ADVdebiasing gi classificato
2044
2045 ## -----
2046 new_data_ADVdebiasing$Two_yr_Recidivism = new_data$Two_yr_Recidivism
2047
2048 ## -----
2049 #ripristino di nuovo TUTTI i livelli categorici per renderlo pi leggibile
2050 new_data_ADVdebiasing$Two_yr_Recidivism = ifelse(new_data_ADVdebiasing$Two_yr_Recidivism==1,
2051   "Yes_recidivism", "No_recidivism")
2052 new_data_ADVdebiasing$ethnicity = ifelse(new_data_ADVdebiasing$ethnicity==1, "Caucasian", "Not_
2053   _caucasian")
2054
2055 new_data_ADVdebiasing$Female = ifelse(new_data_ADVdebiasing$Female==1, "Male", "Female")
2056
2057
2058 new_data_ADVdebiasing$Misdemeanor = ifelse(new_data_ADVdebiasing$Misdemeanor==1, "no", "yes")
2059
2060
2061 new_data_ADVdebiasing$Age = ifelse(new_data_ADVdebiasing$Age==1, "between25_45", "Not_
2062   between25_45")
2063
2064
2065 ## -----
2066 #confronto le metriche di fairness: vedo se ci sono stati miglioramenti rispetto alle
2067   tecniche di preprocessing debias.
2068 #per questi due nuovi dataset non ho l'optimal cutoff quindi prendo 0.5 di default
2069 #confronto con il dataset bilanciato su female
2070 fair_metrics_cmp_graph(data1      = new_data_classified_dt,
2071                         data2 = new_data_ADVdebiasing,
2072                         outcome    = 'Two_yr_Recidivism',
2073                         outcome_base = 'No_recidivism',
2074                         group      = 'ethnicity',
2075                         probs1 =     'pred',

```

```

2075           probs2 =      'preds',
2076           preds_levels = c('No_recidivism', 'Yes_recidivism'),
2077           cutoff1      = optimized_cutoff_new,
2078           cutoff2      = 0.5,
2079           base        = 'Caucasian')
2080
2081 #confronto con il dataset originale
2082 fair_metrics_cmp_graph(data1          = classified_dt,
2083                         data2 = new_data_ADVdebiasing,
2084                         outcome = 'Two_yr_Recidivism',
2085                         outcome_base = 'No_recidivism',
2086                         group = 'ethnicity',
2087                         probs1 = 'pred',
2088                         probs2 = 'preds',
2089                         preds_levels = c('No_recidivism', 'Yes_recidivism'),
2090                         cutoff1 = optimized_cutoff,
2091                         cutoff2 = 0.5,
2092                         base = 'Caucasian')
2093
2094
2095
2096 ## -----
2097 #formatted_dataset_ADVdebiasing quello predetto e new_data quello vero
2098 #la versione di new_data formattata per aif360 esiste gi (creata qualche chunck prima) e si
  chiama ad_fit_transformed, ma non posso usarlo qui perch contiene weights diversi dal
  dataset originale, allora uso formatted_dataset_forPred
2099 true_label_dt = formatted_dataset_forPred
2100 #il dataset che scelgo con le predizioni quello uscito dall'adversarial debiasing e ha le
  predizioni di tipo 0-1
2101 pred_label_dt = formatted_dataset_ADVdebiasing
2102
2103
2104
2105 ## -----
2106 roc <- reject_option_classification(unprivileged_groups = unpri,
2107 privileged_groups = pri,
2108 low_class_thresh = 0.01,
2109 high_class_thresh = 0.99,
2110 num_class_thresh = as.integer(100),
2111 num_ROC_margin = as.integer(50),
2112 metric_name = "Statistical parity difference",
2113 metric_ub = 0.05,
2114 metric_lb = -0.05)
2115 roc <- roc$fit_predict(true_label_dt, pred_label_dt) #il primo il dataset contenente le
  label vere e il secondo contiene le predizioni
2116
2117 ## -----
2118 #ora torniamo a un dataset in formato data frame (non aif360)

```

```
2119 col_names <- c(roc$feature_names, "preds")
2120 col_names
2121 new_data_roc <- data.frame(roc$features, roc$labels)
2122 colnames(new_data_roc) <- col_names
2123 #new_data_ADVdebiasing gi classificato
2124
2125 ## -----
2126 new_data_roc$Two_yr_Recidivism = new_data$Two_yr_Recidivism
2127
2128
2129 ## -----
2130 #ripristino di nuovo TUTTI i livelli categorici per renderlo pi leggibile
2131 new_data_roc$Two_yr_Recidivism = ifelse(new_data_roc$Two_yr_Recidivism==1,"Yes_recidivism",
2132   "No_recidivism")
2133
2134 new_data_roc$ethnicity = ifelse(new_data_roc$ethnicity==1,"Caucasian", "Not_caucasian")
2135
2136 new_data_roc$Female = ifelse(new_data_roc$Female==1,"Male", "Female")
2137
2138
2139 new_data_roc$Misdemeanor = ifelse(new_data_roc$Misdemeanor==1,"no", "yes")
2140
2141
2142 new_data_roc$Age = ifelse(new_data_roc$Age==1,"between25_45", "Not_between25_45")
2143
2144
2145 ## -----
2146 #confronto le metriche di fairness: vedo se ci sono stati miglioramenti rispetto alle
2147   tecniche di preprocessing debias.
2148 #per questi due nuovi dataset non ho l'optimal cutoff quindi prendo 0.5 di default
2149 #confronto con il dataset classificato con adv debiasing
2150 fair_metrics_cmp_graph(data1           = new_data_ADVdebiasing,
2151                         data2 = new_data_roc,
2152                         outcome      = 'Two_yr_Recidivism',
2153                         outcome_base = 'No_recidivism',
2154                         group        = 'ethnicity',
2155                         probs1       = 'preds',
2156                         probs2       = 'preds',
2157                         preds_levels = c('No_recidivism','Yes_recidivism'),
2158                         cutoff1      = 0.5,
2159                         cutoff2      = 0.5,
2160                         base         = 'Caucasian')
2161
2162 #confronto con il dataset originale
2163 fair_metrics_cmp_graph(data1           = classified_dt,
2164                         data2 = new_data_roc,
2165                         outcome      = 'Two_yr_Recidivism',
```

```
2165     outcome_base = 'No_recidivism',
2166     group        = 'ethnicity',
2167     probs1       = 'pred',
2168     probs2       = 'preds',
2169     preds_levels = c('No_recidivism', 'Yes_recidivism'),
2170     cutoff1      = optimized_cutoff,
2171     cutoff2      = 0.5,
2172     base         = 'Caucasian')
```

R code for Drug dataset:

```
1 ## ----echo=FALSE, warning=FALSE-----
2 library(fairness)
3 library(knitr)
4 library(recipes)
5 library(skimr)
6 library(modeldata)
7 library(RColorBrewer)
8 library(tangram)
9 library(dplyr)
10 library(vegan)
11 library(diverse)
12 library(randomForest)
13 library(caret)
14 library(varhandle)
15 library(MASS)
16 library(dostats) #per %contains%
17 library(graphics) #per mosaicplot
18 library(corrplot)
19 library(wesanderson)
20 library(Amelia)
21 library(ROCR) #per ROC function
22 library(ROSE) #binary balancing
23 library(UBL) #multiclass balancing
24 library(ggplot2)
25 library(ggpubr)
26 library(xtable)
27 library(tidyr)
28 library(formatR)
29
30 #carico il file util con le altre funzioni
31 source("util.R")
32
33
34 ## ---- echo = F-----
35 library(knitr)
36 opts_chunk$set(tidy.opts=list(width.cutoff=60), tidy=TRUE)
37
38
```

```
39 ## ----- echo=FALSE -----
40 #Load data
41 raw_data <- read.csv(file = "drug_consumption.data")
42
43
44 ## -----
45 knitr:::purl('Drug.Rmd')
46
47
48 ## -----
49 #Data preparation
50 column.names <- c("ID", "Age", "Gender", "Education", "Country", "Ethnicity", "Nscore", ""
51   "Escore", "Oscore", "Ascore", "Cscore", "Impulsive", "SS", "Alcohol", "Amphet", "Amyl", ""
52   "Benzos", "Caff", "Cannabis", "Choc", "Coke", "Crack", "Ecstasy", "Heroin", "Ketamine", ""
53   "Legalh", "LSD", "Meth", "Mushrooms", "Nicotine", "Semer", "VSA")
54 raw_data = setNames(raw_data, column.names)
55
56
57 ## -----
58 #encode from numerical to categorical
59 #potrei anche considerare "usata pi di 10 anni fa" come Never used
60 #Qui devo mettere stringhe e non valori numerici, oppure cambiarli dopo in stringhe,
61   #altrimenti poi fa regressione invece di classificazione
62 raw_data[raw_data == "CL0"]<- "Never_used" #never used
63 raw_data[raw_data == "CL1"]<- "Used" #used more than a decade ago
64 raw_data[raw_data == "CL2"]<- "Used" #used within the last decade
65 raw_data[raw_data == "CL3"]<- "Used"
66 raw_data[raw_data == "CL4"]<- "Used"
67 raw_data[raw_data == "CL5"]<- "Used"
68 raw_data[raw_data == "CL6"]<- "Used"
69
70 #elimino la colonna ID perch non mi interessa
71 raw_data$ID = NULL
72
73 require(dplyr)
74 require(magrittr)
75 raw_data = raw_data %>%
76   mutate_at(.vars= vars(Age:Ethnicity), .funs = funs(as.factor)) %>%           mutate(Age =
77     factor(Age, labels = c("18_24", "25_34", "35_44", "45_54", "55_64", "65_")))) %>%
78     mutate(Gender = factor(Gender, labels = c("Male", "Female")))) %>%
79     mutate(Education = factor(Education, labels = c("Under16", "At16", "At17", "At18",
80       ", "SomeCollege", "ProfessionalCert", "Bachelors", "Masters", "Doctorate")))) %>%
81     mutate(Country = factor(Country, labels = c("USA", "NewZealand", "Other", "Australia",
82       ", "Ireland", "Canada", "UK")))) %>%
```

