Università degli Studi dell'Aquila

Approach to design Data Science Pipeline with High Quality

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Corso di Laurea Magistrale in Informatica

Candidate

Giordano d'Aloisio
ID number 261821

Thesis Advisors

Prof.ssa Antinisca Di Marco
Prof. Giovanni Stilo

Academic Year 2020/2021

Approach to design Data Science Pipeline with High Quality
Master's thesis. Università degli Studi dell'Aquila

This thesis has been typeset by LaTeX and the uaqthesis class.

Author's email: giordano.daloisio@gmail.it

Alla mia famiglia
che mi ha sempre sostenuto
in tutte le mie scelte

Acknowledgments

This work is partially supported by Territori Aperti a project funded by Fondo Territori Lavoro e Conoscenza CGIL CISL UIL and by SoBigData-PlusPlus H2020-INFRAIA-2019-1 EU project, contract number 871042.

Abstract

With the rise of Data Science and Big Data in the last ten years, Data Science workflows have become an essential tool for scientists, researchers, and industries. The necessity to develop efficient Data Science systems starting from raw datasets, concatenating different steps of data preprocessing, model selection, and model evaluation thus becomes prominent. Over the years, several solutions have been proposed to automate the creation of Data Science pipelines, most of them concerning semantic aspects and characteristics of the input dataset. In addition, the quality of these systems is usually measured using metrics that do not take into account several crucial *quality requirements*, such as *Bias and Fairness* or *Model Explainability.*

In this thesis, we study Data Science workflows with the aim of identifying properties that can be used to define *Quality* requirements and constraints of Data Science pipelines. In particular, we focused on *Bias and Fairness* that in our opinion are among the most critical characteristics that must be studied nowadays. First of all, we analyzed the state of the art, identifying several definitions, metrics, and methods existing in the *Bias and Fairness* literature. Then we deeply analyzed three methods which allowed us to propose a new approach, the *Debiaser for Multiple Variables (DEMV)* which is an extension of the established *Sampling* method. Finally, we compare the proposed approach by comparing its performance with the established bias mitigation methods. To do so we employed a set of well-known biased datasets typically employed in the literature. The experiments have shown that *Debiaser for Multiple Variables (DEMV)* is the preferable method used to improves Bias and Fairness under certain conditions.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

DEMV Debiaser for Multiple Variables. 2, 3, 51, 53, 56–58, 60, 62–64

DS Data Science. 1–4, 6, 17, 19, 20, 64, 65

FPR False Positive Rate. 11

FR Functional Requirement. 17–19

QR Quality Requirement. 2, 4, 8, 19–22, 24–26, 63–65

TPR True Positive Rate. 11

Chapter 1

Introduction

Data Science (DS) is an emergent multidisciplinary research field that combines disciplines as *statistics*, *computer science*, *communication*, in conjunction with the application domain knowledge to extract useful insights and decisions from data and their environments. Nowadays, many domain experts, thus, face the necessity to realize their own Data Science project[14, 36], without having full knowledge of the underlying ivolved disciplines. Data Science (DS) workflows[1] represent useful tools used nowadays to build DS in a more effective way.

The workflows are defined as a set of processes that convert raw (unprocessed) data into actionable answers to business questions by leveraging innovative analysis techniques. Unfortunately, such workflows usually still require good knowledge and high expertise to choose the best techniques and models for the problem type that has to be solved. For this reason, many techniques have been proposed to automate some phases of the pipeline's building. However, these techniques mainly focus on semantic characteristics of the input dataset, ignoring other important *quality requirements* that should be taken into account during a DS project. In addition, the quality of such pipelines is usually measured solely considering performances (e.g. *Accuracy*, *Precision* and *Recall* as reported in [26]) of the used machine learning model without considering other important quality aspects which we will discuss later on.

For the aforementioned reasons, in this thesis, we are focused on study DS

[1] in this thesis, we will use workflows and pipelines in an interchangeably way

workflows from a quality point of view, identifying those quality requirements (QR) that can be employed to define the *Quality* of a DS pipeline. We think that a more extended list of them must include: the **Computational Complexity**, defined both by the memory space and the time required by the pipeline to be completed; the **Prediction's Quality** of the machine learning model, measured using classic metrics as *Accuracy*, *Precision*, *Recall*; the **Bias and Fairness** of the dataset and of the model respectively, the **Explainability** of the model's results, **Privacy** of dataset's sensitive information. Among all these quality attributes we will focus on the *Bias and Fairness* of machine learning systems since is one of the most interested and unexplored ones. It worth notice that the *Bias and Fairness* is still an unsolved research topic: e.g. considering the systems built by companies as Amazon[2] or Facebook[3] that have been proven to be biased towards some sensitive groups. For this reason, a lot of research has been conducted in this field in order to find the best ways to measure and improve the fairness of machine learning algorithms.

To summarise the aforementioned scenarios and aims, in this thesis, we have been focused at first to perform a qualitative analysis of Data Science workflows, highlighting how the selected Quality Requirement can influence the user's choices during all the steps of the pipeline.

Secondly, we made an in-depth analysis of the *Bias and Fairness* QR, making, first of all, a survey of the definitions, metrics, and mitigation approaches existing in the literature. In adjunction, we studied some established bias mitigation algorithms by applying them to datasets known in the literature to be biased. All the aforementioned works have led us to overcome some limits of the employed bias mitigation methods, by proposing a new method, the *Debiaser for Multiple Variables (DEMV)*, which is an extension of the *Sampling* method to a more general case. Lastly, exhaustive experimentation has proved the strength of the proposed approach.

[2] https://www.bbc.com/news/technology-47117299

[3] https://www.theverge.com/2021/4/9/22375366/facebook-ad-gender-bias-delivery-algorithm-discrimination

1.1 Thesis structure

This thesis is organised as follows:

- In chapter 2, we introduce the background knowledge and related works. In particular, we describe examples of systems that has been created to ease the development of DS workflows and show methods that have been developed to automate the building of specific pipelines steps. Then, we start focusing on the *Bias and Fairness* of machine learning models by making a survey of all the different definitions of *bias* and all the different metrics used to measure the *fairness* of a model. In addition, we describe a categorization of such metrics and their possible use cases. Finally, we describe existing algorithms to mitigate the bias of machine methods.

- In chapter 3, we describe the context in which our work has been done. In particular, we describe more in-depth our qualitative point of view on DS pipeline. We define the different selected quality requirements and show a high-level model of a workflow, highlighting how these requirements can influence the user's choices in all the pipeline's steps.

- Chapter 4 describes the experimental analysis we have conducted on bias mitigation methods and details our proposed extension of one of these methods. In particular, we first describe the selected datasets and the analysis methodology. Then, we describe the classic algorithms selected from literature and show the results of their application to the aforementioned datasets. Finally, we present the *Debiaser for Multiple Variables (DEMV)* and compare it with the other established methods.

- Chapter 5 concludes the thesis by first making a survey of the obtained results. Then, it describes the future works on both *Bias and Fairness* and the study of *High-Quality* DS workflows.

Chapter 2

Background Knowledge and Related Works

Over the years, a lot of literature has been produced about both the automation of DS workflow [47, 49] and bias and fairness [16, 53].

In this chapter, we make a survey of several types of research that are of interest for this thesis or our future works. We start by first make an overview of the most relevant scientific workflow platforms which can be of interest for the future works of our project (section 2.1). Then we highlight several approaches on automation of DS workflows, which focus mainly on functional aspects of a DS pipeline (section 2.2). Next, we start focusing on one of the selected QR, *Bias and Fairness*, by making a survey of all the different definitions and metrics existing in literature. Finally, we describe some proposed approaches to mitigate classification bias showing also a categorization of them (section 2.3).

2.1 Overview of Scientific Workflow Management Systems

A *Scientific Workflow Management System (SWfMS)* is a tool for develop and execute workflows and manage data sets in various computing environments [49]. Over the years many SWfMS have been developed addressing different user needs; here we will describe the ones that are more interesting for our domain. More

precisely, for the selection of the relevant SWfMS we have taken in consideration the following features: *i)* the presence of a user interface *ii)* the ease of extending the workflow features introducing new custom nodes *iii)* the ease of analyzing and reason about the generated workflow.

Galaxy [33] is a web-based platform used mainly to analyze large biomedical datasets. It provides a GUI for developing scientific workflows through browsers and a python API [60] for creating new modules and extending the already large repository of available methods. The created workflows can be exported as `.ga` files, which are nothing more than a JSON structured in a specific way, so they are easy to read and analyze. The main benefit of Galaxy over the other selected platforms is the existence of different public web-servers located at different geographic locations [1] allowing researchers to develop and run their workflow without the need of installing anything on their machines.

KNIME [11] is a modular environment, which enables visual assembly and interactive execution of data pipelines. It allows simple integration of new algorithms and tools as well as data manipulation or visualization methods in the form of new modules or nodes. KNIME provides a free and open-source desktop application built upon the Eclipse ecosystem and a commercial server application. A workflow description file can be exported as an XML file allowing researchers to read and analyze them easily. One main advantage of KNIME over the other SWfMS is his general-purpose nature and the magnitude of his repository of heterogeneous methods ranging from data preprocessing and manipulation to machine learning model selection and evaluation.

Kepler [51, 1] is an SWfMS build upon the Ptolemy II system as a desktop application. It provides an intuitive GUI for the design and execution of scientific workflows. Kepler workflows can be exchanged and analyzed in XML using Ptolemy's own markup language (MoML). Differently from other SWfMS, Kepler has an actor-oriented paradigm for modelling workflows. Each workflow step is implemented as an actor that abstracts from the underlying execution model allowing the plugin of different execution models into workflows.

[1] https://galaxyproject.org/use/

SWfMS	UI Presence	Extension ease	Analyze ease
Galaxy	X	X	X
KNIME	X	X	X
Kepler	X		

Table 2.1. SWfMS features

HyWare [13] is a novel SWfMS introducing the concept of *hybrid workflows*. A hybrid workflow is a workflow that contains both automated and manual tasks, where an automated task is a workflow step executed automatically by the system. In contrast, a manual task is a workflow step that the end-user must manually execute (like installing software or download a file from a specific repository). At the date of this thesis, the language has not been fully implemented yet. Still, the language definition and feature implementation approach has been beneficial for defining our own project work.

Table 2.1 summarizes the described features for each *SWfMS* considered. We have deliberately excluded HyWare from this synthesis since it has not been implemented yet, and so the features can not be analyzed. From the table, it can be seen that both Galaxy and KNIME are suited for our use case and can be considered as base platforms for our future works.

2.2 Approaches on automating Data Science workflows

Over the years, many solutions have been proposed to automate the data preprocessing or the model selection and evaluation steps of a DS workflow. The solutions that we evaluated can be grouped into two macro-categories: solutions that use a metadata-based approach and solutions that use a machine learning approach to automate these steps.

Starting from the first category of solutions, Rönkkö et al. [58] proposed the use of characterizations and a reachability algorithm to solve the problem of the selection and sequencing of preprocessing methods based on the user requirements. Their solution has been applied on environmental data, which are, citing of the authors,

"heterogeneous, as it may consist of data from sources such as weather stations, weather radars, chemical sensors, acoustic sensors, and off-line laboratory analysis". Goncalves Jr. and Barros [34] present a graphical application for the Data Mining Preparation Markup Language (DMPML), which is an XML application designed to represent the data preparation phase of the Knowledge Discovery in Databases (KDD) process. DMPML supports the reuse of data preprocessing directives using XSLT to map raw data into data ready to be used by many data mining algorithms. The application presented here, DMPML-TS, automates the data preparation phase, speeding up the codification and transformation of data, and providing support to facilitate the use of different data mining algorithms in the same and/or similar data based on their codification stored in separate XML documents. Gil et al. [32, 31, 64] present an approach where semantic metadata is generated as scientific data is being prepared and then used to configure models and customize them to the data. This metadata is then used in a workflow system to select analytic models dynamically and to set up model parameters automatically. In addition, all aspects of data processing are documented, and the system is able to generate extensive provenance records for new data products based on the metadata. As a result, the system can dynamically select analytic models based on the metadata properties of the data it is processing, generating more accurate results.

About the second category of solutions, AutoML [37] is an entire research branch focused on automating various steps of the data science workflow using machine learning techniques. A lot of research has been conducted on this branch, and describing the different proposed solutions would be out of the scope of this work. As an example of how an AutoML approach works on automating the creation of a data science pipeline, we cite TPOT [56], an open-source genetic programming-based AutoML system that optimizes a series of feature preprocessors and machine learning models to maximize classification accuracy on a supervised classification task.