```

80     mutate(Ethnicity = factor(Ethnicity, labels = c("Black", "Asian", "White", " "
81     "White/Black", "Other", "White/Asian", "Black/Asian")))) %>%
82     mutate(Coke = factor(Coke, labels = c("Never_used", "Used")))) %>%
83     mutate(Crack = factor(Crack, labels = c("Never_used", "Used"))))
84
85
86 # data.table(raw_data, options = list(scrollX = TRUE, searching = FALSE, pageLength = 5),
87             caption = 'Table 2: Clean Drug Use Data')
88
89 #seleziona due colonne di droghe che mi interessano
90 #scelgo Crack e Coke (cocaina)
91 #elimino tutte le altre colonne che non mi interessano
92 raw_data = subset(raw_data, select=-c(Alcohol, Amphet, Amyl, Benzos, Caff, Cannabis, Choc,
93     Ecstasy, Heroin, Ketamine, Legalh, LSD, Meth, Mushrooms, Nicotine, Semer, VSA) )
94
95 h = head(raw_data)
96
97
98
99 ## -----
100
101 #data visualization CATEGORICAL
102
103 #Filter out categorical columns
104 cat_data <- raw_data[, sapply(raw_data, is.categorical)]
105 cat_names <- names(cat_data)
106
107 #pie print per ogni colonna categorica
108 for ( i in seq(1,length( cat_data ),1) ){
109   pie_print(cat_data,cat_names[i])
110 }
111
112 #####
113 a = table(cat_data) #lo faccio solo sui categorici
114 a = as.data.frame(a)
115 #a$Number_of_Priors = NULL
116 ggballoonplot(a, x = "Ethnicity", y = "Age", size = "Freq",
117                 color = "#8591C2", fill = "#8591C2", facet_by = "Gender",
118                 ggtheme = theme_bw()) #
119 #scale_fill_gradientn(colors = wes_palette(2, name="GrandBudapest"))
120
121 ## -----
122 #giusto per farci un'idea, quanti hanno consumato cocaina e quanti hanno consumato crack nel
123 nostro dataset
124 table(raw_data$Coke) #847

```

```
124 table(raw_data$Crack) #258
125
126
127 ## -----
128 #missing values
129 missmap(raw_data, main = "Missing values vs observed")
130 #no missing values
131
132
133 ## -----
134 # Create a boxplot using base R
135 #posso provare a farlo con gli score, ma dovrei capire se davvero questi score (o anche solo
136 # uno) mi interessano
137 boxplot(Nscore~Ethnicity,data=raw_data)
138 boxplot(Nscore~Age,data=raw_data)
139 boxplot(Nscore~Gender,data=raw_data)
140
141 ## -----
142 ##COKE
143
144 #plotto lo Coke consumption al variare di Gender
145 ##########
146 d = group_by_feature_sensitive(raw_data, "Coke", "Gender")
147 bar(dv = count_val,
148     factors = c("Coke", "Gender"), #non riesco a risolvere il problema di passare in questa
149     # lista una variabile contenente la stringa, invece della stringa stessa.
150     dataframe = as.data.frame(d),
151     errbar = FALSE,
152     ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
153     ) #I increased the upper y-limit to accommodate the legend.
154 #####
155 d = group_by_feature_sensitive(raw_data, "Coke", "Ethnicity")
156 bar(dv = count_val,
157     factors = c("Coke", "Ethnicity"), #non riesco a risolvere il problema di passare in
158     # questa lista una variabile contenente la stringa, invece della stringa stessa.
159     dataframe = as.data.frame(d),
160     errbar = FALSE,
161     ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
162     ) #I increased the upper y-limit to accommodate the legend.
163 ##########
164 d = group_by_feature_sensitive(raw_data, "Coke", "Education")
165 bar(dv = count_val,
166     factors = c("Coke", "Education"), #non riesco a risolvere il problema di passare in
167     # questa lista una variabile contenente la stringa, invece della stringa stessa.
168     dataframe = as.data.frame(d),
169     errbar = FALSE,
```

```
168     ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
169     ) #I increased the upper y-limit to accommodate the legend.
170 #####Age
171 d = group_by_feature_sensitive(raw_data, "Coke", "Age")
172 bar(dv = count_val,
173     factors = c("Coke", "Age"), #non riesco a risolvere il problema di passare in questa
174     lista una variabile contenente la stringa, invece della stringa stessa.
175     dataframe = as.data.frame(d),
176     errbar = FALSE,
177     ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
178     ) #I increased the upper y-limit to accommodate the legend.
179 #####Country
180 d = group_by_feature_sensitive(raw_data, "Coke", "Country")
181 bar(dv = count_val,
182     factors = c("Coke", "Country"), #non riesco a risolvere il problema di passare in questa
183     lista una variabile contenente la stringa, invece della stringa stessa.
184     dataframe = as.data.frame(d),
185     errbar = FALSE,
186     ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
187     ) #I increased the upper y-limit to accommodate the legend.
188 ## -----
189 ##Crack
190
191 #plotto lo Crack consumption al variare di Gender
192 #####Gender
193 d = group_by_feature_sensitive(raw_data, "Crack", "Gender")
194 bar(dv = count_val,
195     factors = c("Crack", "Gender"), #non riesco a risolvere il problema di passare in questa
196     lista una variabile contenente la stringa, invece della stringa stessa.
197     dataframe = as.data.frame(d),
198     errbar = FALSE,
199     ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
200     ) #I increased the upper y-limit to accommodate the legend.
201
202 #####Ethnicity
203 d = group_by_feature_sensitive(raw_data, "Crack", "Ethnicity")
204 bar(dv = count_val,
205     factors = c("Crack", "Ethnicity"), #non riesco a risolvere il problema di passare in
206     questa lista una variabile contenente la stringa, invece della stringa stessa.
207     dataframe = as.data.frame(d),
208     errbar = FALSE,
209     ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
210     ) #I increased the upper y-limit to accommodate the legend.
211 #####Education
```

```

212 questa lista una variabile contente la stringa, invece della stringa stessa.
213 dataframe = as.data.frame(d),
214 errbar = FALSE,
215 ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
216 ) #I increased the upper y-limit to accommodate the legend.
217 ##########
218 d = group_by_feature_sensitive(raw_data, "Crack", "Age")
219 bar(dv = count_val,
220 factors = c("Crack", "Age"), #non riesco a risolvere il problema di passare in questa
221 lista una variabile contente la stringa, invece della stringa stessa.
222 dataframe = as.data.frame(d),
223 errbar = FALSE,
224 ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
225 ) #I increased the upper y-limit to accommodate the legend.
226 ##########
227 d = group_by_feature_sensitive(raw_data, "Crack", "Country")
228 bar(dv = count_val,
229 factors = c("Crack", "Country"), #non riesco a risolvere il problema di passare in questa
230 lista una variabile contente la stringa, invece della stringa stessa.
231 dataframe = as.data.frame(d),
232 errbar = FALSE,
233 ylim=c(0, as.numeric(d[4])+0.1*as.numeric(d[4]))
234 ) #I increased the upper y-limit to accommodate the legend.
235 ## -----
236 #Coke
237 frquent_comb1(raw_data, filter = "Coke", "Age")
238 frquent_comb1(raw_data, filter = "Coke", c("Age", "Ethnicity"))
239 frquent_comb1(raw_data, filter = "Coke", c("Age", "Ethnicity", "Gender"))
240 frquent_comb1(raw_data, filter = "Coke", c("Age", "Ethnicity", "Gender", "Education"))
241 ## -----
242 ##########
243 #plotto la distribuzione delle combinazioni (conteggio casi)
244 a = frquent_comb1(raw_data, filter = "Coke", c("Age", "Ethnicity", "Gender"))
245 a1 = as.data.frame(a[1]) #la tabella 1 ha sempre Used
246 a2 = as.data.frame(a[2]) #la tabella 2 ha sempre Never_Used
247
248 #Coloro sempre Used di rosso
249
250 ggballoonplot(a1, x = "Ethnicity", y = "Age", size = "percentage_on_Used",
251 color = "black", fill = "#F24D4D", facet.by = "Gender", ggtheme = theme_bw())
252 #percentage_on_Age_ethnicity_Female
253
254 ggballoonplot(a2, x = "Ethnicity", y = "Age", size = "percentage_on_Never_used",
255 color = "black", fill = "#50C395", facet.by = "Gender", ggtheme = theme_bw())
256

```

```

257 ggballoonplot(a1, x = "Ethnicity", y = "Age", size = "percentage_on_total",
258             color = "black", fill = "#F24D4D", facet.by = "Gender", ggtheme = theme_bw())
259 #percentage_on_Age_ethnicity_Female
260
261 ggballoonplot(a2, x = "Ethnicity", y = "Age", size = "percentage_on_Age_Ethnicity_Gender",
262             color = "black", fill = "#50C395", facet.by = "Gender", ggtheme = theme_bw())
263
264
265 g = ggballoonplot(a1, x = "Ethnicity", y = "Age", size = "percentage_on_Age_Ethnicity_Gender"
266             ,
267             color = "black", fill = "#F24D4D", facet.by = "Gender", ggtheme = theme_bw())
268 ggsave("DRUGSCokeperOnComb.jpg", plot = g, width = 20, height = 12, units = "cm")
269
270 ggballoonplot(a2, x = "Ethnicity", y = "Age", size = "percentage_on_total",
271             color = "black", fill = "#50C395", facet.by = "Gender", ggtheme = theme_bw())
272
273
274
275 ## -----
276 #Crack
277 frquent_comb1(raw_data, filter = "Crack", "Age")
278 frquent_comb1(raw_data, filter = "Crack", c("Age", "Ethnicity"))
279 frquent_comb1(raw_data, filter = "Crack", c("Age", "Ethnicity", "Gender"))
280 frquent_comb1(raw_data, filter = "Crack", c("Age", "Ethnicity", "Gender", "Education"))
281 #non posso fare tutte le combinazioni. C' un modo per capire a priori quali attributi possono
282             essere i pi interessanti da valutare?
283
284 ## -----
285
286 #####
287 #plotto la distribuzione delle combinazioni (conteggio casi)
288 a = frquent_comb1(raw_data, filter = "Crack", c("Age", "Ethnicity", "Gender"))
289
290 a1 = as.data.frame(a[2]) #la tabella 1 ha sempre Used
291 a2 = as.data.frame(a[1]) #la tabella 2 ha sempre Never_Used
292
293 #Coloro sempre Used di rosso
294
295 ggballoonplot(a1, x = "Ethnicity", y = "Age", size = "percentage_on_Used",
296             color = "black", fill = "#F24D4D", facet.by = "Gender", ggtheme = theme_bw())
297 #percentage_on_Age_ethnicity_Female
298
299 ggballoonplot(a2, x = "Ethnicity", y = "Age", size = "percentage_on_Never_used",
300             color = "black", fill = "#50C395", facet.by = "Gender", ggtheme = theme_bw())
301
302 ggballoonplot(a1, x = "Ethnicity", y = "Age", size = "percentage_on_total",

```