These described methods mostly focus on semantic aspects of the data to suggest pre-processing algorithms or select the best machine learning models. Instead, our approach uses quality requirements, defined by the user, to guide they during all the steps of the pipeline.

2.3 Background knowledge on Bias and Fairness

Data is often heterogeneous, generated by subgroups with their own characteristics and behaviours. This heterogeneities can bias the data. A model learned on biased data may lead to unfair and inaccurate predictions [53].

In this section, we start focusing on the QR that concerns to *Bias and Fairness*. We start by first making a survey of the existing different definitions of bias in literature (section 2.3.1). Then, we will describe some existing definitions and metrics to measure model fairness and show a classification of them (section 2.3.2).

2.3.1 Bias definitions

A biased dataset is a dataset in which there is a statistical sample in which the probability of inclusion in the sample of individuals belonging to the population depends on the characteristics of the population under study. Bias in data can exist in many shapes and forms, some of which can lead to unfairness in different downstream learning tasks. For years, many definitions have been proposed, each trying to identify a different source of bias. By the time this thesis has been written, at least 23 different definitions of bias have been identified [53]. In the following, we cite the most common and important of them:

1. **Historical Bias** *Historical bias is the already existing bias and socio-technical issues in the world and can seep into from the data generation process even given a perfect sampling and feature selection* [62]

2. **Aggregation Bias** *Aggregation bias happens when false conclusions are drawn for a subgroup based on observing other different subgroups or generally when false assumptions about a population affect the model's outcome and definition.* [62]

3. **Temporal Bias** *Temporal bias arises from differences in populations and behaviors over time.* [57]

4. **Social Bias** *Social bias happens when other people's actions or content coming from them affect our judgment.* [3]

5. **Popularity Bias** *Items that are more popular tend to be exposed more. However, popularity metrics are subject to manipulation—for example, by fake reviews or social bots.* [18]

6. **Ranking Bias** *The idea that top-ranked results are the most relevant and important will result in attraction of more clicks than others.* [53]

7. **Evaluation Bias** *Evaluation bias happens during model evaluation. This includes the use of inappropriate and disproportionate benchmarks for evaluation of models.* [62]

8. **Emergent Bias** *Emergent bias happens as a result of use and interaction with real users. This bias arises as a result of change in population, cultural values, or societal knowledge usually some time after the completion of design.* [28]

9. **Behavioral Bias** *Behavioral bias arises from different user behavior across platforms, contexts, or different datasets* [57]

10. **Presentation Bias** *Presentation bias is a result of how information is presented* [4]

11. **Linking Bias** *Linking bias arises when network attributes obtained from user connections, activities, or interactions differ and misrepresent the true behavior of the users.* [57]

12. **Content Production Bias** *Content Production bias arises from structural, lexical, semantic, and syntactic differences in the contents generated by users.* [57]

Figure 2.1 describes a possible categorization of the different definitions of bias based on the cause that generated it. Bias can be generated by the data, by the machine learning algorithm or by the user interaction with some system that again generates data. However, this categorization is not strict because bias can be propagated through the pipeline and generate another type of bias at another point of the workflow. In particular, bias in data can cause a bias in the algorithm that in turn can cause a bias in the user interaction favouring, for example, one thing

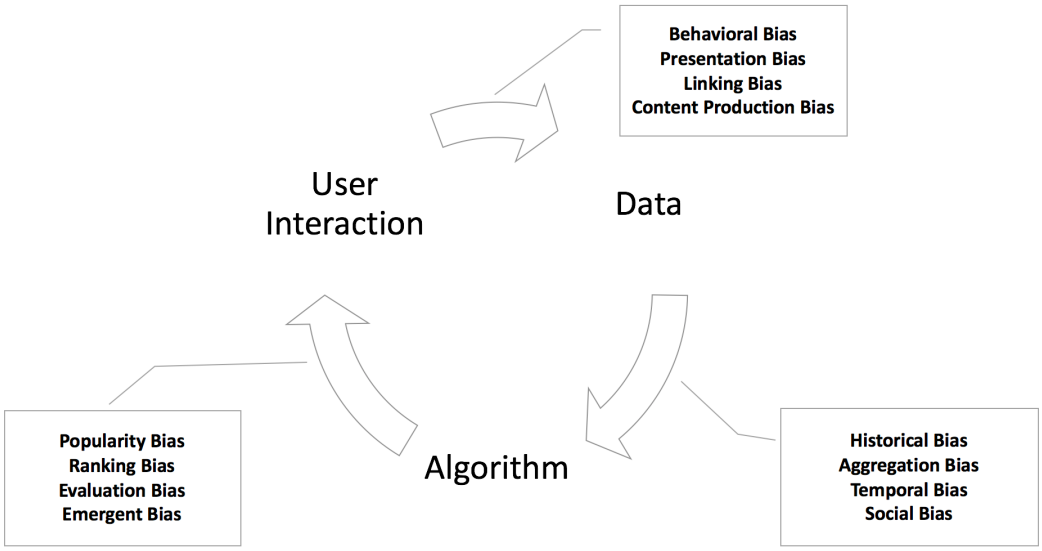


Figure 2.1. Bias Feedback Loop

instead of another. This process is called **Feedback Loop** [53] and is the reason why it is not possible to treat each definition of bias separately.

2.3.2 Definitions of Algorithmic Fairness and metrics

Algorithmic Fairness (that from now on we will call simply Fairness) can be defined as the absence of any prejudice or favouritism towards an individual or a group based on their intrinsic or acquired traits in the context of decision-making [59]. Usually, discrimination occurs with respect to some sensitive groups, identified by some *sensitive* (or *protected*) variables in the dataset . In particular, we define a privileged (unprivileged) group as a group (often defined by one or more sensitive variables) that are disproportionately (less) more likely to be positively/negatively classified [16]. Protected variables define the aspects of data that are socioculturally precarious for the application of ML. Common examples are gender, ethnicity, and age (as well as their synonyms). However, the notion of a protected variable can encompass any feature of the data that involves or concerns people [6].

The different definitions of fairness are strictly related to the metrics we use to measure the fairness of an algorithm. In the following, we list some fairness definitions and the related metric formulation:

1. **Statistical/Demographic Parity**

   One of the earliest definitions of fairness, this metric defines fairness as an equal probability of being classified with the positive label [23]:

$$Pr(\hat{y} = 1|A = 0) = Pr(\hat{y} = 1|A = 1) \tag{2.1}$$

2. **Disparate Impact**

   Similar to statistical parity, this definition looks at the probability of being classified with the positive label. However, in contrast to parity, Disparate Impact considers the ratio between unprivileged and privileged groups. Its origins are in legal fairness considerations for selection procedures which sometimes use an 80% rule to define if a process has disparate impact (ratio smaller than 0.8) or not [25]:

$$\frac{Pr(\hat{y} = 1|A = 0)}{Pr(\hat{y} = 1|A = 1)} \tag{2.2}$$

3. **Equal Opportunity**

   An algorithm is considered to be fair under equal opportunity if its TPR is the same across different groups. This means that the probability of a person in a positive class being assigned to a positive outcome should be equal for both protected and unprotected group members [65]:

$$Pr(\hat{y} = 1|A = 0, y = 1) = Pr(\hat{y} = 1|A = 1, y = 1) \tag{2.3}$$

4. **Equalized Odds (Average Opportunity)**

   Similarly to equal opportunity, in addition to TPR, equalized odds simultaneously considers FPR as well, i.e., the percentage of actual negatives that are predicted as positive [10]:

$$\begin{aligned} &Pr(\hat{y} = 1|y = 1 \ \ \& \ \ A = 1) = Pr(\hat{y} = 1|y = 1 \ \ \& \ \ A = 0) \ \ \& \\ &\qquad Pr(\hat{y} = 1|y = 0 \ \ \& \ \ A = 1) = Pr(\hat{y} = 1|y = 0 \ \ \& \ \ A = 0) \end{aligned} \tag{2.4}$$

5. **Generalized Entropy Index**

   The Generalized Entropy Index (GEI) [61] considers differences in an individual's prediction (b_i) to the average prediction accuracy (μ). It can be adjusted based on the parameter α, where $b_i = \hat{y}_i - y_i + 1$ and $\mu = \frac{\sum_i b_i}{n}$:

$$GEI = \frac{1}{n\alpha(\alpha-1)} \sum_{i}^{n} = 1[(\frac{b_i}{\mu})^{\alpha} - 1] \tag{2.5}$$

6. **Theil Index**

   Theil Index is a special case of GEI for $\alpha = 1$. In this case, the formula simplifies to:

$$Theil = \frac{1}{n} \sum_{i=1}^{n} (\frac{b_i}{\mu}) \log(\frac{b_i}{\mu}) \tag{2.6}$$

Finally, there are two other versions of **Statistical Parity** and **Disparate Impact** that take into consideration the true label of the items and not the predicted ones. Formally they are defined as:

$$\textbf{Statistical Parity} : Pr(y = 1|A = 0) = Pr(y = 1|A = 1) \tag{2.7}$$

$$\textbf{Disparate Impact} : \frac{Pr(y = 1|A = 0)}{Pr(y = 1|A = 1)} \tag{2.8}$$

These metrics can be used to measure the fairness of the dataset instead of the classifier.

Fairness definitions can be classified into two different groups [23, 45]:

- **Individual Fairness:** give similar predictions to similar individuals
- **Group Fairness:** treat different groups equally

Table 2.2 shows the categorization of the definitions described above. Based on the type of application and fairness they want to have, users may choose to use one category of metrics or another.

Concerning group fairness, we distinguish two different definitions of it: *We Are All Equal (WAE)* [27] and *What You See Is What You Get (WYSIWYG)* [66]. The

Definition	Group	Individual
Statistical Parity	X	
Disparate Impact	X	
Equal Opportunity	X	
Equalized Odds	X	
GEI Index		X
Theil Index		X

Table 2.2. Categorization of fairness definitions

Definition	WAE	WYSIWYG
Statistical Parity	X	
Disparate Impact	X	
Equal Opportunity		X
Equalized Odds		X

Table 2.3. Categorization of group fairness

WAE definition holds that all groups have similar abilities with respect to the task (i.e. have the same probability to be classified in a certain way). The *WYSIWYG* definition holds, instead, that the observations reflect ability with respect to the task (i.e. an item should be classified in a certain way only if the other attributes imply it). If the user application follows the *WAE* definition, then the demographic parity metrics should be used: *Disparate Impact* and *Statistical Parity*. If the application follows the *WYSIWYG* definition, then the equality of odds metrics should be used: *Equal Opportunity* and *Equalized Odds*. Table 2.3 synthesizes these categories of group fairness.

2.3.3 Methods for bias mitigation

Over the years, many approaches have been proposed to mitigate bias and improve the fairness of machine learning algorithms [53, 16]. These approaches can

be categorized into three groups [45]:

- **Pre-processing**

  Pre-processing techniques try to transform the data so that the underlying discrimination is removed.

- **In-processing**

  In-processing techniques try to modify and change state-of-the-art learning algorithms in order to remove discrimination during the model training process.

- **Post-processing**

  Post-processing is performed after training by accessing a holdout set that was not involved during the training of the model.

The choice among algorithm categories can partially be made based on the system's ability to intervene at different parts of a machine learning pipeline. If the system is allowed to modify the training data, then pre-processing can be used. If the system is allowed to change the learning algorithm, then in-processing can be used. If the system can only treat the learned model as a black box without any ability to modify the training data or learning algorithm, then only post-processing can be used. However, it is recommended to apply the earliest category of methods possible in order to have the most flexibility and opportunity to correct bias [9].

The literature is dominated by approaches for mitigating bias and unfairness in machine learning within the problem class of *binary classification*. There are many reasons for this, but most notably [16]:

1. Many of the most contentious application areas that motivated the domain are binary classification problems (hiring vs. not hiring; offering a loan vs. not offering a loan; re-offending vs. not re-offending etc.).

2. Quantifying fairness on a binary dependent variable is mathematically more convenient; addressing multi-class problems would at the very least add terms in the fairness quantity.

In the following we cite the main approaches to handle bias in the binary classification case. Within these approaches, many methods have been defined. They are mostly *pre-processing* methods, however some of them can belong to multiple classes (e.g. *pre-processing* and *in-processing*):

1. **Blinding**

   Blinding is the approach of making a classifier *immune* to one or more sensitive variables [67]. A classifier is blind for a sensitive variable if there is no observable outcome differentiation based on that variable. Some works have termed the omission of sensitive variables from the training data as blinding [16]. However, omission has been shown to decrease the model accuracy and not improve the model fairness [35, 41].