```
303           color = "black", fill = "#F24D4D", facet.by = "Gender", ggtheme = theme_bw())
304 #percentage_on_Age_ethnicity_Female
305
306 ggballoonplot(a2, x = "Ethnicity", y = "Age", size = "percentage_on_Age_Ethnicity_Gender",
307                 color = "black", fill = "#50C395", facet.by = "Gender", ggtheme = theme_bw())
308
309
310 g = ggballoonplot(a1, x = "Ethnicity", y = "Age", size = "percentage_on_Age_Ethnicity_Gender"
311
312           ,
313           color = "black", fill = "#F24D4D", facet.by = "Gender", ggtheme = theme_bw())
314 #percentage_on_Age_ethnicity_Female
315 ggsave("DRUGSCrackPerOnComb.jpg", plot = g, width = 20, height = 12, units = "cm")
316
317
318
319 ## -----
320 #chi squared test (consideriamo le colonne come categoriche)
321 source("util.R")
322 chisq_fun(raw_data, "Coke", "Age")
323 chisq_fun(raw_data, "Coke", "Ethnicity")
324 chisq_fun(raw_data, "Coke", "Gender")
325
326
327 ## -----
328 chisq_fun(raw_data, "Crack", "Age")
329 chisq_fun(raw_data, "Crack", "Ethnicity")
330 chisq_fun(raw_data, "Crack", "Gender")
331 #In console possiamo osservare la differenza tra i valori osservati (prima tabella) e i
332     valori attesi (seconda tabella).
333
334
335 ## -----
336 #SHANNON INDEX
337 #uso funzioni create nella libreria
338 shannon_diversity(raw_data)
339 print_shannon(raw_data)
340 #print_cmp_shannon(raw_data, raw_data)
341
342
343 ## -----
344 #sostituisco i livelli con valori numerici
345 raw_data$Coke = as.factor(raw_data$Coke)
346 levels(raw_data$Coke)[levels(raw_data$Coke)=="Used"] <- 1
347 levels(raw_data$Coke)[levels(raw_data$Coke)=="Never_used"] <- 0
348
349
350 #split into train and test set
```

```

349 sample_size = floor(0.7*nrow(raw_data))
350 #set a random seed per rendere l'esperimento ripetibile
351 set.seed(42)
352 # randomly split data
353 picked = sample(seq_len(nrow(raw_data)), size = sample_size)
354 training_set = raw_data[picked,]
355 test_set = raw_data[-picked,]

356
357 #as.factor mi serve per fare una classificazione binaria
358 raw_data$Coke = as.factor(raw_data$Coke)
359 #devo anche eliminare Coke dal training_set
360 cols_to_drop <- names(training_set) %in% c("Coke")
361 #training e test set senza variabile target
362 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
363 #anche sui dati di test
364 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
365
366
367 ## -----
368 #REGRESSIONE LOGISTICA
369 #elimino colonna crack
370 cols_to_drop <- names(raw_data) %in% c("Crack")
371 raw_data_copy = raw_data
372 raw_data <- raw_data[, !cols_to_drop, drop = FALSE]
373
374 #####
375 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
376 logitMod <- glm(Coke ~ ., data = training_set, family=binomial(link="logit"))
377 logitMod #capire cosa significano questi dati
378 #saveRDS(model, "./logitMod_compas_rf.rds") #commento perch l'ho già salvato
379 #####
380 #logitMod = readRDS(file = "./model_compas_rf.rds")
381 #logitMod
382
383 #faccio la predizione su entrambi training e test set, perch per le metriche di
# classificazione mi servono dataset originale e dataset classificato *della stessa
# dimensione*
384 #pred = predict(logitMod, training_set_data)
385 #classified_dt1 = cbind(training_set_data, pred)
386 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
387 classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria fairness
388 classified_dt2 = cbind(test_set_data, pred)
389 #classified_dt = rbind(classified_dt1, classified_dt2)
390

```

```

391
392
393 ## -----
394 #ROC
395 pred1 <- prediction(pred, test_set$Coke) #pred1 un oggetto pred.obj di ROCR
396 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
397 plot(perf)
398 #posso anche fare una ROC comparison per le categorie di un attributo sensibile, tipo Female,
      con la libreria fairness:
399 roc_fairness <- roc_parity(data          = classified_dt,
400                               outcome       = 'Coke',
401                               group         = 'Gender',
402                               probs         = 'pred',
403                               preds_levels = c('Used', 'Never_used'),
404                               base          = 'Male')
405 roc_fairness$Metric
406 roc_fairness$ROCAUC_plot
407
408 #AUC
409 auc.perf = performance(pred1, measure = "auc")
410 auc.perf@y.values #stampa a video il valore di AUC
411 #####
412 #plotto sensitivity e specificity in modo da individuare il cutoff ottimizzato
413 sens <- data.frame(x=unlist(performance(pred1, "sens")@x.values),
414                      y=unlist(performance(pred1, "sens")@y.values))
415 spec <- data.frame(x=unlist(performance(pred1, "spec")@x.values),
416                      y=unlist(performance(pred1, "spec")@y.values))
417
418 sens %>% ggplot(aes(x,y)) +
419   geom_line() +
420   geom_line(data=spec, aes(x,y,col="red")) +
421   scale_y_continuous(sec.axis = sec_axis(~., name = "Specificity")) +
422   labs(x='Cutoff', y="Sensitivity") +
423   theme(axis.title.y.right = element_text(colour = "red"), legend.position="none")
424 #####
425
426 ## -----
427 #troviamo l'optimal cutoff, cio l'optimal score che minimizza il misclassification error del
      modello di regressione logistica
428 #The best threshold (or cutoff) point to be used in glm models is the point which maximises
      the specificity and the sensitivity. This threshold point might not give the highest
      prediction in your model, but it wouldn't be biased towards positives or negatives.
429 optimized_cutoff = opt.cut(perf, pred1)#il cutoff ottimale leggermente piu basso di quello
      standard, che sarebbe invece 0.5
430
431 #convertendo la colonna pred di classified_dt2 in yes/no recidivism usando il cutoff ottimizzato
432 #yes era 1, quindi ho recidivismo se maggiore del cutoff
433 classified_dt2$pred = ifelse(classified_dt2$pred<optimized_cutoff, "Never_used", "Used")

```

```

434 #sostituisco i valori dei livelli per poter plottare la matrice di confusione
435 levels(test_set$Coke)[levels(test_set$Coke)==1] <- "Used"
436 levels(test_set$Coke)[levels(test_set$Coke)==0] <- "Never_used"
437 #levels(test_set$Coke)
438 #anche per classified_dt2
439 classified_dt2$pred <- factor(classified_dt2$pred, levels = c("Used", "Never_used"))
440
441 ## ----message=FALSE, warning=FALSE-----
442 #matrice di confusione
443 cm <- confusionMatrix(data = as.factor(classified_dt2$pred), reference = as.factor(test_set$Coke))
444 #plotto la matrice di confusione
445 png("DRUGSCokeCM.png")
446 draw_confusion_matrix(cm)
447 dev.off()
448
449 ## -----
450 #ci occupiamo di group fairness: indipendenza, separazione e sufficienza
451
452 #lo facciamo per tutti i sensitive attribute (Marital, ageR)
453
454 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
455 levels(classified_dt$Coke)[levels(classified_dt$Coke)==1] <- "Used"
456 levels(classified_dt$Coke)[levels(classified_dt$Coke)==0] <- "Never_used"
457
458 #per scegliere la categoria "base" per cui confrontare tutte le altre, per ogni attributo
#sensibile, ci conviene guardare le tabelle delle frequenze plottate in precedenza:
459
460
461 ## -----
462 fair_metrics(data      = classified_dt,
463                 outcome    = 'Coke',
464                 group      = 'Age',
465                 probs     = 'pred',
466                 preds_levels = c('Never_used', 'Used'),
467                 cutoff     = optimized_cutoff,
468                 base       = '18_24')
469
470 fair_metrics(data      = classified_dt,
471                 outcome    = 'Coke',
472                 group      = 'Ethnicity',
473                 probs     = 'pred',
474                 preds_levels = c('Never_used', 'Used'),
475                 cutoff     = optimized_cutoff,
476                 base       = 'White')
477
478 fair_metrics(data      = classified_dt,
479                 outcome    = 'Coke',

```

```
480             group      = 'Gender',
481             probs     = 'pred',
482             preds_levels = c('Never_used', 'Used'),
483             cutoff     = optimized_cutoff,
484             base       = 'Male')

485 fair_metrics_cmp_graph2(data      = classified_dt,
486                         outcome   = 'Coke',
487                         group1    = 'Gender',
488                         group2 = 'Ethnicity',
489                         probs     = 'pred',
490                         preds_levels = c('Never_used', 'Used'),
491                         cutoff     = optimized_cutoff,
492                         base1     = 'Male',
493                         base2     = 'White')

495 fair_metrics_cmp_graph2(data      = classified_dt,
496                         outcome   = 'Coke',
497                         group1    = 'Age',
498                         group2 = 'Ethnicity',
499                         probs     = 'pred',
500                         preds_levels = c('Never_used', 'Used'),
501                         cutoff     = optimized_cutoff,
502                         base1     = '18_24',
503                         base2     = 'White')

504 fair_metrics_cmp_graph2(data      = classified_dt,
505                         outcome   = 'Coke',
506                         group1    = 'Ethnicity',
507                         group2 = 'Age',
508                         probs     = 'pred',
509                         preds_levels = c('Never_used', 'Used'),
510                         cutoff     = optimized_cutoff,
511                         base1     = 'White',
512                         base2     = '18_24')

513 fair_metrics_cmp_graph2(data      = classified_dt,
514                         outcome   = 'Coke',
515                         group1    = 'Gender',
516                         group2 = 'Age',
517                         probs     = 'pred',
518                         preds_levels = c('Never_used', 'Used'),
519                         cutoff     = optimized_cutoff,
520                         base1     = 'Male',
521                         base2     = '18_24')