2. **Sampling and Subgroup Analysis**

   Sampling methods have two primary objectives:

   (a) To correct the training data and eliminate bias [38]

   (b) To identify groups (or subsamples) of the data that are significantly disadvantaged by a classifier [16]

   Approaches that seek to create fair training samples include notions of fairness in the sampling strategy. [41] proposed a way to sample (by over-sampling) instances that belong to some *improbable groups*. This method is one of the selected algorithms for our analysis.

3. **Transformation**

   Transformation approaches learn a new representation of the data, often as a mapping or projection function, in which fairness is ensured, but still preserving the fidelity of the machine learning task [25]. Current transformation approaches operate mainly on numeric data, which is a significant limitation [25]. There are many ways to transform the training data: operating on the label ([21]), transforming the numerical non-sensitive variables ([25]), mapping individuals to an input space which is independent of specific protected

subgroups. Among all the proposed methods we cite the *Disparate Impact Remover* [25], which is one of the methods used in our analysis.

4. **Relabelling and Perturbation**

   Relabelling and perturbation approaches are a specific subset of transformation approaches: they either flip or modify the dependent variable (*relabelling* [19]), or otherwise change the distribution of one or more variables in the training data directly (*permutation* [39]). Referred to as *data-massaging* by [41], relabelling involves the modification of the labels of training data instances so that the proportion of positive instances are equal across all protected groups. Perturbation instead often aligns with notions of "*repairing*" some aspects of the data with regard to notions of fairness. However, is important to say that the modification of the data via relabelling and perturbation is not always legally permissible [7] and changes to the data should be minimised [52].

5. **Reweighing**

   Unlike transformation, relabelling, and perturbation approaches which alter certain instances of the data, reweighing assigns weights to instances of the training data while leaving the data itself unchanged [16]. Weights can serve multiple purposes:

   (a) To indicate a frequency count for an instance type ([12])

   (b) To indicate lower or higher importance on sensitive training samples ([41])

   (c) To improve classifier stability ([43])

   Methods using the reweighing approach can be classified between the *pre-processing* and *in-processing* classes. With appropriate sampling, reweighing can maintain high accuracy when compared to relabelling and blinding (omission) approaches [41]. Among all the proposed reweighing methods, we cite the algorithm proposed in [41] which will be used in our analysis

Chapter 3

Context Definition

In this chapter, we describe our proposed approach by defining a set of *Functional* and, especially, *Quality* requirements to which a DS workflow must comply. We start by first defining a set of *Functional requirements* that compose most generic DS pipelines (section 3.1). Next, we focus on the requirements that can be used to define our concept of *High Quality* of a DS workflow (section 3.2). Finally, we show a high-level model of a DS pipeline, highlighting how the different quality attributes that we have selected can influence the user's choices while developing it (section 3.3).

3.1 Functional Requirements

In software engineering, a *Functional Requirement* (FR) defines a function of a system or its components, where a function is described as a behaviour between inputs and outputs [29].

In the context of a DS workflow, the functional requirements define the steps of the workflow that go from a raw dataset to a machine learning model ready for production. These steps can be described as follows:

1. **Data Preprocessing**

   The system must be able to manipulate and transform data according to the user's needings.

This FR includes all the operations required to transform a raw dataset to a clean and processed dataset that can be given as an input to train and test a machine learning model.

2. **Model Selection**

   The system must allow the user to select a machine learning model from a range of possible models suited for the problem that has to be addressed.

   This FR describes the model selection phase, in which the user can select a machine learning model that is able to solve the required task (like classification or regression problem). The range of selectable machine learning models has to be filtered accordingly to the type of task that has to be solved. The user can return to this step also after advancing in the pipeline if he has to change the selected model for some reasons.

3. **Model Tuning**

   The system must allow the user to train and test the selected machine learning model in order to find the best hyper-parameters that optimize the model's performances and results.

   This requirement describes the training and testing phase of a machine learning model. There are many ways to train and test a model, but as a general rule, we can say that the model must be able to generalize to data that he have never seen before. For this reason, *in sample* evaluation (training and testing a model on the same set of features) must be avoided [26]. Instead, a more correct way of doing training and testing is by using a *holdout set*, which is a set of data removed from the training set and used only for testing the model. *Cross validation* is a more sophisticated version of the *holdout set* technique, which provides splitting the initial dataset into k subsets (usually k is equal to 5 or 10 [26]) and then iteratively train the model on $k-1$ sets taking out each time a different set for testing. This technique is preferable over the *holdout set* technique because allows the model to generalize better and to test it in a more realistic case scenario.

4. **Model Explainability**

   The system must allow the user to apply explainability methods to the trained model.

   This requirement is related to applying machine learning explainability methods to the final machine learning model. We will discuss more in detail about model explainability in the next section.

As said at the beginning, these FR are the building blocks of most generic DS pipelines, and we have used them to model a pipeline that is as generic as possible.

3.2 Quality Requirements

In software engineering, a *Quality Requirement* (QR or Non-Functional Requirement) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours [17].

To analyze a DS workflow from a qualitative point of view, we have to determine which are the quality aspects that we can use to judge the operation of the workflow and that can actually influence the user's decisions. In the following, we will describe the identified QR distinguishing between quantitative QR (in which the user have to define a threshold with respect to some metrics) and qualitative QR (in which the value is a yes or no flag):

1. **Computational Complexity**

   This quantitative QR defines the computational complexity of a DS pipeline as the pair *Space Complexity (SC)* and *Time Complexity (TC)*, where the first indicates the amount of memory required by a workflow to perform all its operations. Instead, the second indicates the time required by the workflow to complete. Both *SC* and *TC* are a function of at least the size of the input dataset [63]. This QR has influence in various stages of the workflow and has also influence on other QR like *Quality of Predictions* or *Explainability*.

2. **Quality of Predictions**

This quantitative QR is used to define how good the model must be in predicting outcomes. There are different metrics in literature to compute the prediction quality of a machine learning model, each addressing a different goal of the user. Some examples of metrics for the prediction quality could be [26]:

- *Precision*: fraction of true positives (TP) with respect to the total positive predictions:
$$\frac{TP}{(FP + TP)} \tag{3.1}$$
- *Recall*: fraction of TP to the total positive items of the dataset:
$$\frac{TP}{(TP + FN)} \tag{3.2}$$
- *F1 Score*: harmonic mean of *Precision* and *Recall*:
$$\frac{2 * Precision * Recall}{Precision + Recall} \tag{3.3}$$
- *Accuracy*: fraction of True Positives (TP) and True Negatives (TN) above the total of predictions:
$$\frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$
- *Root Mean Squared Error*: square root of the mean of the square of all the errors (this metric is mostly used for regression problems):
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.5}$$

This requirement is quite essential for developing a *high-quality* DS workflow. In fact, it is sometimes used as the only metric to quantify the quality of a pipeline. For this reason, we have decided to include it in our definition. It impacts mostly the **Model Tuning** step of our workflow and is influenced by other selected QR.

3. **Explainability**

Explainability can be defined as the ability of a system of enabling user-driven explanations of how a model conclusion is reached, [20]. There are researches in

literature explaining why explainability must be considered a QR [46] allowing human-in-the-loop systems to make better decisions and outcomes. This QR can influence the *Privacy Level* of the dataset and the *Computation Complexity* of the workflow and can be influenced by the *Bias* and *Unfairness* mitigation methods.

4. **Privacy**

   Privacy [30] can be defined as a qualitative QR allowing sensitive information of a dataset to be hidden, changing the value of some attributes. Privacy can be quantified with a crescent level where 0 means that the dataset has no privacy encryption. This feature impacts mostly the *explainability* since a higher level of privacy can cause the model to be less explainable.

5. **Bias and Fairness**

   Bias (and consequently *unfairness*) has several definitions, and its common usage is decidedly negative. We typically use it to mean systematic favouritism of a group. In Data Science, bias is a deviation from expectation in the data. More fundamentally, bias refers to an error in the data [24]. Instead we say that a model is *fair* if it does not have any prejudice or favouritism towards an individual or a group based on their inherent or acquired characteristics [53]. We will deep more on several definition and metrics for bias and fairness in chapter 2.3, but for now, we say that this qualitative QR influences the *explainability* (since some methods for bias mitigation require to change the label of some sensitive attributes), the *computational complexity* (since some methods for bias mitigation are quite complex computationally) and the *quality of predictions* (since mitigating the bias can sometimes influence the performance of a model) of the pipeline.

Figure 3.1 summarizes the QR that compose our definition of *quality*, showing in yellow the quantitative QR and blue the qualitative ones. Arrows in the graph mean that a QR A as an influence on a QR B. Sometimes, for example, for the *computational complexity* and the *quality of predictions*, the influence is mutual;

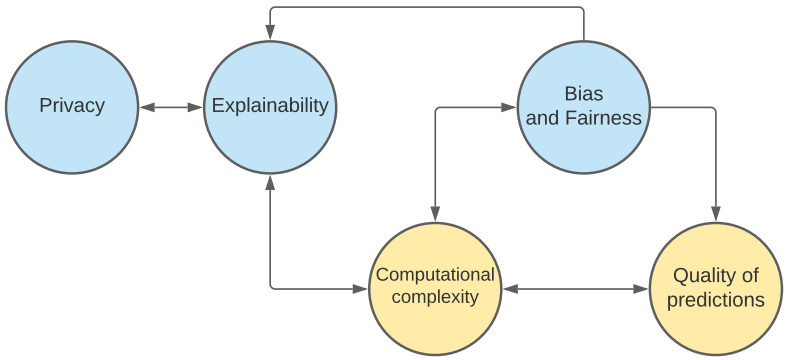


Figure 3.1. Quality attributes influence

in other cases, like for the *problem type* and the *computational complexity*, it is mono-directional.

3.3 Data Science Workflow overview

Figure 3.2 shows a high-level view of the modelled workflow. As before, the quality QR are distinct in qualitative (blue) and quantitative (orange), and, as can be seen from the picture, they are arranged all along the pipeline, meaning that they will guide the user through the development of all the pipeline's phases. The user symbol above the features means that they have to be defined by the user.

The pipeline is composed of several steps, each influenced by one or more QR. In the following, we will describe more in detail each of these phases and, to simplify the workflow, we will assume that the user wants to generate a fair model; if instead, a user does not want to generate a fair model, the workflow will skip all the bias and fairness related tasks.

3.3.1 Data Preprocessing

Figure 3.3 describes the data pre-processing phase. After explicitly defining the thresholds for the quality metrics, the user has to load the data and apply the first set of pre-processing operations (like handle missing values, encode the data, and so on) that are not influenced by our quality attributes. After this process, there is

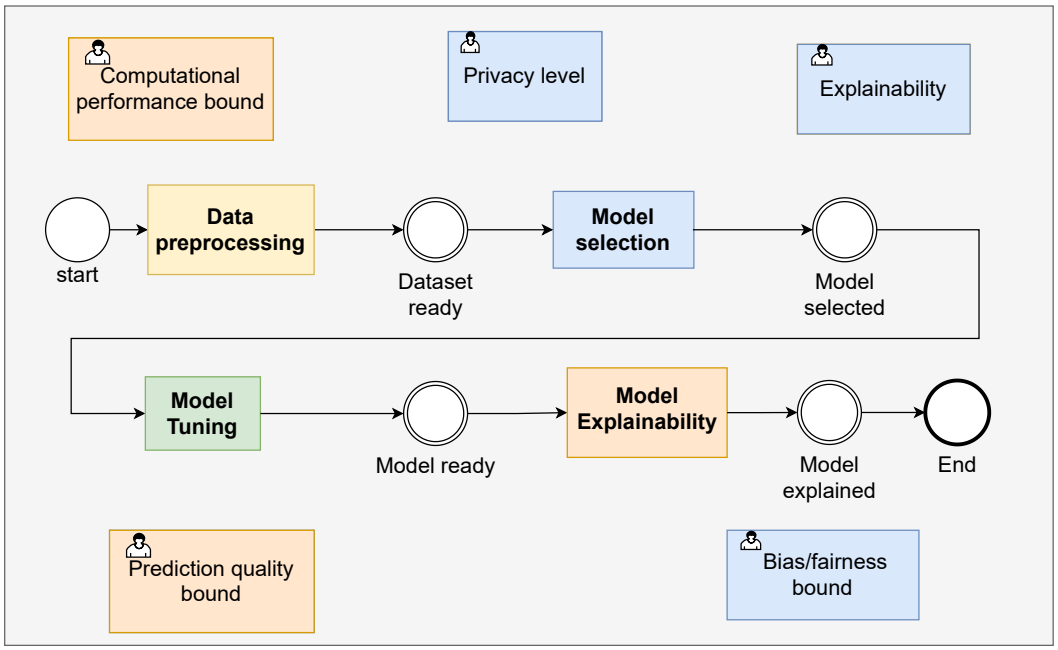


Figure 3.2. Data Science Workflow

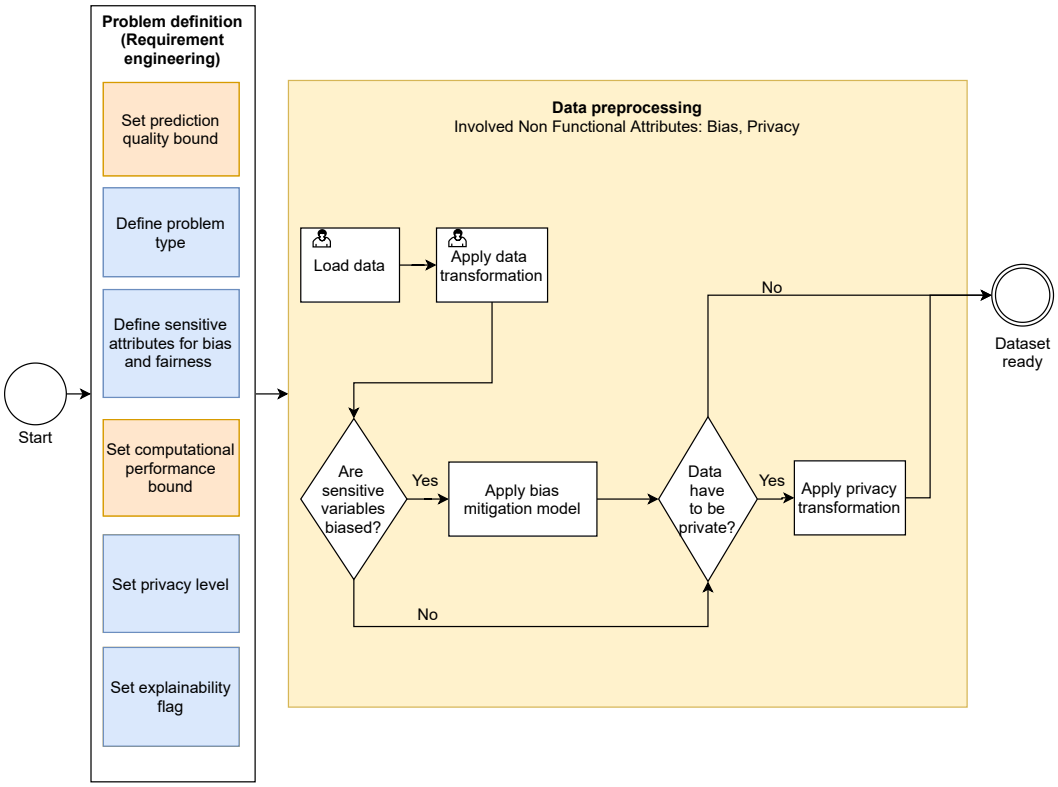


Figure 3.3. Data Preprocessing workflow

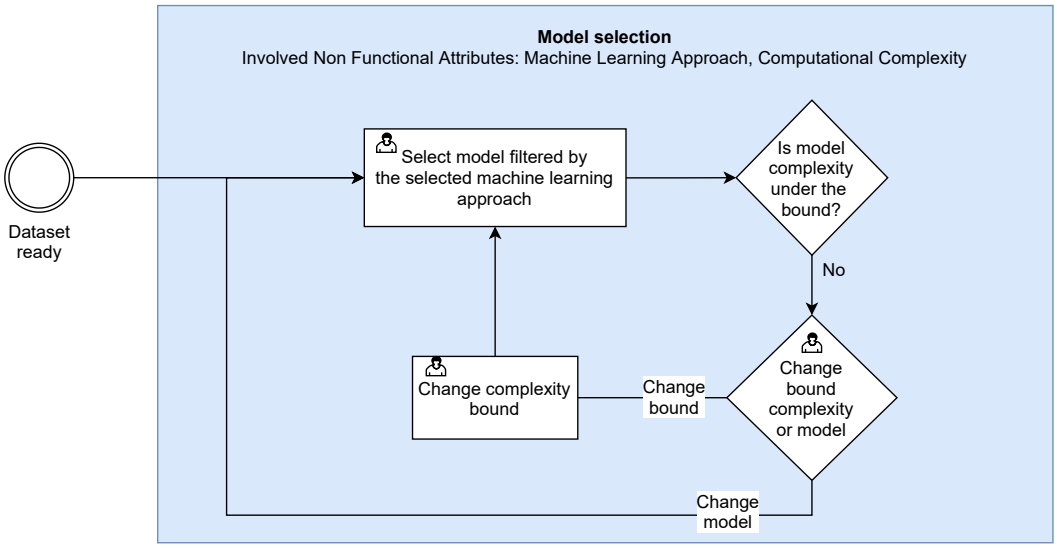


Figure 3.4. Model Selection workflow

a first decision point caused by one of the quality attributes: in fact, the process checks for bias in the data set using the sensitive attributes defined by the user as a reference and, if it finds that a certain subgroup is discriminated against compared to another group, it applies a bias mitigation algorithm to remove it. Then there is another decision point motivated this time by the privacy QR: if the data has to be private, then a privacy transformation is applied to them; otherwise, the workflow moves to the next step.

3.3.2 Model Selection

The next step of our pipeline is the Model Selection phase, which is shown in figure 3.4. This step takes as input the processed dataset from the previous data preprocessing phase and returns a machine learning model suitable for the requirements defined. In particular, the set of selectable machine learning models is filtered by the system based on the type of machine learning problem that has to be handled and on the computational complexity threshold defined by the user at the beginning. Each machine learning model will have space and time computational bounds associated; as an example for a classification problem, let n be the size of the training sample, d the number of dimensions, and k the number of classes. Space and time complexity for *KNN*, *SVM* and *Logistic regression* models will be [5, 44]:

$$\text{KNN}: \quad TC = O(knd) \quad SC = O(nd) \tag{3.6}$$

$$\text{SVM}: \quad TC = O(n^2) \quad SC = O(kd) \tag{3.7}$$

$$\text{Logistic regression}: \quad TC = SC = O(nd + d + n) \tag{3.8}$$

So this step starts by first filtering the models by the machine learning approach (classification, regression, etc...), then, after the user selects one of the models, the system checks if his complexity is under the bound defined by the user, if so the process continues to the next step; otherwise, the system asks the user to change the complexity bound or the selected model. We have decided to not filter the models by their complexity in the selection phase but to do this check only after the selection of the model because, in this way, the user has access to all the possible models for a specific type of problem and then, if he wants to use a specific type of model whose complexity his higher than his bound, he can directly change the pipeline's complexity bound.

3.3.3 Model Tuning

The workflow continues with the Model Tuning step shown in figure 3.5. This step takes as input the selected model and the processed dataset from the previous steps and returns the same model trained and tested ready for production. This phase is one of the most critical in the pipeline since it is influenced by several QR, some of which are in contrast with each other. The step starts with the training and testing of the model to find the best hyperparameters. After tuning the model, the system checks his quality of predictions (using one of the metrics defined in section 3.2) and, if the quality is under the threshold defined by the user, it asks him to change the defined quality bound or directly the machine learning model (and consequently the computational bound). If the user chooses to change the quality, bound the workflow restarts to train and test the model using the new bound as a threshold; otherwise, if the user chooses to change the model, the workflow returns to the Model Selection step allowing the user to change the computational complexity threshold. If the results are above the quality bound, then the system

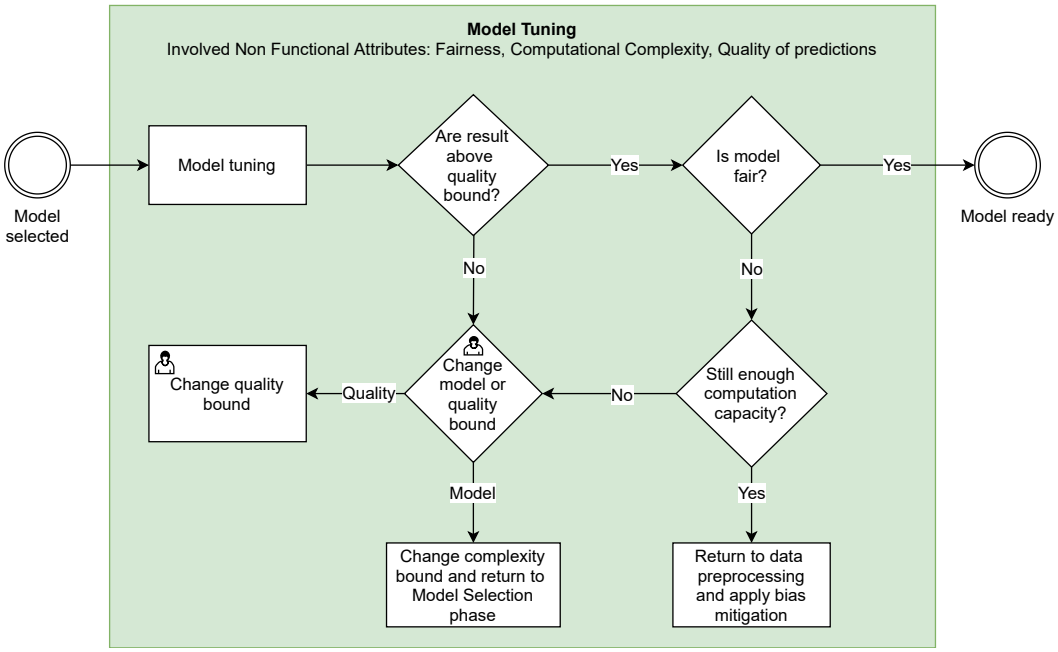


Figure 3.5. Model Tuning workflow

checks for the fairness of the classifier: if the classifier is unfair and there is still enough computation capacity, then the workflow returns to the Data Preprocessing phase, allowing the user to apply bias mitigation methods and eventually change the machine learning model. When the system verifies the fairness of the classifier, the workflow can proceed to the final step.

3.3.4 Model Explainability

The last step of the workflow is the Model Explainability step, in which we apply explainability algorithms to the final trained model. As can be seen in figure 3.6, this step is influenced by several QR. A first decision point is made at the beginning of the workflow. We check if the explainability is a user requirement; if so, the pipeline continues; otherwise, it skips this step entirely, going directly to the end. If the explainability is a user requirement, a second decision point concerns the self-explainability of the selected machine learning model. In fact, there are machine learning models, like Decision Trees, that are *white-box* and, for this reason, they do not require any other algorithm to make them explainable; on the other hand, *black-box* models, like Neural Networks, require the application of

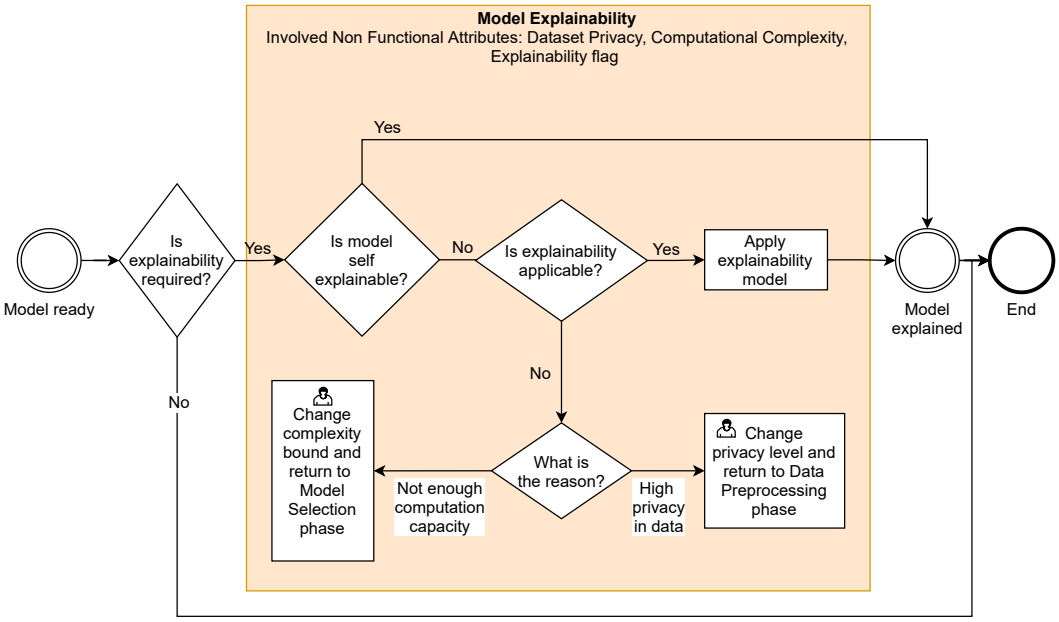


Figure 3.6. Model explainability workflow

post-processing algorithms to explain their results [50]. If we are using a *black-box* model, a third decision point concerns the applicability of explainability models. There are two reasons for an explainability model not to be applicable: high privacy applied to data and not enough computation capacity still available. In the first case, changing the dataset's values to ensure high privacy can impact the model explainability since it could be difficult to go back to the original values. In this case, the user can choose to change the privacy applied to the dataset returning to the Data Preprocessing phase. Concerning the second reason, explainability models are usually quite complex computationally [48], and so it could be possible not to have enough computational capacity still available. In this case, the user can change the computational complexity threshold returning to the Model Selection step. The user can also choose to remove the explainability requirement in both cases, keeping the defined privacy and computational boundaries. This second choice has not been highlighted in the figure to keep the diagram more concise and readable. Finally, if explainability is applicable, the selected explainability algorithm is applied to the model, and the workflow ends.