522
523
524 ## -----
525 #CREO I SEGUENTI DATASET BILANCIATI PER COKE:
526 #balanced_data0: bilanciato per Ethnicity
527 #balanced_data1: bilanciato per Country
```

```
528 #balanced_data2: bilanciato per Ethnicity and Age
529
530 #balanced_data0: bilanciato per Ethnicity
531 #questo invece ha ethnicity bilanciato con smote ed meglio usare questo!
532 raw_data = as.data.frame(raw_data)
533 balanced_data0 = SmoteClassif(Ethnicity ~ ., raw_data, dist = "HEOM", k=2)
534
535 #balanced_data1: bilanciato per Country
536 raw_data = as.data.frame(raw_data)
537 balanced_data = SmoteClassif(Country ~ ., raw_data, dist = "HEOM", k=2)
538 balanced_data1 = NULL
539 for(i in levels(balanced_data$Country)){
540   #faccio il rebalancing di Two_yr_Recidivism per ogni categoria di ethnicity
541   temp_dt = balanced_data[balanced_data$Country==i, ]
542   s = SmoteClassif(Ethnicity ~ ., temp_dt, dist = "HEOM", k=2)
543   #metto poi insieme i vari piccoli dataset bilanciati
544   balanced_data1 = rbind(balanced_data1, s)
545 }
546
547 balanced_data2 = NULL
548 for(i in levels(balanced_data0$Ethnicity)){
549   #faccio il rebalancing di Two_yr_Recidivism per ogni categoria di ethnicity
550   temp_dt = balanced_data0[balanced_data0$Ethnicity==i, ]
551   s = SmoteClassif(Age ~ ., temp_dt, dist = "HEOM")
552   #metto poi insieme i vari piccoli dataset bilanciati
553   balanced_data2 = rbind(balanced_data2, s)
554 }
555
556 cat_data <- balanced_data1[, sapply(balanced_data1, is.categorical)]
557 cat_names <- names(cat_data)
558 for ( i in seq(1,length( cat_data ),1) ){
559   pie_print(cat_data,cat_names[i])
560 }
561
562 ## -----
563 #balanced_data0
564 #split into train and test set
565 sample_size = floor(0.7*nrow(balanced_data0))
566 #set a random seed per rendere l'esperimento ripetibile
567 set.seed(42)
568 # randomly split data
569 picked = sample(seq_len(nrow(balanced_data0)), size = sample_size)
570 training_set = balanced_data0[picked,]
571 test_set = balanced_data0[-picked,]
572
573 #as.factor mi serve per fare una classificazione binaria
574 balanced_data0$Coke = as.factor(balanced_data0$Coke)
575 #devo anche eliminare Coke dal training_set
```

```
576 cols_to_drop <- names(training_set) %in% c("Coke")
577 #training e test set senza variabile target
578 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
579 #anche sui dati di test
580 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
581
582
583 #REGRESSIONE LOGISTICA
584 #elimino colonna crack
585 cols_to_drop <- names(balanced_data0) %in% c("Crack")
586 balanced_data0_copy = balanced_data0
587 balanced_data0 <- balanced_data0[, !cols_to_drop, drop = FALSE]
588
589 #####
590 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
591 logitMod <- glm(Coke ~ ., data = training_set, family=binomial(link="logit"))
592 logitMod #capire cosa significano questi dati
593 #saveRDS(model, "./logitMod_compas_rf.rds") #commento perch l'ho gi salvato
594 #####
595 #logitMod = readRDS(file = "./model_compas_rf.rds")
596 #logitMod
597
598
599 #devo fare queste due righe altrimenti non riconosce i livelli di country
600 logitMod$xlevels[["Country"]] <- union(logitMod$xlevels[["Country"]], levels(raw_data$Country))
601 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
602 balanced0_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
# fairness
603 #classified_dt2 = cbind(test_set_data, pred)
604 #classified_dt = rbind(classified_dt1, classified_dt2)
605 pred1 <- prediction(pred, test_set$Coke) #pred1 un oggetto pred.obj di ROCR
606 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
607 optimized_cutoff0 = opt.cut(perf, pred1)
608
609 ## -----
610 #balanced_data1
611 #split into train and test set
612 sample_size = floor(0.7*nrow(balanced_data1))
613 #set a random seed per rendere l'esperimento ripetibile
614 set.seed(42)
615 # randomly split data
616 picked = sample(seq_len(nrow(balanced_data1)), size = sample_size)
617 training_set = balanced_data1[picked,]
```

```
618 test_set =balanced_data1[-picked,]
619
620 #as.factor mi serve per fare una classificazione binaria
621 balanced_data1$Coke = as.factor(balanced_data1$Coke)
622 #devo anche eliminare Coke dal training_set
623 cols_to_drop <- names(training_set) %in% c("Coke")
624 #training e test set senza variabile target
625 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
626 #anche sui dati di test
627 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
628
629
630 #REGRESSIONE LOGISTICA
631 #elimino colonna crack
632 cols_to_drop <- names(balanced_data1) %in% c("Crack")
633 balanced_data1_copy = balanced_data1
634 balanced_data1 <- balanced_data1[, !cols_to_drop, drop = FALSE]
635
636 #####
637 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
638 logitMod <- glm(Coke ~ ., data = training_set, family=binomial(link="logit"))
639 logitMod #capire cosa significano questi dati
640 #saveRDS(model, "./logitMod_compas_rf.rds") #commento perch l'ho già salvato
641 #####
642 #logitMod = readRDS(file = "./model_compas_rf.rds")
643 #logitMod
644
645 #devo fare queste due righe altrimenti non riconosce i livelli di country
646 logitMod$xlevels[["Country"]] <- union(logitMod$xlevels[["Country"]], levels(raw_data$Country))
647 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
648 balanced1_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
# fairness
649 #classified_dt2 = cbind(test_set_data, pred)
650 #classified_dt = rbind(classified_dt1, classified_dt2)
651 pred1 <- prediction(pred, test_set$Coke) #pred1 un oggetto pred.obj di ROCR
652 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
653 optimized_cutoff1 = opt.cut(perf, pred1)
654
655
656
657 ## -----
658 #balanced_data2
659 #split into train and test set
```

```
660 sample_size = floor(0.7*nrow(balanced_data2))
661 #set a random seed per rendere l'esperimento ripetibile
662 set.seed(42)
663 # randomly split data
664 picked = sample(seq_len(nrow(balanced_data2)), size = sample_size)
665 training_set = balanced_data2[picked,]
666 test_set = balanced_data2[-picked,]
667
668 #as.factor mi serve per fare una classificazione binaria
669 balanced_data2$Coke = as.factor(balanced_data2$Coke)
670 #devo anche eliminare Coke dal training_set
671 cols_to_drop <- names(training_set) %in% c("Coke")
672 #training e test set senza variabile target
673 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
674 #anche sui dati di test
675 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
676
677
678 #REGRESSIONE LOGISTICA
679 #elimino colonna crack
680 cols_to_drop <- names(balanced_data2) %in% c("Crack")
681 balanced_data2_copy = balanced_data2
682 balanced_data2 <- balanced_data2[, !cols_to_drop, drop = FALSE]
683
684 #####
685 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
686 logitMod <- glm(Coke ~ ., data = training_set, family=binomial(link="logit"))
687 logitMod #capire cosa significano questi dati
688 #saveRDS(model, "./logitMod_compas_rf.rds") #commento perch l'ho già salvato
689 #####
690 #logitMod = readRDS(file = "./model_compas_rf.rds")
691 #logitMod
692
693
694 #devo fare queste due righe altrimenti non riconosce i livelli di country
695 logitMod$xlevels[["Country"]] <- union(logitMod$xlevels[["Country"]], levels(raw_data$Country))
696 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perch poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
697 balanced2_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
# fairness
698 #classified_dt2 = cbind(test_set_data, pred)
699 #classified_dt = rbind(classified_dt1, classified_dt2)
700 pred1 <- prediction(pred, test_set$Coke) #pred1 un oggetto pred.obj di ROCR
701 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
```

```

702 optimized_cutoff2 = opt.cut(perf, pred1)
703
704
705
706 ## -----
707 #ora confronto ciascun dataset bilanciato con quello originale
708 #poi seleziono quelli che sembrano effettivamente migliori del dataset originale
709 #poi confronto tra loro questi dataset bilanciati rimanenti e scelgo il migliore
710 levels(balanced0_classified_dt$Coke)[levels(balanced0_classified_dt$Coke)==1] <- "Used"
711 levels(balanced0_classified_dt$Coke)[levels(balanced0_classified_dt$Coke)==0] <- "Never_used"
712 levels(balanced1_classified_dt$Coke)[levels(balanced1_classified_dt$Coke)==1] <- "Used"
713 levels(balanced1_classified_dt$Coke)[levels(balanced1_classified_dt$Coke)==0] <- "Never_used"
714 levels(balanced2_classified_dt$Coke)[levels(balanced2_classified_dt$Coke)==1] <- "Used"
715 levels(balanced2_classified_dt$Coke)[levels(balanced2_classified_dt$Coke)==0] <- "Never_used"
716
717 fair_metrics(data      = balanced2_classified_dt,
718                 outcome    = 'Coke',
719                 group      = 'Ethnicity',
720                 probs     = 'pred',
721                 preds_levels = c('Never_used', 'Used'),
722                 cutoff     = optimized_cutoff,
723                 base       = 'White')
724
725 #creo anche un dataset con i valori delle mean deviation per ogni dataset bilanciato, per
    ogni metrica
726 msd_balanced=NULL
727
728 source("util.R")
729 png("DRUGSbOrdComp.png")
730 p = fair_metrics_cmp_graph(data1      = classified_dt,
731                             data2 = balanced0_classified_dt,
732                             outcome    = 'Coke',
733                             outcome_base = 'Never_used',
734                             group      = 'Ethnicity',
735                             probs1     = 'pred',
736                             probs2     = 'pred',
737                             preds_levels = c('Never_used', 'Used'),
738                             cutoff1    = optimized_cutoff,
739                             cutoff2    = optimized_cutoff0,
740                             base       = 'White')
741 dev.off()
742 #praticamente solo peggioramenti
743 msd_balanced=p[[1]]
744 msd_balanced= rbind(msd_balanced, p[[2]])
745
746 png("DRUGSbirdComp.png")
747 p = fair_metrics_cmp_graph(data1      = classified_dt,
748                             data2 = balanced1_classified_dt,

```