Chapter 4

Experimental analysis on Bias and Fairness

In chapter 2 we described the different sources of *bias* and shown how it can be propagated through different points of the workflow. As a proof of concepts, figure 4.1 shows the classification report of a classifier trained with a biased dataset. As we can see from the figure, the model tends to favor the privileged group assigning more positive labels to it. Instead, for the unprivileged group we can see that the *Recall* is near zero, meaning that the model is unable to predict the true positives at all. This is an example of the aforementioned **Feedback Loop**, in which the bias of the dataset is propagated to the model, which, in turn, could influence negatively the user's behaviors. For this reason, many methods have been developed in order to mitigate the bias and prevent his propagation.

In this chapter, we selected and analyzed the performances of three established pre-processing algorithms and proposed two novel approaches. We have focused our analysis only on pre-processing methods because, as said in chapter 2, they are the preferred choice if pre-processing is applicable. This chapter is structured as follows: in section 4.1 we describe the datasets used for our analysis. In section 4.2, we describe the applied methodology. In section 4.3 we describe the selected fairness algorithms and show how they affect the datasets. In section 4.5 we describe in detail our proposed extension of *Sampling*. Finally, in section 4.4 we show the experiment results.

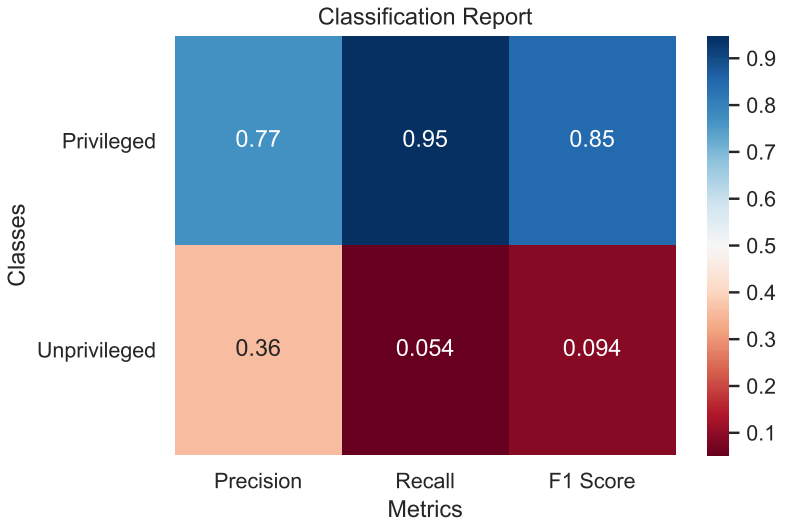


Figure 4.1. Classification report of a bias classifier

4.1 Selected datasets

In this section, we describe the datasets selected for our experiment. In particular, we selected a heterogeneous set of datasets that are known in literature to be biased [53]. In addition, we also created a synthetic dataset which allowed us to make a first analysis of the methods in a controlled environment. The selected datasets are the following:

1. **Synthetic Dataset**[1]

   This is a synthetic dataset created from scratch using the `make_classification` function of the **Scikit-learn** library [2]. This dataset is made of 10000 random samples for 12 attributes, in which the attribute `10` is the label to predict and the attribute `s` is the sensitive variable. We have analyzed this dataset in three versions:

   (a) An unbias version in which the distribution of the labels is the same for both groups.

[1] https://github.com/giordanoDaloisio/bias-mitigation-methods/blob/main/synthetic/synthetic.csv

[2] https://scikit-learn.org/stable/index.html

(b) A bias, unbalanced version of the dataset, in which the number of negative labels is the double of the positive ones and are mostly for items with a value of `s` equal to zero.

(c) A bias, balanced version, in which the number of negative and positive labels is the same, but most of the positive labels are for items of the privileged group.

2. **Adult Income Dataset**[3]

The **Adult Dataset** [42] is a dataset made of 30940 items for 15 features. The records of this dataset contain information about people extracted from the 1994 census bureau database. The goal is to predict if a person has an income higher than 50k a year. This information is represented by the `income` variable. This dataset has been analyzed in two version:

(a) A single sensitive variable version, where the protected variable is `sex` and the men are privileged over women

(b) A double sensitive variable version, where the protected attributes are `sex` and `race` and the privileged group are white men, while the the unprivileged group are black women.

3. **Bank Marketing Dataset**[4]

The **Bank Marketing** dataset [55] is a dataset related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe to a term deposit (variable `y`). The sensitive variable is `age`, and the privileged group are people with less than 25 years (`age` variable equal to one). The dataset is made of 30488 samples for 21 columns.

4. **German Credit Dataset**[5]

The **German Credit** dataset [22] classifies people described by a set of attributes as good or bad credit risks (`credit` variable). The dataset consists

[3] https://archive.ics.uci.edu/ml/datasets/Adult

[4] https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

[5] https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29

of 1000 instances and 20 features. Like the **Bank** dataset, here the sensitive variable is `age`. However, in this case, the privileged group are people with more than 25 years (`age` variable equal to one), which are more likely to be classified as *good* credit risk (`credit` variable equal to one). In addition, we considered also a multiple sensitive variable version of it where the sensitive variables are `age` and `age`. In this case, the privileged group are men with more than 25 years, while the unprivileged group are women with less than 25 years

5. **ProPublica Recidivism (COMPAS) Dataset**[6]

   The **ProPublica Recidivism** dataset (that for simplicity we will call **COMPAS** dataset) is the dataset used to train the COMPAS algorithm. COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is a popular commercial algorithm used by judges and parole officers for scoring criminal defendant's likelihood of reoffending (recidivism). It has been shown that the algorithm is biased in favour of white women defendants, and against black men inmates, based on a 2 year follow up study (i.e. who actually committed crimes or violent crimes after 2 years) [2]. This dataset is made of 6167 samples for 398 attributes. The sensitive variables are `sex` and `race`. The goal is to predict if a person will be a recidivist in the next two years. The favourable label, in this case, is 0, and the privileged group are caucasian women (items with `sex` and `race` equal to one).

4.2 Methodology

In this section, we describe the methodology used for our analysis. In particular, we have evaluated the performances of the selected methods using *10-fold* cross-validation in order to have more stable results. Cross-validation is a model-evaluation technique that involves the split of the training data in k different folds (smaller subsets). A model is trained using $k-1$ of the folds of the training data; the resulting trained model is then evaluated on the remaining part of the data (used as a test set). The performance measures reported by k-fold cross-validation are

[6] https://github.com/propublica/compas-analysis

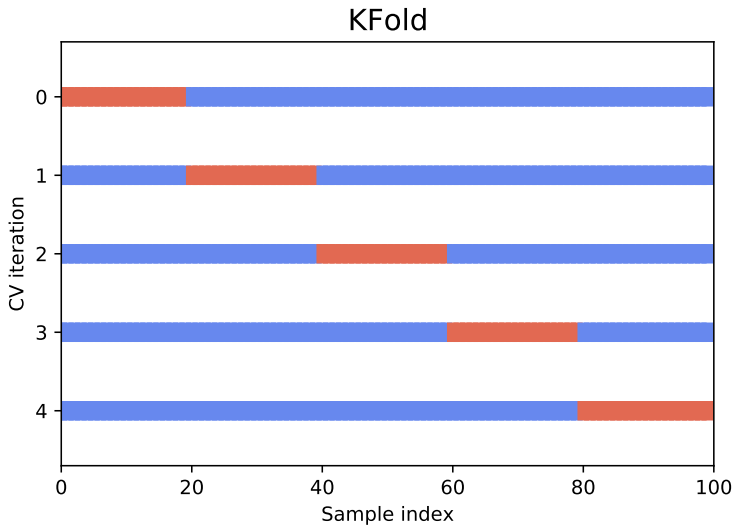


Figure 4.2. Example of *5-fold* cross-validation

the average of the metrics obtained for each fold. Figure 4.2 shows an example of *5-fold* cross-validation where in orange there are the testing sets and in blue are the training sets. As we can see, in each iteration a different subset is selected for testing: this ensures better stability and more realistic results.

The classifier used for our analysis is a *Logistic Regression* classifier. This algorithm, despite its name, implements a linear classification model. In this model, the probabilities describing the possible outcomes of a single trial are modelled using a *logistic function*, which is a function whose values are between zero and one. The logistic function is defined as:

$$\text{logistic}(\theta) = \frac{1}{1 + exp(-\theta)} \tag{4.1}$$

Where θ is the right side of a classic linear regression model [54]:

$$\theta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \tag{4.2}$$

In other words, *Logistic Regression* models can be seen an adaptation of Linear Regression models to classification problems. We have decided to use this model because it is quite simple and does not require much hyper-parameters configuration. In this way, we can focus our analysis only on the debias and fairness methods

leaving the model as it is.

The evaluation process has been implemented as follows:

1. First, we split the dataset in *training* and *testing* sets using the k-fold splitting described above.

2. Then, we apply the selected fairness algorithm **only** on the training set and train the *Logistic Regression* classifier with it

3. Then, using the testing set, we compute the following metrics: *Balanced Accuracy*, *Disparate Impact*, *Statistical Parity*, *Average Odds*, *Equal Opportunity*, *Theil Index* and dataset's *Statistical Parity* and *Disparate Impact*. For each dataset, we compute also the classification report (*Precision*, *Recall* and *F1 Score*) for both privileged and unprivileged groups.

4. We repeat the steps 2 and 3 for each dataset's split and then return the average of all these metrics.

For *Reweighing* and *Sampling* we have not removed the sensitive variables from the training and testing sets, while for *DIR* we have removed the sensitive variables before training and testing the models as suggested in [25].

4.3 Employed classic debias algorithms

For our analysis, we have selected three of the most used pre-processing bias mitigation algorithms and compared their performances using the metrics described in section 2.3. For the sake of simplicity, we have limited our analysis to algorithms related to classification problems. In the following, we describe the selected methods.

4.3.1 Reweighing

Reweighing [41] is an algorithm that applies weights to the dataset's items according to the sensitive group they belong to and the label values. For example, items of the unprivileged group with a positive label will get higher weights than those with a negative label. Weights are calculated as the ratio between the expected

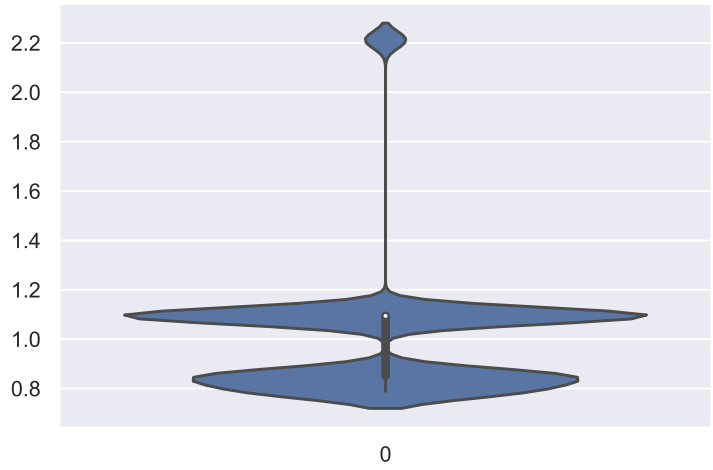


Figure 4.3. Distribution of weights

probability of an item of a group to have a certain label and the observed probability of an item of a group to have a certain label:

$$W(X) = \frac{P_{exp}(S = X(S) \wedge Class = X(Class))}{P_{obs}(S = X(S) \wedge Class = X(Class))} \tag{4.3}$$

where:

$$P_{exp}(S = s \wedge Class = c) = \frac{|\{X \in D | X(S) = s\}|}{|D|} \cdot \frac{|\{X \in D | X(Class) = c\}|}{|D|} \tag{4.4}$$

$$P_{obs}(S = s \wedge Class = c) = \frac{|\{X \in D | X(S) = s \wedge X(Class) = c\}|}{|D|} \tag{4.5}$$

Figure 4.3 shows an example distribution of weights for a biased dataset with a single sensitive variable. As we can see, there are two clusters with low weights and one small with a higher weight corresponding to items that differ from the expected observations.

4.3.2 Disparate Impact Remover

Disparate Impact Remover (DIR) [25] is a pre-processing bias mitigation algorithm that changes the unprotected features of the dataset to remove the correlation between the protected attributes and them. The idea behind this algorithm is that

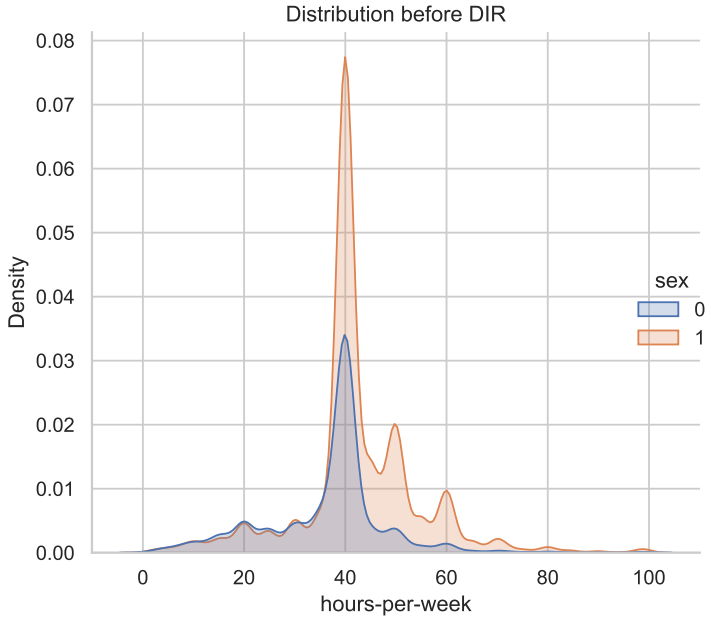
even if we remove sensitive attributes from the dataset, a classifier can always trace this information from the value of other variables related to them. Given a protected attribute X and a single numerical unprotected attribute Y, let $Y_x = Pr(Y|X = x)$ be the marginal distribution of Y conditioned on $X = x$. The *rank* of $y \in Y_x$ is then defined as the cumulative distributions $F_x : Y_x \to [0, 1]$ for values $y \in Y$. In order to preserve the ability to predict the label correctly, this algorithm preserves the *rank* of the items inside each group. Formally, let $\bar{Y}$ be the repaired version of Y in the repaired dataset $\bar{D}$. We say that $\bar{D}$ *strongly preserves rank* if for any $y \in Y_x$ and $x \in X$, its repaired version $\bar{y} \in \bar{Y}_x$ has $F_x(y) = F_x(\bar{y})$.

Figure 4.4 shows the distribution of a nonsensitive variable (`hours-per-week`) before and after the application of the *DIR* algorithm. In this case, `sex` was the sensitive variable, and we can see in figure 4.4(a) how the unprotected variable was related to it. In particular, a model could infer that if an item has a value of `hours-per-week` greater than 40, then it is very likely to have a value of `sex` equal to one, and so apply discrimination based on this information. Instead, after the application of *DIR*, we can see in figure 4.4(b) how the distribution of `hours-per-week` is more balanced. In this case, it is more difficult for a model to infer the membership group of an item looking only at the unprotected variable. It is also worth noting that the shape of the distributions is preserved, meaning that the rank of items inside the groups is preserved.

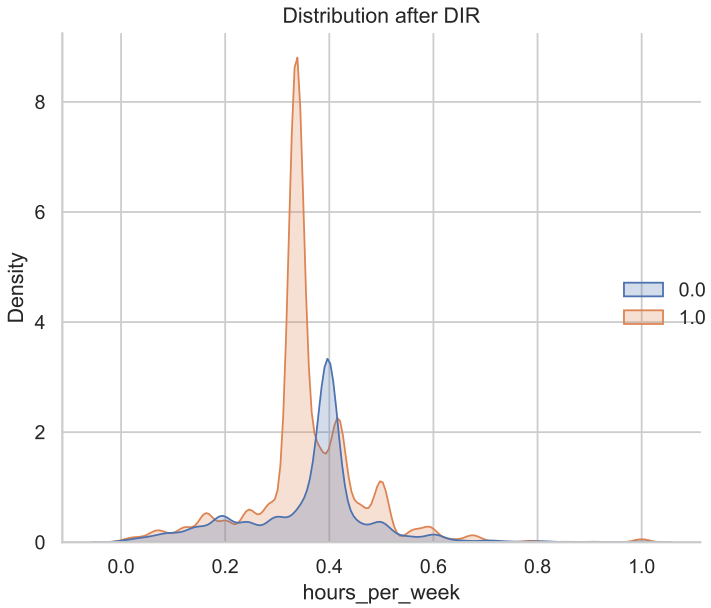
However, in order to work, this algorithm requires that most of the dataset variables are continuous. For example, figure 4.5 shows an application of the *DIR* algorithm to a dummy variable. In this case, we can see that the algorithm did not affect it since the variable has only two values, and so it is not possible to repair it, preserving the ranking. For this reason, if the dataset is mostly made of categorical, not orderable variables, this algorithm is not able to mitigate bias.

4.3.3 Sampling

Sampling [41] is a modified version of the *Reweighing* algorithm, in which weights are used to balance the dataset to remove discrimination. Sampling overcomes the limit of Reweighing i.e. that not all the classifiers consider weights during the

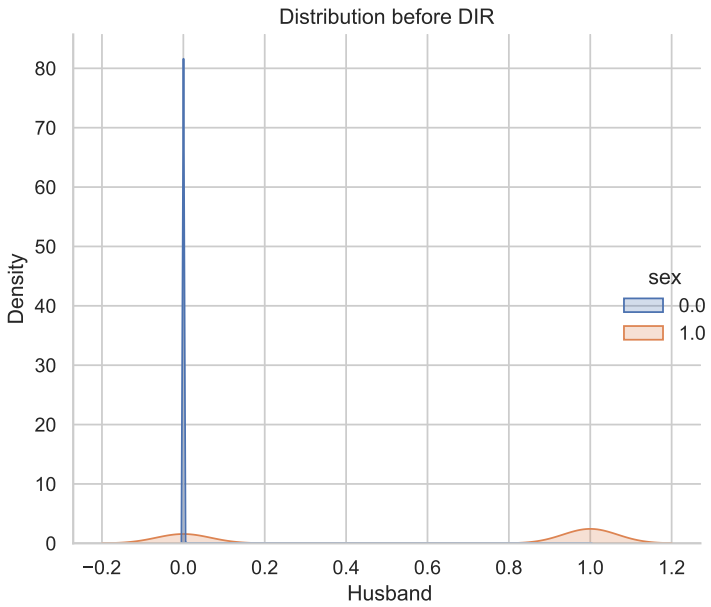


(a) Distribution of unprotected variable before DIR

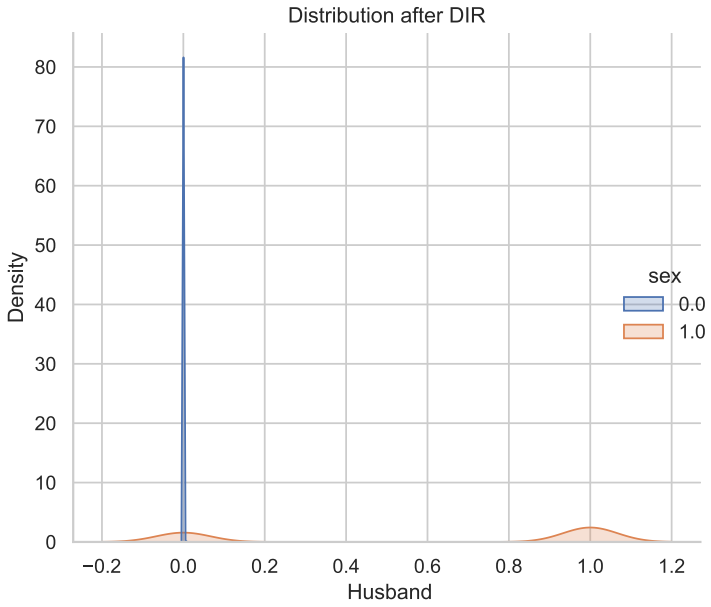


(b) Distribution of unprotected variable after DIR

Figure 4.4. Application of DIR to numerical variable



(a) Distribution of categorical variable before DIR



(b) Distribution of categorical variable after DIR

Figure 4.5. Application of DIR to categorical variable

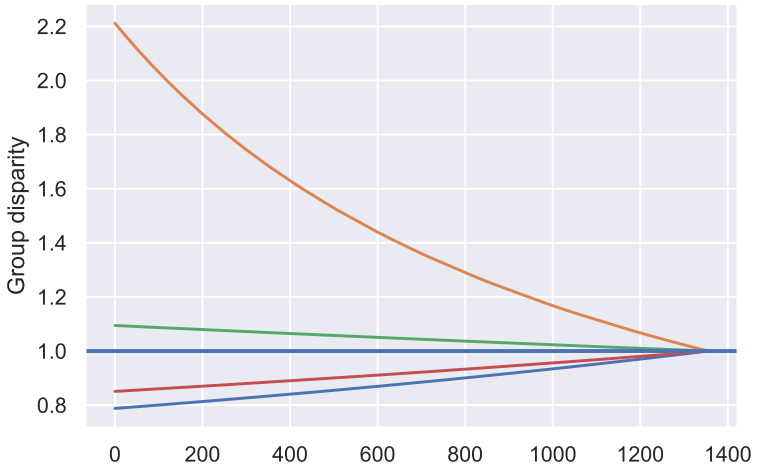


Figure 4.6. Sampling algorithm

learning process. This methods starts by partitioning the dataset in four groups: DP (*Deprived* group with *Positive* labels), DN (*Deprived* group with *Positive* labels), FP (*Favored* group with *Positive* labels) and FN (*Favored* group with *Negative* labels):

$$DP = \{X \in D | X(S) = b \wedge X(Class) = +\} \tag{4.6}$$

$$DN = \{X \in D | X(S) = b \wedge X(Class) = -\} \tag{4.7}$$

$$FP = \{X \in D | X(S) = w \wedge X(Class) = +\} \tag{4.8}$$

$$FN = \{X \in D | X(S) = w \wedge X(Class) = -\} \tag{4.9}$$

As for Reweighing, for each group, this algorithm computes the expected weight (W_{exp}) and the observed weight (W_{obs}). The ratio between these two values will be used to balance each group until the expected weight is reached. In particular, in the case of a biased dataset, DN and FP will have an observed weight higher than the expected weight, while DP and FN will be the vice-versa. In this case, the algorithm will randomly remove items from DN and FP and randomly duplicate items from DP and FN until the expected size is reached.

Figure 4.6 shows how the group disparity ($\frac{W_{exp}}{W_{obs}}$) of each group converges to one during the iterations of the algorithm. In particular, in about 1400 iterations the algorithm is able to balance the groups.

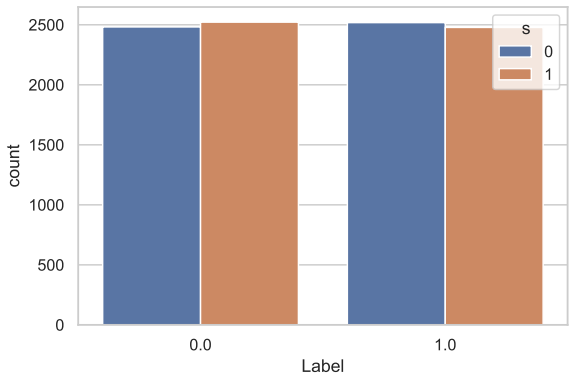


Figure 4.7. Distribution of labels for the unbiased dataset

4.4 Experimental comparison of classic debias algorithms

This section shows the results of the analysis we have done using the methodology described above. For this purpose, we have used the **aif360** [7] implementation of the fairness metrics, Reweighing and DIR, while Sampling has been implemented from scratch. We have then used: **Pandas**[8] and **Numpy**[9] for data manipulation, the **Scikit-learn**[10] implementation of the *Logistic Regression* classifier and other model evaluation methods and **Matplotlib**[11] and **Seaborn**[12] for data visualization. All the analysis has been done in Python.