```
749     outcome      = 'Coke',
750     outcome_base = 'Never_used',
751     group        = 'Ethnicity',
752     probs1       = 'pred',
753     probs2       = 'pred',
754     preds_levels = c('Never_used', 'Used'),
755     cutoff1      = optimized_cutoff,
756     cutoff2      = optimized_cutoff1,
757     base         = 'White')

758 dev.off()
759 msd_balanced= rbind(msd_balanced, p[[2]])

760
761 png("DRUGSb2rdComp.png")
762 p = fair_metrics_cmp_graph(data1      = classified_dt,
763                             data2 = balanced2_classified_dt,
764                             outcome      = 'Coke',
765                             outcome_base = 'Never_used',
766                             group        = 'Ethnicity',
767                             probs1       = 'pred',
768                             probs2       = 'pred',
769                             preds_levels = c('Never_used', 'Used'),
770                             cutoff1      = optimized_cutoff,
771                             cutoff2      = optimized_cutoff2,
772                             base         = 'White')

773 dev.off()
774 msd_balanced= rbind(msd_balanced, p[[2]])

775
776
777
778
779 #CONFRONTO TRA LORO BALANCED DATASETS
780 #####
781 png("DRUGSb0b1Comp.png")
782 fair_metrics_cmp_graph(data1      = balanced0_classified_dt,
783                         data2 = balanced1_classified_dt,
784                         outcome      = 'Coke',
785                         outcome_base = 'Never_used',
786                         group        = 'Ethnicity',
787                         probs1       = 'pred',
788                         probs2       = 'pred',
789                         preds_levels = c('Never_used', 'Used'),
790                         cutoff1      = optimized_cutoff0,
791                         cutoff2      = optimized_cutoff1,
792                         base         = 'White')

793 dev.off()
794
795 png("DRUGSb0b2Comp.png")
796 fair_metrics_cmp_graph(data1      = balanced0_classified_dt,
```

```

797     data2 = balanced2_classified_dt,
798     outcome      = 'Coke',
799     outcome_base = 'Never_used',
800     group        = 'Ethnicity',
801     probs1       = 'pred',
802     probs2       = 'pred',
803     preds_levels = c('Never_used', 'Used'),
804     cutoff1      = optimized_cutoff0,
805     cutoff2      = optimized_cutoff2,
806     base         = 'White')
807 dev.off()
808
809
810
811 msd_balanced
812 long_msd_balanced <- msd_balanced %>% gather(metrics, values, dem_parity:spec_parity)
813
814
815 g = ggplot(long_msd_balanced, aes(fill=metrics, y=values, x=datasets)) +
816   geom_bar(position="dodge", stat="identity") +
817   #ggtitle("Studying 4 species..") +
818   facet_wrap(~metrics, ncol=2) +
819   #facet_grid(metrics ~ datasets)
820   theme(legend.position="none",
821         strip.text = element_text(face="bold", size=15, lineheight=5.0),
822         strip.background = element_rect(fill="lightblue", colour="black",
823                                         size=1), axis.text.x = element_text(angle = 90),
824         axis.text=element_text(size=20, face="bold"),
825         axis.title=element_text(size=20, face="bold")) +
826   xlab("")
827 ggsave("OverallCoke.jpg", plot = g, width = 24, height = 33, units = "in")
828
829 ## -----
830 raw_data = raw_data_copy
831 #sostituisco i livelli con valori numerici
832 raw_data$Crack = as.factor(raw_data$Crack)
833 levels(raw_data$Crack)[levels(raw_data$Crack)=="Used"] <- 1
834 levels(raw_data$Crack)[levels(raw_data$Crack)=="Never_used"] <- 0
835
836 #split into train and test set
837 sample_size = floor(0.7*nrow(raw_data))
838 #set a random seed per rendere l'esperimento ripetibile
839 set.seed(42)
840 # randomly split data
841 picked = sample(seq_len(nrow(raw_data)), size = sample_size)
842 training_set = raw_data[picked,]
843 test_set = raw_data[-picked,]
844

```

```

845 #as.factor mi serve per fare una classificazione binaria
846 raw_data$Crack = as.factor(raw_data$Crack)
847 #devo anche eliminare Crack dal training_set
848 cols_to_drop <- names(training_set) %in% c("Crack")
849 #training e test set senza variabile target
850 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
851 #anche sui dati di test
852 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
853
854 ## -----
855 #REGRESSIONE LOGISTICA
856 #elimino colonna coke
857 cols_to_drop <- names(raw_data) %in% c("Coke")
858 raw_data_copy = raw_data
859 raw_data <- raw_data[, !cols_to_drop, drop = FALSE]
860
861 #####
862 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
#       sufficiente salvarlo e ricaricarlo)
863 logitMod <- glm(Crack ~ ., data = training_set, family=binomial(link="logit"))
864 logitMod #capire cosa significano questi dati
865 #saveRDS(model, "./logitMod_compas_rf.rds") #commento perch l'ho già salvato
866 #####
867 #logitMod = readRDS(file = "./model_compas_rf.rds")
868 #logitMod
869
870
871 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
#               mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
#               tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
#               fare la predizione
872 classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria fairness
873 classified_dt2 = cbind(test_set_data, pred)
874 #classified_dt = rbind(classified_dt1, classified_dt2)
875
876
877 ## -----
878 #ROC
879 pred1 <- prediction(pred, test_set$Crack) #pred1 un oggetto pred.obj di ROCR
880 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
881 plot(perf)
882 #posso anche fare una ROC comparison per le categorie di un attributo sensibile, tipo Female,
#     con la libreria fairness:
883 roc_fairness <- roc_parity(data      = classified_dt,
884                               outcome    = 'Crack',
885                               group      = 'Gender',
886                               probs      = 'pred',
887                               preds_levels = c('Never_used', 'Used'),

```

```
888         base      = 'Male')
889 roc_fairness$Metric
890 roc_fairness$ROCAUC_plot
891
892 #AUC
893 auc.perf = performance(pred1, measure = "auc")
894 auc.perf@y.values #stampa a video il valore di AUC
895 #####
896 #plotto sensitivity e specificity in modo da individuare il cutoff ottimizzato
897 sens <- data.frame(x=unlist(performance(pred1, "sens")@x.values),
898                      y=unlist(performance(pred1, "sens")@y.values))
899 spec <- data.frame(x=unlist(performance(pred1, "spec")@x.values),
900                      y=unlist(performance(pred1, "spec")@y.values))
901
902 sens %>% ggplot(aes(x,y)) +
903   geom_line() +
904   geom_line(data=spec, aes(x,y,col="red")) +
905   scale_y_continuous(sec.axis = sec_axis(~ ., name = "Specificity")) +
906   labs(x='Cutoff', y="Sensitivity") +
907   theme(axis.title.y.right = element_text(colour = "red"), legend.position="none")
908 #####
909
910 ## -----
911 #troviamo l'optimal cutoff, cio l'optimal score che minimizza il misclassification error del
#modello di regressione logistica
912 #The best threshold (or cutoff) point to be used in glm models is the point which maximises
#the specificity and the sensitivity. This threshold point might not give the highest
#prediction in your model, but it wouldn't be biased towards positives or negatives.
913 optimized_cutoff = opt.cut(perf, pred1)#il cutoff ottimale leggermente piu basso di quello
#standard, che sarebbe invece 0.5
914 optimized_cutoff
915
916 #convertendo la colonna pred di classified_dt2 in yes/no recidivism usando il cutoff ottimizzato
917 #yes era 1, quindi ho recidivismo se maggiore del cutoff
918 classified_dt2$pred = ifelse(classified_dt2$pred<optimized_cutoff,"Never_used","Used")
919 #sostituisco i valori dei livelli per poter plottare la matrice di confusione
920 levels(test_set$Crack)[levels(test_set$Crack)==1] <- "Used"
921 levels(test_set$Crack)[levels(test_set$Crack)==0] <- "Never_used"
922 levels(test_set$Crack)
923 #anche per classified_dt2
924 classified_dt2$pred <- factor(classified_dt2$pred, levels = c("Used", "Never_used"))
925
926 ## -----
927 #matrice di confusione
928 cm <- confusionMatrix(data = as.factor(classified_dt2$pred), reference = as.factor(test_set$Crack))
929 #plotto la matrice di confusione
930 png("DRUGSCrackCM.png")
```

```

931 draw_confusion_matrix(cm)
932 dev.off()
933
934 ## -----
935 #ci occupiamo di group fairness: indipendenza, separazione e sufficienza
936
937 #lo facciamo per tutti i sensitive attribute (Marital, ageR)
938
939 #ripristino di nuovo i livelli yes/no per renderlo pi leggibile
940 levels(classified_dt$Crack)[levels(classified_dt$Crack)==1] <- "Used"
941 levels(classified_dt$Crack)[levels(classified_dt$Crack)==0] <- "Never_used"
942
943 #per scegliere la categoria "base" per cui confrontare tutte le altre, per ogni attributo
# sensibile, ci conviene guardare le tabelle delle frequenze plottate in precedenza:
944
945
946 ## -----
947 source("util.R")
948 levels(classified_dt$Crack) #se non metto un outcome_base, la libreria fairness prende il
# primo di questi valori come valore positivo ---- qui va bene perch il primo never used
949 fair_metrics(data      = classified_dt,
950                 outcome    = 'Crack',
951                 group      = 'Age',
952                 probs     = 'pred',
953                 preds_levels = c('Never_used', 'Used'),
954                 cutoff     = optimized_cutoff,
955                 base       = '18_24')
956
957 fair_metrics(data      = classified_dt,
958                 outcome    = 'Crack',
959                 group      = 'Ethnicity',
960                 probs     = 'pred',
961                 preds_levels = c('Never_used', 'Used'),
962                 cutoff     = optimized_cutoff,
963                 base       = 'White')
964
965 fair_metrics(data      = classified_dt,
966                 outcome    = 'Crack',
967                 group      = 'Gender',
968                 probs     = 'pred',
969                 preds_levels = c('Never_used', 'Used'),
970                 cutoff     = optimized_cutoff,
971                 base       = 'Male')
972 fair_metrics_cmp_graph2(data      = classified_dt,
973                 outcome    = 'Crack',
974                 group1     = 'Gender',
975                 group2 = 'Ethnicity',
976                 probs     = 'pred',

```