4.4.1 Synthetic dataset

Figures 4.7, 4.8 and 4.9 shows the label and sensitive variable distribution respectively for the *unbiased*, *biased unbalanced* and *biased balanced* versions of the synthetic dataset.

Figure 4.10 shows the computed metrics for the three versions of the dataset. Concerning the unbiased dataset, we can see that all the methods are able to preserve the fairness of the classifier, leaving a high accuracy of predictions. Instead, about

[7] https://github.com/Trusted-AI/AIF360
[8] https://pandas.pydata.org/
[9] https://numpy.org/
[10] https://scikit-learn.org/stable/index.html
[11] https://matplotlib.org/
[12] https://seaborn.pydata.org/

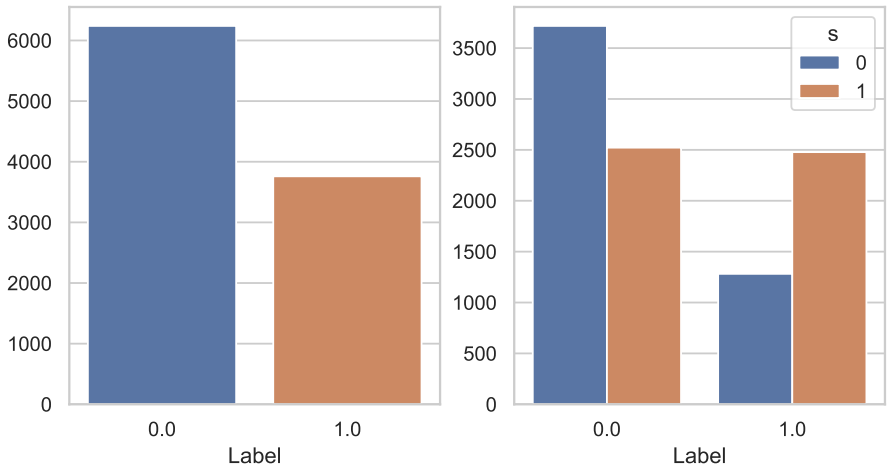


Figure 4.8. Unbalanced biased dataset

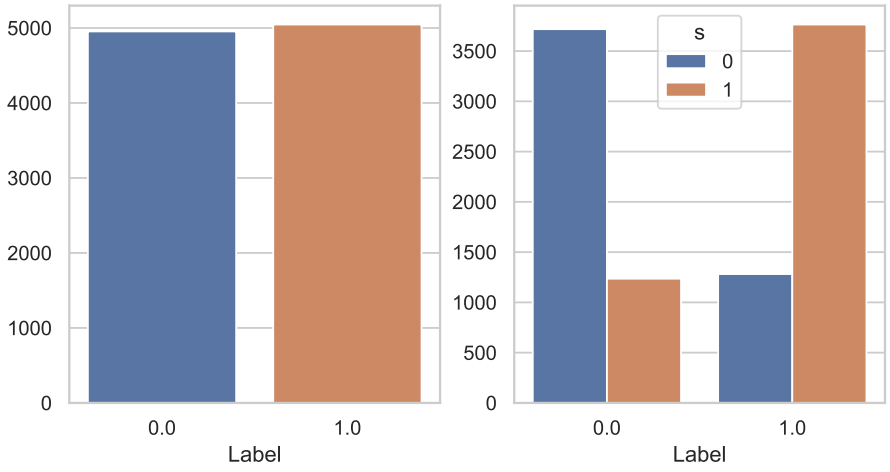


Figure 4.9. Balanced biased dataset

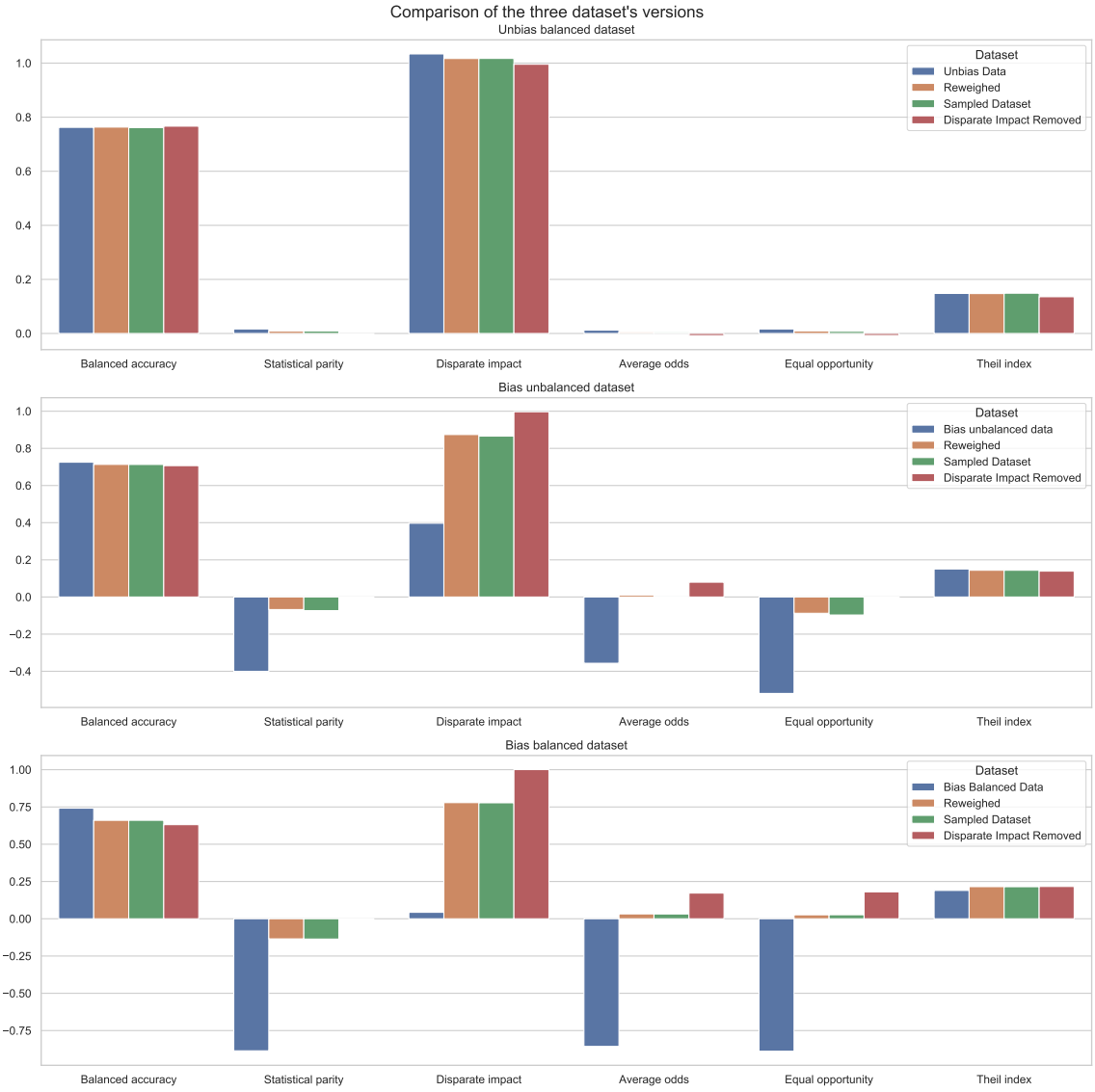


Figure 4.10. Metrics comparison for the three versions of the synthetic dataset

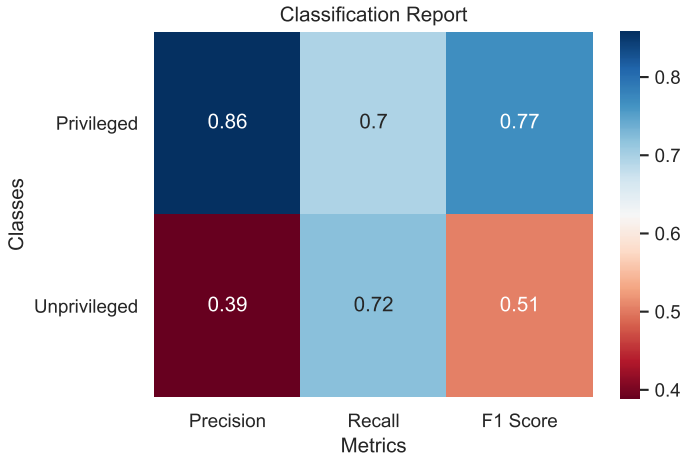


Figure 4.11. Classification report for the biased balanced dataset after the application of Reweighing

the biased datasets, it is worth noting that the higher the bias, the lower the model's accuracy after bias mitigation. This can be explained by the fact that, after bias mitigation, the model tries to assign positive labels with the same probability for both groups. But, if the items of the unprivileged group with positive labels are actually few, the model may create false positives for them. This fact is shown in the classification report of the classifier after the application of *Reweighing* in figure 4.11. As we can see, the Precision value for the unprivileged group is low, while the value of the Recall is high. This means that the model makes more false positives trying to predict positive labels for the unprivileged group.

Another thing worth noting from the metrics comparison is that the *DIR* algorithm performs better than *Reweighing* and *Sampling* in removing the bias. This improvement, however, can be explained by the removal of the sensitive variable from the dataset before training and testing the classifier in the case of *DIR* application. Since randomly generated variables make this dataset, the sensitive variable, which has been added to the dataset after its creation, is not correlated to any other dataset variable. For this reason, removing the sensitive variable before training and testing the model is sufficient, in this case, to improve the model's fairness. This fact is proven in figure 4.12, where we show the variables most correlated with the sensitive variable `s`. In particular, we show the variables with a value of the *Pearson*

Figure 4.12. Variables high correlated with the sensitive variable `s`

correlation index higher than 0.1. As we can see, only the label is highly correlated with `s`, and the bias present explains this in the dataset.

After this analysis, we combined *Reweighing* and *DIR* and check how this combination performs in mitigating bias. In particular, the combination of these methods has been implemented as follows:

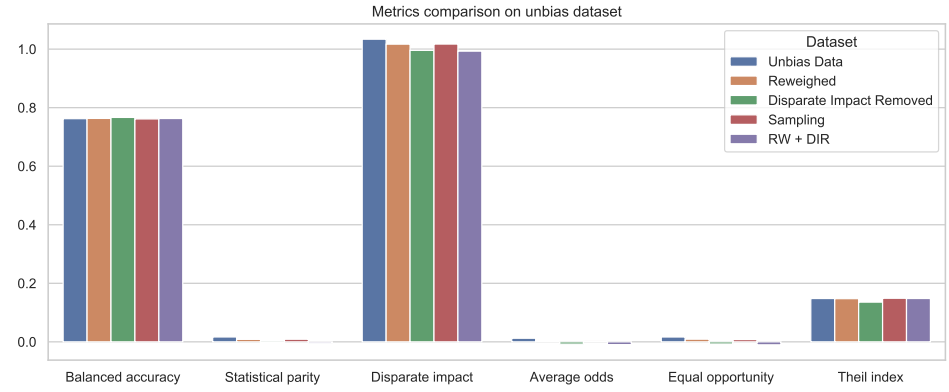
1. First, we compute the *DIR* algorithm to the biased dataset
2. Then, we apply *Reweighing* to the transformed dataset

The comparison of this method with the others for the three versions of the dataset can be seen in figure 4.13. As we can see, the performances of this method are compared with the ones of *DIR*. In particular, we may conclude that combining *Reweighing* and *DIR*, in this case, brings no advantage since we may obtain the same results applying only the *DIR* algorithm.

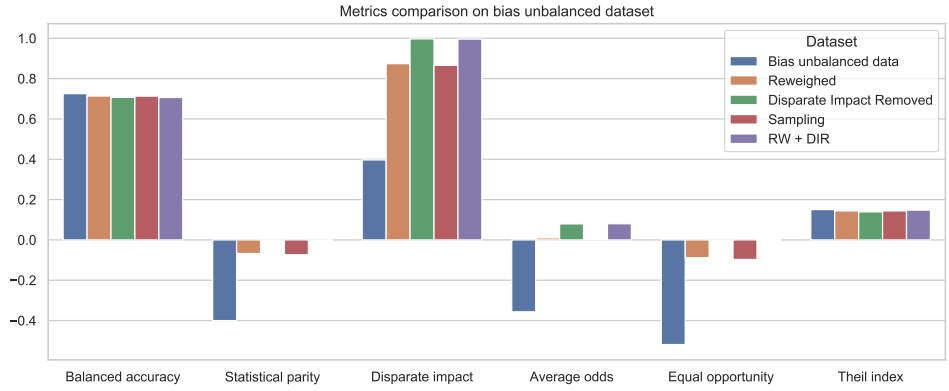
From this analysis, we have seen that all these methods can improve the fairness of the classifier. All the selected metrics, with the exception of the *Theil Index*, reflect this improvement. The reasons why the *Theil Index* is not affected by these methods can be found in the fact that this metric is an **individual fairness** metric. Our methods, instead, consider and improve **group fairness**, trying to make the classifier fair for both privileged and unprivileged groups. Finally, in this case, *DIR* performs better than other methods. Still, we have seen that this improvement is explained by the absence of correlation between the sensitive and unprotected ones.