```

977             preds_levels = c('Never_used', 'Used'),
978             cutoff      = optimized_cutoff,
979             base1       = 'Male',
980             base2       = 'White')
981 fair_metrics_cmp_graph2(data      = classified_dt,
982                         outcome   = 'Crack',
983                         group1    = 'Age',
984                         group2 = 'Ethnicity',
985                         probs     = 'pred',
986                         preds_levels = c('Never_used', 'Used'),
987                         cutoff      = optimized_cutoff,
988                         base1       = '18_24',
989                         base2       = 'White')
990 fair_metrics_cmp_graph2(data      = classified_dt,
991                         outcome   = 'Crack',
992                         group1    = 'Ethnicity',
993                         group2 = 'Age',
994                         probs     = 'pred',
995                         preds_levels = c('Never_used', 'Used'),
996                         cutoff      = optimized_cutoff,
997                         base1       = 'White',
998                         base2       = '18_24')
999 fair_metrics_cmp_graph2(data      = classified_dt,
1000                        outcome   = 'Crack',
1001                        group1    = 'Gender',
1002                        group2 = 'Age',
1003                        probs     = 'pred',
1004                        preds_levels = c('Never_used', 'Used'),
1005                        cutoff      = optimized_cutoff,
1006                        base1       = 'Male',
1007                        base2       = '18_24')
1008
1009
1010 ## -----
1011 #CREO I SEGUENTI DATASET BILANCIATI PER CRACK:
1012 #balanced_data0: bilanciato per Crack
1013 #balanced_data1: bilanciato per Crack and Ethnicity
1014 #balanced_data2: bilanciato per Crack, Ethnicity and Age
1015
1016 #balanced_data0: bilanciato per Ethnicity
1017 #questo invece ha ethnicity bilanciato con smote ed meglio usare questo!
1018 raw_data = raw_data_copy
1019 raw_data = as.data.frame(raw_data)
1020 balanced_data0 = ovun.sample(Crack~, data = raw_data, method="both", N = nrow(raw_data), p
=0.5, seed = 42)$data
1021 #balanced_data0 = SmoteClassif(Age ~ ., balanced_data0, dist = "HEOM")
1022
1023

```

```
1024 #balanced_data1 = SmoteClassif(Country ~ ., raw_data, dist = "HEOM", k=2)
1025 balanced_data1 = SmoteClassif(Ethnicity ~ ., balanced_data0, dist = "HEOM", k=2)
1026
1027
1028 balanced_data2 = NULL
1029 balanced_data2 = SmoteClassif(Age ~ ., balanced_data0, dist = "HEOM", k=2)
1030 # for(i in levels(balanced_data0$Ethnicity)){
1031 #   #faccio il rebalancing di Two_yr_Recidivism per ogni categoria di ethnicity
1032 #   temp_dt = balanced_data0[balanced_data0$Ethnicity==i, ]
1033 #   s = SmoteClassif(Age ~ ., temp_dt, dist = "HEOM")
1034 #   #metto poi insieme i vari piccoli dataset bilanciati
1035 #   balanced_data2 = rbind(balanced_data2, s)
1036 #
1037
1038 cat_data <- balanced_data0[, sapply(balanced_data0, is.categorical)]
1039 cat_names <- names(cat_data)
1040 for ( i in seq(1,length( cat_data ),1) ){
1041   pie_print(cat_data,cat_names[i])
1042 }
1043
1044 ## -----
1045 #balanced_data0
1046 #split into train and test set
1047 sample_size = floor(0.7*nrow(balanced_data0))
1048 #set a random seed per rendere l'esperimento ripetibile
1049 set.seed(42)
1050 # randomly split data
1051 picked = sample(seq_len(nrow(balanced_data0)),size = sample_size)
1052 training_set =balanced_data0[picked,]
1053 test_set =balanced_data0[-picked,]
1054
1055 #as.factor mi serve per fare una classificazione binaria
1056 balanced_data0$Crack = as.factor(balanced_data0$Crack)
1057 #devo anche eliminare Coke dal training_set
1058 cols_to_drop <- names(training_set) %in% c("Crack")
1059 #training e test set senza variabile target
1060 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
1061 #anche sui dati di test
1062 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
1063
1064
1065 #REGRESSIONE LOGISTICA
1066 #elimino colonna crack
1067 cols_to_drop <- names(balanced_data0) %in% c("Crack")
1068 balanced_data0_copy = balanced_data0
1069 balanced_data0 <- balanced_data0[, !cols_to_drop, drop = FALSE]
1070 #####
1071 #####
```

```
1072 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo ( 
1073   sufficiente salvarlo e ricaricarlo)
1074 logitMod <- glm(Crack ~ ., data = training_set, family=binomial(link="logit"))
1075 logitMod #capire cosa significano questi dati
1076 #saveRDS(model, "./logitMod_compas_rf.rds") #commento perch l'ho già salvato
1077 ##########
1078 #logitMod = readRDS(file = "./model_compas_rf.rds")
1079 #logitMod
1080
1081 #devo fare queste due righe altrimenti non riconosce i livelli di country
1082 logitMod$xlevels[["Country"]] <- union(logitMod$xlevels[["Country"]], levels(raw_data$Country))
1083 logitMod$xlevels[["Ethnicity"]] <- union(logitMod$xlevels[["Ethnicity"]], levels(raw_data$Ethnicity))
1084 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
1085   mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
1086   tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
1087   fare la predizione
1088 balanced0_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
1089   fairness
1090
1091
1092
1093 ## -----
1094 #balanced_data1
1095 #split into train and test set
1096 sample_size = floor(0.7*nrow(balanced_data1))
1097 #set a random seed per rendere l'esperimento ripetibile
1098 set.seed(42)
1099 # randomly split data
1100 picked = sample(seq_len(nrow(balanced_data1)), size = sample_size)
1101 training_set = balanced_data1[picked,]
1102 test_set = balanced_data1[-picked,]
1103
1104 #as.factor mi serve per fare una classificazione binaria
1105 balanced_data1$Crack = as.factor(balanced_data1$Crack)
1106 #devo anche eliminare Crack dal training_set
1107 cols_to_drop <- names(training_set) %in% c("Crack")
1108 #training e test set senza variabile target
1109 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
1110 #anche sui dati di test
1111 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
1112
```

```
1113 
1114 #REGRESSIONE LOGISTICA
1115 #elimino colonna crack
1116 cols_to_drop <- names(balanced_data1) %in% c("Coke")
1117 balanced_data1_copy = balanced_data1
1118 balanced_data1 <- balanced_data1[, !cols_to_drop, drop = FALSE]
1119 
1120 #####
1121 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
1122 logitMod <- glm(Crack ~ ., data = training_set, family=binomial(link="logit"))
1123 logitMod #capire cosa significano questi dati
1124 #saveRDS(model, "./logitMod_compas_rf.rds") #commento perch l'ho già salvato
1125 #####
1126 #logitMod = readRDS(file = "./model_compas_rf.rds")
1127 #logitMod
1128 
1129 #devo fare queste due righe altrimenti non riconosce i livelli di country
1130 logitMod$xlevels[["Country"]] <- union(logitMod$xlevels[["Country"]], levels(raw_data$Country))
1131 logitMod$xlevels[["Age"]] <- union(logitMod$xlevels[["Age"]], levels(raw_data$Age))
1132 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perch poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
1133 balanced1_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
# fairness
1134 #classified_dt2 = cbind(test_set_data, pred)
1135 #classified_dt = rbind(classified_dt1, classified_dt2)
1136 pred1 <- prediction(pred, test_set$Crack) #pred1 un oggetto pred.obj di ROCR
1137 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
1138 optimized_cutoff1 = opt.cut(perf, pred1)
1139 
1140 
1141 
1142 ## -----
1143 #balanced_data2
1144 #split into train and test set
1145 sample_size = floor(0.7*nrow(balanced_data2))
1146 #set a random seed per rendere l'esperimento ripetibile
1147 set.seed(42)
1148 # randomly split data
1149 picked = sample(seq_len(nrow(balanced_data2)), size = sample_size)
1150 training_set = balanced_data2[picked,]
1151 test_set = balanced_data2[-picked,]
1152 
1153 #as.factor mi serve per fare una classificazione binaria
1154 balanced_data2$Crack = as.factor(balanced_data2$Crack)
```

```

1155 #devo anche eliminare Coke dal training_set
1156 cols_to_drop <- names(training_set) %in% c("Crack")
1157 #training e test set senza variabile target
1158 training_set_data <- training_set[, !cols_to_drop, drop = FALSE]
1159 #anche sui dati di test
1160 test_set_data <- test_set[, !cols_to_drop, drop = FALSE]
1161
1162
1163 #REGRESSIONE LOGISTICA
1164 #elimino colonna crack
1165 cols_to_drop <- names(balanced_data2) %in% c("Coke")
1166 balanced_data2_copy = balanced_data2
1167 balanced_data2 <- balanced_data2[, !cols_to_drop, drop = FALSE]
1168
1169 #####
1170 #poi commento le righe in cui calcola il modello perch ci potrebbe mettere troppo (
# sufficiente salvarlo e ricaricarlo)
1171 logitMod <- glm(Crack ~ ., data = training_set, family=binomial(link="logit"))
1172 logitMod #capire cosa significano questi dati
1173 #saveRDS(model, "./logitMod_compas_rf.rds") #commento perch l'ho già salvato
1174 #####
1175 #logitMod = readRDS(file = "./model_compas_rf.rds")
1176 #logitMod
1177
1178 #devo fare queste due righe altrimenti non riconosce i livelli di country
1179 logitMod$xlevels[["Country"]] <- union(logitMod$xlevels[["Country"]], levels(raw_data$Country))
1180 pred = predict(logitMod, test_set_data, type = "response") #se non metto il type = response,
# mi restituisce una predizione tra -1 e 1, altrimenti me la restituisce tra 0 e 1. Scelgo
# tra -1 e 1 perc poi ottengo il cutoff ottimizzato anch'esso tra -1 e 1 e mi pi facile
# fare la predizione
1181 balanced2_classified_dt = cbind(test_set, pred) #questo quello che uso per la libreria
# fairness
1182 #classified_dt2 = cbind(test_set_data, pred)
1183 #classified_dt = rbind(classified_dt1, classified_dt2)
1184 pred1 <- prediction(pred, test_set$Crack) #pred1 un oggetto pred.obj di ROCR
1185 perf <- performance(pred1, measure = "tpr", x.measure = "fpr")
1186 optimized_cutoff2 = opt.cut(perf, pred1)
1187
1188 ## -----
1189 summary(balanced1_classified_dt)
1190
1191
1192 ## -----
1193 #ora confronto ciascun dataset bilanciato con quello originale
1194 #poi seleziono quelli che sembrano effettivamente migliori del dataset originale
1195 #poi confronto tra loro questi dataset bilanciati rimanenti e scelgo il migliore
1196 levels(balanced0_classified_dt$Crack)[levels(balanced0_classified_dt$Crack)==1] <- "Used"

```

```
1197 levels(balanced0_classified_dt$Crack)[levels(balanced0_classified_dt$Crack)==0] <- "Never_
    used"
1198 levels(balanced1_classified_dt$Crack)[levels(balanced1_classified_dt$Crack)==1] <- "Used"
1199 levels(balanced1_classified_dt$Crack)[levels(balanced1_classified_dt$Crack)==0] <- "Never_
    used"
1200 levels(balanced2_classified_dt$Crack)[levels(balanced2_classified_dt$Crack)==1] <- "Used"
1201 levels(balanced2_classified_dt$Crack)[levels(balanced2_classified_dt$Crack)==0] <- "Never_
    used"
1202
1203 fair_metrics(data      = balanced1_classified_dt,
1204                 outcome     = 'Crack',
1205                 group       = 'Ethnicity',
1206                 probs       = 'pred',
1207                 preds_levels = c('Never_used', 'Used'),
1208                 cutoff      = optimized_cutoff1,
1209                 base        = 'White')
1210
1211 #creo anche un dataset con i valori delle mean deviation per ogni dataset bilanciato, per
    ogn
1212 msd_balanced=NULL
1213
1214 source("util.R")
1215 png("DRUGSb0rdComp.png")
1216 p = fair_metrics_cmp_graph(data1      = classified_dt,
1217                             data2 = balanced0_classified_dt,
1218                             outcome     = 'Crack',
1219                             outcome_base = 'Never_used',
1220                             group       = 'Ethnicity',
1221                             probs1     = 'pred',
1222                             probs2     = 'pred',
1223                             preds_levels = c('Never_used', 'Used'),
1224                             cutoff1    = optimized_cutoff,
1225                             cutoff2    = optimized_cutoff0,
1226                             base        = 'White')
1227 dev.off()
1228 #praticamente solo peggioramenti
1229 msd_balanced=p[[1]]
1230 msd_balanced= rbind(msd_balanced, p[[2]])
1231
1232 png("DRUGSb1rdComp.png")
1233 p = fair_metrics_cmp_graph(data1      = classified_dt,
1234                             data2 = balanced1_classified_dt,
1235                             outcome     = 'Crack',
1236                             outcome_base = 'Never_used',
1237                             group       = 'Ethnicity',
1238                             probs1     = 'pred',
1239                             probs2     = 'pred',
1240                             preds_levels = c('Never_used', 'Used'),
```

```
1241         cutoff1      = optimized_cutoff,
1242         cutoff2      = optimized_cutoff1,
1243         base        = 'White')
1244 dev.off()
1245 msd_balanced= rbind(msd_balanced, p[[2]])
1246
1247 png("DRUGSb2rdComp.png")
1248 p = fair_metrics_cmp_graph(data1           = classified_dt,
1249                             data2 = balanced2_classified_dt,
1250                             outcome       = 'Crack',
1251                             outcome_base = 'Never_used',
1252                             group        = 'Ethnicity',
1253                             probs1       = 'pred',
1254                             probs2       = 'pred',
1255                             preds_levels = c('Never_used', 'Used'),
1256                             cutoff1      = optimized_cutoff,
1257                             cutoff2      = optimized_cutoff2,
1258                             base        = 'White')
1259 dev.off()
1260 msd_balanced= rbind(msd_balanced, p[[2]])
1261
1262 #CONFRONTO TRA LORO BALANCED DATASETS
1263 #####
1264 png("DRUGSbOb1Comp.png")
1265 fair_metrics_cmp_graph(data1           = balanced0_classified_dt,
1266                           data2 = balanced1_classified_dt,
1267                           outcome       = 'Crack',
1268                           outcome_base = 'Never_used',
1269                           group        = 'Ethnicity',
1270                           probs1       = 'pred',
1271                           probs2       = 'pred',
1272                           preds_levels = c('Never_used', 'Used'),
1273                           cutoff1      = optimized_cutoff0,
1274                           cutoff2      = optimized_cutoff1,
1275                           base        = 'White')
1276 dev.off()
1277
1278 png("DRUGSbOb2Comp.png")
1279 fair_metrics_cmp_graph(data1           = balanced0_classified_dt,
1280                           data2 = balanced2_classified_dt,
1281                           outcome       = 'Crack',
1282                           outcome_base = 'Never_used',
1283                           group        = 'Ethnicity',
1284                           probs1       = 'pred',
1285                           probs2       = 'pred',
1286                           preds_levels = c('Never_used', 'Used'),
1287                           cutoff1      = optimized_cutoff0,
1288                           cutoff2      = optimized_cutoff2,
```

```
1289           base      = 'White')
1290 dev.off()
1291
1292 msd_balanced
1293 long_msd_balanced <- msd_balanced %>% gather(metrics, values, dem_parity:spec_parity)
1294
1295
1296 g = ggplot(long_msd_balanced, aes(fill=metrics, y=values, x=datasets)) +
1297   geom_bar(position="dodge", stat="identity") +
1298   #ggtitle("Studying 4 species..") +
1299   facet_wrap(~metrics, ncol=2) +
1300   #facet_grid(metrics ~ datasets)
1301   theme(legend.position="none",
1302         strip.text = element_text(face="bold", size=15, lineheight=5.0),
1303         strip.background = element_rect(fill="lightblue", colour="black",
1304         size=1), axis.text.x = element_text(angle = 90),
1305         axis.text=element_text(size=20, face="bold"),
1306         axis.title=element_text(size=20, face="bold")) +
1307         xlab(""))
1308 ggsave("OverallCrack.jpg", plot = g, width = 24, height = 33, units = "in")
```


Data files

COMPAS fairness metrics data

| | between25_45 | above45 | below25 |
|------------------------|--------------|------------|-------------|
| Positively classified | 460 | 71.0000000 | 365.0000000 |
| Demographic Parity | 1 | 0.1543478 | 0.7934783 |
| | | | |
| | between25_45 | above45 | below25 |
| Proportion | 0.4283054 | 0.1863517 | 0.9193955 |
| Proportional Parity | 1.0000000 | 0.4350907 | 2.1465885 |
| | | | |
| | between25_45 | above45 | below25 |
| Sensitivity | 0.7244224 | 0.9031008 | 0.1294118 |
| Equalized odds | 1.0000000 | 1.2466494 | 0.1786413 |
| | | | |
| | between25_45 | above45 | below25 |
| Precision | 0.7149837 | 0.7516129 | 0.6875000 |
| Predictive Rate Parity | 1.0000000 | 1.0512308 | 0.9615604 |
| | | | |
| | between25_45 | above45 | below25 |
| Accuracy | 0.6815642 | 0.7322835 | 0.6020151 |
| Accuracy Parity | 1.0000000 | 1.0744159 | 0.8832845 |
| | | | |
| | between25_45 | above45 | below25 |
| FNR | 0.2755776 | 0.09689922 | 0.8705882 |
| FNR Parity | 1.0000000 | 0.35162234 | 3.1591405 |
| | | | |
| | between25_45 | above45 | below25 |
| FPR | 0.3739316 | 0.6260163 | 0.04405286 |
| FPR Parity | 1.0000000 | 1.6741463 | 0.11780994 |
| | | | |
| | between25_45 | above45 | below25 |
| MCC | 0.3512148 | 0.3327091 | 0.1551546 |
| MCC Parity | 1.0000000 | 0.9473094 | 0.4417657 |

Appendix B. Data files

| | | | | | |
|------------------------|---|------------|---------------------|--------------------|-----------|
| | between25_45 above45 below25 | | | | |
| NPV | 0.6369565 | 0.6478873 | 0.5945205 | | |
| NPV Parity | 1.0000000 | 1.0171610 | 0.9333770 | | |
| | between25_45 above45 below25 | | | | |
| Specificity | 0.6260684 | 0.3739837 | 0.9559471 | | |
| Specificity Parity | 1.0000000 | 0.5973529 | 1.5269053 | | |
| | Caucasian African_American Asian Hispanic | | | | |
| Positively classified | 214 | 602.000000 | 0 54.0000000 | | |
| Demographic Parity | 1 | 2.813084 | 0 0.2523364 | | |
| | Native_American Other | | | | |
| Positively classified | 1.000000000 | 25.0000000 | | | |
| Demographic Parity | 0.004672897 | 0.1168224 | | | |
| | Caucasian African_American Asian Hispanic Native_American Other | | | | |
| Proportion | 0.3328149 | 0.6438503 | 0 0.3253012 | 0.3333333 | 0.2551020 |
| Proportional Parity | 1.0000000 | 1.9345594 | 0 0.9774237 | 1.0015576 | 0.7664982 |
| | Caucasian African_American Asian Hispanic Native_American Other | | | | |
| Sensitivity | 0.7608142 | 0.5314534 | 1.000000 0.8073394 | 1.000000 | 0.859375 |
| Equalized odds | 1.0000000 | 0.6985323 | 1.314381 1.0611519 | 1.314381 | 1.129546 |
| | Caucasian African_American Asian Hispanic Native_American Other | | | | |
| Precision | 0.6969697 | 0.7357357 | 0.8571429 0.7857143 | 0.5000000 | 0.7534247 |
| Predictive Rate Parity | 1.0000000 | 1.0556208 | 1.2298137 1.1273292 | 0.7173913 | 1.0810006 |
| | Caucasian African_American Asian Hispanic Native_American Other | | | | |
| Accuracy | 0.651633 | 0.6748663 | 0.8571429 0.7289157 | 0.6666667 | 0.7244898 |
| Accuracy Parity | 1.000000 | 1.0356540 | 1.3153767 1.1185985 | 1.0230708 | 1.1118065 |
| | Caucasian African_American Asian Hispanic Native_American Other | | | | |
| FNR | 0.2391858 | 0.4685466 | 0 0.1926606 | 0 0.1406250 | |
| FNR Parity | 1.0000000 | 1.9589237 | 0 0.8054851 | 0 0.5879322 | |
| | Caucasian African_American Asian Hispanic Native_American Other | | | | |
| FPR | 0.52 | 0.1856540 | 1.000000 0.4210526 | 0.5000000 | 0.5294118 |
| FPR Parity | 1.00 | 0.3570269 | 1.923077 0.8097166 | 0.9615385 | 1.0180995 |
| | Caucasian African_American Asian Hispanic Native_American Other | | | | |
| MCC | 0.2491225 | 0.3610301 | 0 0.3915207 | 0.500000 0.3602994 | |
| MCC Parity | 1.0000000 | 1.4492073 | 0 1.5715990 | 2.007045 1.4462742 | |
| | Caucasian African_American Asian Hispanic Native_American Other | | | | |
| NPV | 0.5607477 | 0.641196 | NaN 0.6111111 | 1.000000 0.640000 | |
| NPV Parity | 1.0000000 | 1.143466 | NaN 1.0898148 | 1.783333 1.141333 | |
| | Caucasian African_American Asian Hispanic Native_American Other | | | | |
| Specificity | 0.48 | 0.814346 | 0 0.5789474 | 0.500000 0.4705882 | |

| | | | | | | |
|------------------------|-----------|-----------|---|-----------|----------|-----------|
| Specificity Parity | 1.00 | 1.696554 | 0 | 1.2061404 | 1.041667 | 0.9803922 |
| | Male | Female | | | | |
| Positively classified | 802 | 94.000000 | | | | |
| Demographic Parity | 1 | 0.117207 | | | | |
| | Male | Female | | | | |
| Proportion | 0.5357381 | 0.2647887 | | | | |
| Proportional Parity | 1.0000000 | 0.4942503 | | | | |
| | Male | Female | | | | |
| Sensitivity | 0.6231343 | 0.8391304 | | | | |
| Equalized odds | 1.0000000 | 1.3466285 | | | | |
| | Male | Female | | | | |
| Precision | 0.7208633 | 0.7394636 | | | | |
| Predictive Rate Parity | 1.0000000 | 1.0258028 | | | | |
| | Male | Female | | | | |
| Accuracy | 0.6680027 | 0.7042254 | | | | |
| Accuracy Parity | 1.0000000 | 1.0542254 | | | | |
| | Male | Female | | | | |
| FNR | 0.3768657 | 0.1608696 | | | | |
| FNR Parity | 1.0000000 | 0.4268618 | | | | |
| | Male | Female | | | | |
| FPR | 0.2799423 | 0.544000 | | | | |
| FPR Parity | 1.0000000 | 1.943258 | | | | |
| | Male | Female | | | | |
| MCC | 0.3431249 | 0.3194837 | | | | |
| MCC Parity | 1.0000000 | 0.9311003 | | | | |
| | Male | Female | | | | |
| NPV | 0.6221945 | 0.6063830 | | | | |
| NPV Parity | 1.0000000 | 0.9745875 | | | | |
| | Male | Female | | | | |
| Specificity | 0.7200577 | 0.4560000 | | | | |
| Specificity Parity | 1.0000000 | 0.6332826 | | | | |

References

- [1] 2018 reform of eu data protection rules. https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf.
- [2] Codice in materia di protezione dei dati personali. https://www.agid.gov.it/sites/default/files/repository_files/leggi_decreti_direttive/decreto_legislativo_196_2003.pdf.
- [3] Part 1607 - uniform guidelines on employee selection procedures. <https://www.govinfo.gov/content/pkg/CFR-2011-title29-vol4/xml/CFR-2011-title29-vol4-part1607.xml>.
- [4] Wikipedia:facebook-cambridge analytica data scandal. https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal.
- [5] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>.
- [6] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. <http://www.fairmlbook.org>.
- [7] Solon Barocas and Andrew D. Selbst. Big Data's Disparate Impact. *SSRN eLibrary*, 2014.
- [8] Jonathan Baron and John Hershey. Outcome bias in decision evaluation. *Journal of personality and social psychology*, 54:569–79, 05 1988.
- [9] Gustavo Batista, Ronaldo Prati, and Maria-Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6:20–29, 06 2004.
- [10] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, 0(0):0049124118782533, 0.
- [11] Colin R. Blyth. On simpson's paradox and the sure-thing principle. *Journal of the American Statistical Association*, 67(338):364–366, 1972.
- [12] Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.
- [13] Joymallya Chakraborty, Suvodeep Majumder, Zhe Yu, and Tim Menzies. Fairway: a way to build fair ml software. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Nov 2020.

- [14] Le Chen, Ruijun Ma, Anikó Hannák, and Christo Wilson. Investigating the impact of gender on rank in resume search engines. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18*, New York, NY, USA, 2018. Association for Computing Machinery.
- [15] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21, 12 2020.
- [16] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. volume 06, 06 2006.
- [17] Thomas G Dietterich and Eun Bae Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Technical report, Department of Computer Science, Oregon State University, 1995.
- [18] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Rich Zemel. Fairness through awareness. 2011.
- [19] Vincent Egan, Ian Deary, and Elizabeth Austin. The neo-ffi: Emerging british norms and an item-level analysis suggest n, a and c are more reliable than o and e. *Personality and Individual Differences*, 29:907–920, 11 2000.
- [20] Marc Elliott, Allen Fremont, Peter Morrison, Philip Pantoja, and Nicole Lurie. A new method for estimating race/ethnicity and associated disparities where administrative records lack self-reported race/ethnicity. *Health services research*, 43:1722–36, 05 2008.
- [21] Robert Epstein and Ronald E. Robertson. The search engine manipulation effect (seme) and its possible impact on the outcomes of elections. *Proceedings of the National Academy of Sciences*, 112(33):E4512–E4521, 2015.
- [22] E. Fehrman, A. K. Muhammad, E. M. Mirkes, V. Egan, and A. N. Gorban. The five factor model of personality and evaluation of drug consumption risk, 2017.
- [23] Pratik Gajane and Mykola Pechenizkiy. On formalizing fairness in prediction with machine learning. 2018.
- [24] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 01 1992.
- [25] Farrokh Habibzadeh, Parham Habibzadeh, and Mahboobeh Yadollahie. On determining the most appropriate test cut-off value: The case of tests with continuous results. *Biochimia Medica*, 26:297–307, 10 2016.
- [26] Anikó Hannák, Claudia Wagner, David Garcia, Alan Mislove, Markus Strohmaier, and Christo Wilson. Bias in online freelance marketplaces: Evidence from taskrabbit and fiverr. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW '17*, page 1914–1933, New York, NY, USA, 2017. Association for Computing Machinery.
- [27] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. *CoRR*, abs/1610.02413, 2016.
- [28] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.
- [29] Alex Hern. Twitter apologises for 'racist' image-cropping algorithm, 2020. Available on line.
- [30] Feng Hu and Hang Li. A novel boundary oversampling algorithm based on neighborhood rough set model: Nrsboundary-smote. *Mathematical Problems in Engineering*, 2013, 11 2013.

- [31] Clive Humby. *British mathematician and entrepreneur*. 2006.
- [32] Jevan A. Hutson, Jessie G. Taft, Solon Barocas, and Karen Levy. Debiasing desire: Addressing bias discrimination on intimate platforms. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW), November 2018.
- [33] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6:429–449, 11 2002.
- [34] Lauren Kirchner Julia Angwin Jeff Larson, Surya Mattu. Propublica/compas-analysis. <https://github.com/propublica/compas-analysis>, 2016.
- [35] Justin Johnson and Taghi Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6:27, 03 2019.
- [36] Faysal Kamiran, Indre Zliobaite, and Toon Calders. Quantifying explainable discrimination and removing illegal discrimination in automated decision making. *Knowledge and Information Systems*, 35(3):613–644, 2013.
- [37] Nicolas Kayser-Bril. Female historians and male nurses do not exist, google translate tells its european users, 2020. Available on line.
- [38] Niki Kilbertus, Mateo Rojas-Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. Avoiding discrimination through causal reasoning. 2018.
- [39] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. 2018.
- [40] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine*, AIME '01, page 63–66, Berlin, Heidelberg, 2001. Springer-Verlag.
- [41] X. Liu and Z. Zhou. The influence of class imbalance on cost-sensitive learning: An empirical study. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 970–974, 2006.
- [42] Kirsten Lloyd. Bias amplification in artificial intelligence systems. 09 2018.
- [43] Robert McCrae and Paul Costa. A contemplated revision of the neo five-factor inventory. *Personality and Individual Differences*, 36:587–596, 02 2004.
- [44] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning, 2019.
- [45] David Meyer. Amazon reportedly killed an ai recruitment system because it couldn't stop the tool from discriminating against women. <https://fortune.com/2018/10/10/amazon-ai-recruitment-bias-women-sexist/>.
- [46] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Emre Kiciman. Social data: Biases, methodological pitfalls, and ethical boundaries. *SSRN Electronic Journal*, 01 2016.
- [47] Jahna Otterbacher, Jo Bates, and Paul Clough. Competent men and warm women: Gender stereotypes and backlash in image search results. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 6620–6631, New York, NY, USA, 2017. Association for Computing Machinery.
- [48] Neal J. Roes and James M. Olson. Counterfactuals, causal attributions, and the hindsight bias: A conceptual integration. *Journal of Experimental Social Psychology*, 32(3):197 – 227, 1996.

- [49] Harini Suresh and John V. Guttag. A framework for understanding unintended consequences of machine learning. 2020.
- [50] Latanya Sweeney. Discrimination in online ad delivery. *Commun. ACM*, 56(5):44–54, May 2013.
- [51] S. Verma and J. Rubin. Fairness definitions explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, pages 1–7, 2018.
- [52] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer, New York, 2004.
- [53] Daniel Westreich. Berkson’s bias, selection bias, and missing data. *Epidemiology (Cambridge, Mass.)*, 23:159–64, 11 2011.
- [54] P.A. Whigham. Search bias, language bias and genetic programming. 02 1970.
- [55] Jing Nathan Yan, Ziwei Gu, Hubert Lin, and Jeffrey M. Rzeszotarski. Silva: Interactively assessing machine learning fairness using causality. page 1–13, 2020.
- [56] Muhammad Zafar, Isabel Valera, Manuel Rodriguez, Krishna P. Gummadi, and Adrian Weller. From parity to preference-based notions of fairness in classification. 06 2017.
- [57] Qiuming Zhu. On the performance of matthews correlation coefficient (mcc) for imbalanced dataset. *Pattern Recognition Letters*, 136, 05 2020.