4.4.2 Datasets with a single protected attribute

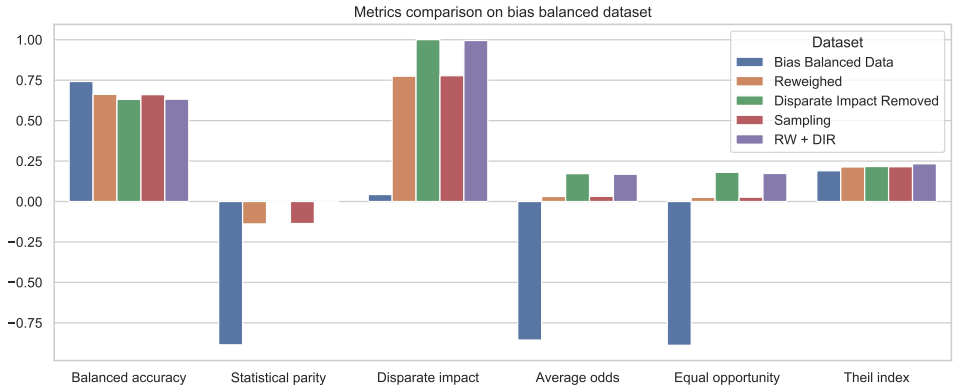
In this section, we describe the results of analysis conducted on real datasets known in the literature to be biased [53]. In particular, in this section we focus on datasets with a single protected variable. The selected datasets are: **Adult Income**, **Bank Marketing** and **German Credit**.



(a) Comparison for the unbiased dataset



(b) Comparison for the biased, unbalanced dataset



(c) Comparison for the biased, balanced dataset

Figure 4.13. Comparison of Reweighing + DIR with the other methods for the Synthetic Dataset

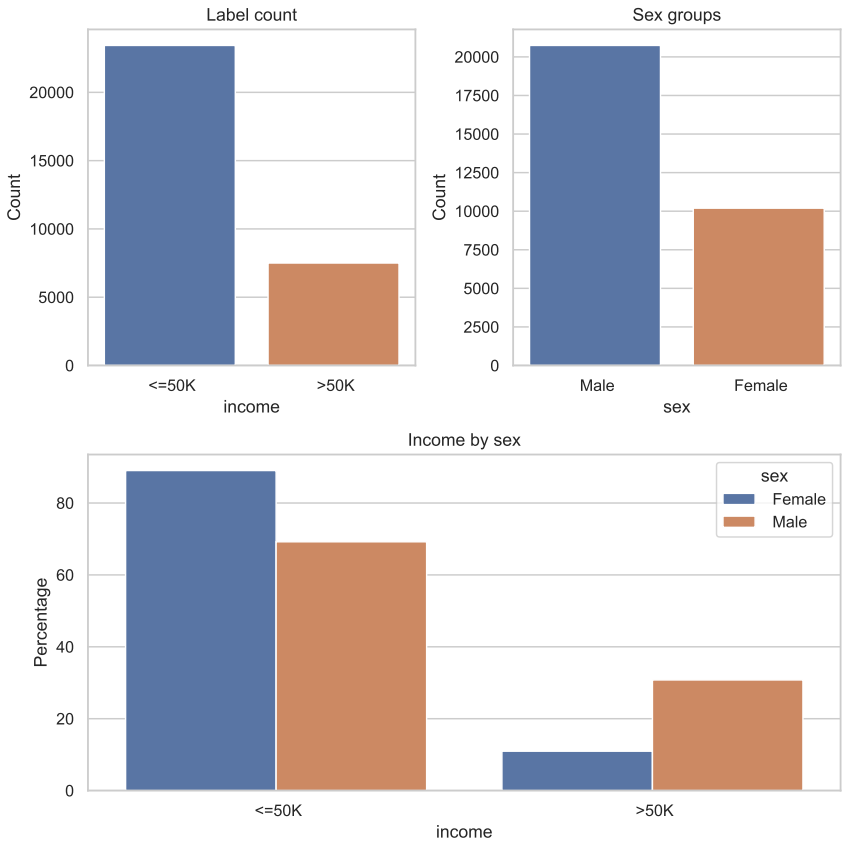


Figure 4.14. Distribution of features of the Adult Dataset

In the following, we describe the results of the analysis applied to each of them.

Adult Dataset

Figure 4.14 shows the distribution of the `sex` and `income` features and the distribution of `income` for the privileged and unprivileged groups. As we can see, the distribution of `income` for both groups is unbalanced. This could be a symptom of bias.

Figure 4.15 shows the performances of the algorithms for this dataset. As we can see, Sampling and Reweighing are able to improve the fairness of the classifier, while DIR performs worse. The worst performances of DIR can be explained by the fact that Adult is mostly made of categorical variables, which has been *one hot encoded* before being input to the classifier. *One Hot Encoding* is a pre-processing data transformation technique that transforms a categorical variable S to a set of n

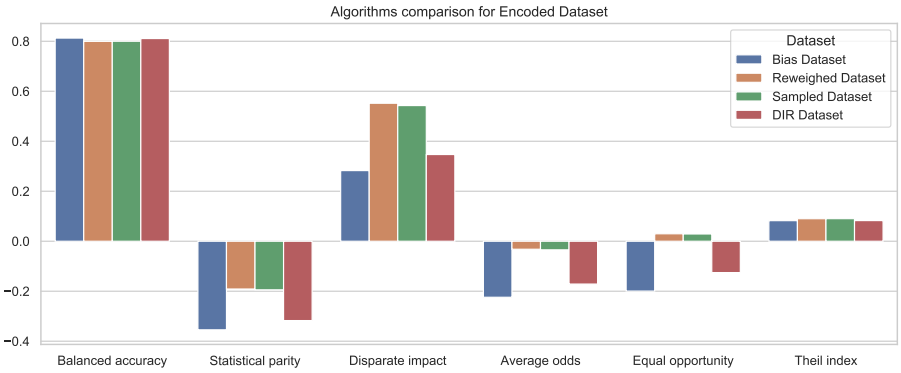


Figure 4.15. Metrics comparison for the Adult Dataset

dummy variables (variables whose values are zero or one), where n is the number of values of S. Each dataset item will have only one of the n variables set to one, i.e. the variable corresponding to the original value of S.

As for the synthetic dataset, we then tried to combine *Reweighing* and *DIR*. Figure 4.16 shows the comparison of all these methods for both one hot encoded and numerical datasets. As we can see in both figures, adding the *Reweighing* transformation to *DIR* gives us no advantages. In particular, we can see that for the one-hot encoded dataset, the performances of *Reweighing + DIR* are worse than *Reweighing* or *Sampling*. For the numerical dataset, instead, the performances are comparable to *DIR*.

One last thing worth noting is that when the methods can improve the classifier's fairness (like in the case of the numerical version of the dataset), the classifier's accuracy tends to decrease. This phenomenon has already been noticed for the synthetic dataset. It can be explained by the fact that the model is trying to predict positive labels with the same probability for both groups. But, in the case of the unprivileged group, it may predict more false positives.

Bank Marketing Dataset

Figure 4.17 shows the sensitive variable's and the label's distributions and how the label is distributed between the two groups. From the figure, we can see that, although the privileged group is much smaller than the other, the favourable label is

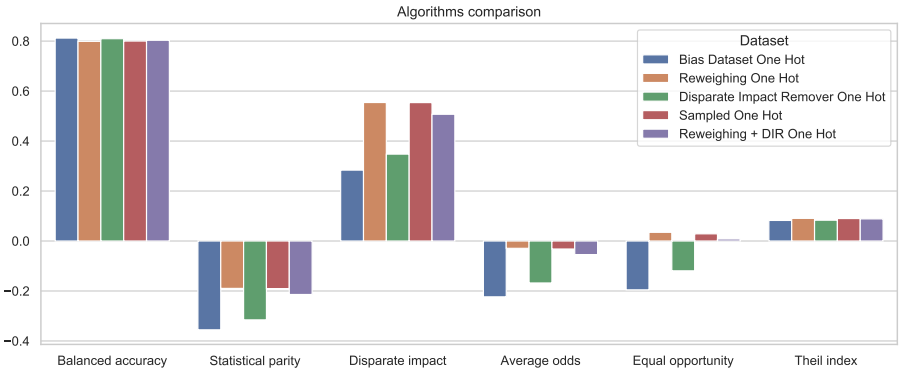


Figure 4.16. Reweighing + DIR comparison on the Adult Dataset

more present in the privileged group. This is a symptom of bias.

Figure 4.18 shows the performances of the algorithms for this dataset. Since this dataset is made mostly by categorical variables, DIR is not able to improve the fairness of the classifier. In this case, we are not able to consider only numerical variables (like we have done for the Adult) because they are too few for proper classification. Reweighing and Sampling instead can mitigate the classifier's bias, preserving the accuracy of the predictions. Combining *Reweighing* and *DIR*, also in this case, is of no advantage since it is not able to mitigate the bias and give us results comparable to *DIR*.

Finally, we can note that in this case, we are dealing with a dataset not much biased (like the Adult was), and so improving the fairness of the classifier does not have a big impact on the accuracy of the predictions.

German Credit Dataset

The last selected dataset with a single protected variable is the **German Credit** dataset. As we can see from figure 4.19, this dataset, differently from the others, is not particularly biased. However, we have selected it to see if the methods can also detect a small classifier's small bias. Figure 4.20 shows the metrics for this dataset. In this case, all the methods can mitigate the bias. Since the dataset is not only made of categorical variables, also *DIR* performs well. As before, combining Reweighing and DIR does not give particular advantages since the performances of

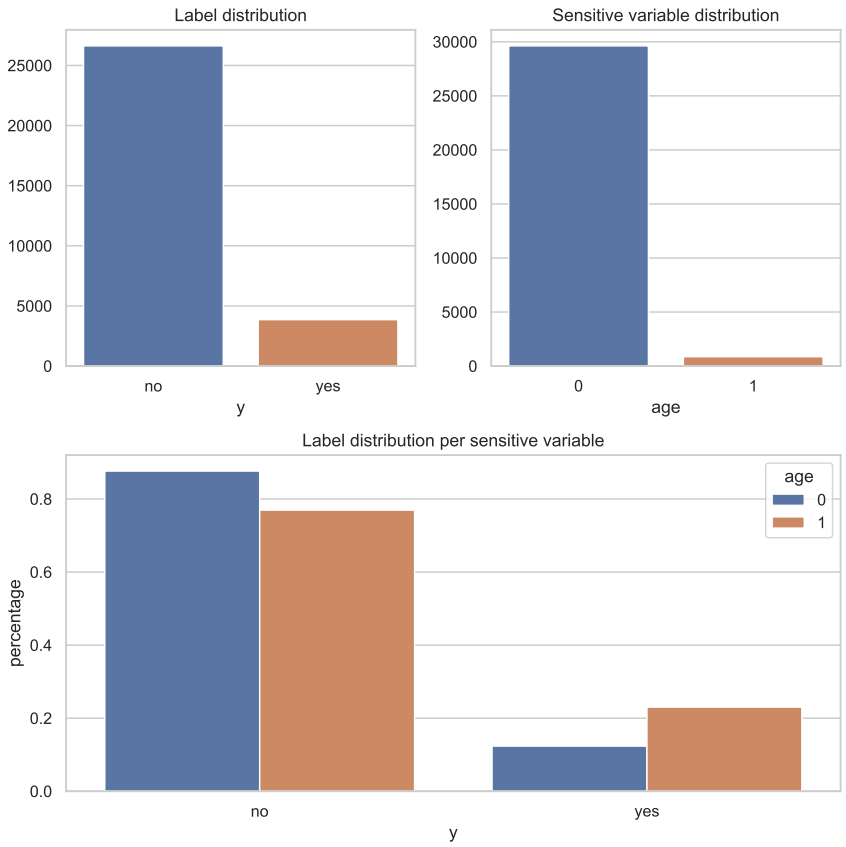


Figure 4.17. Distribution of sensitive variable and label of the Bank Dataset

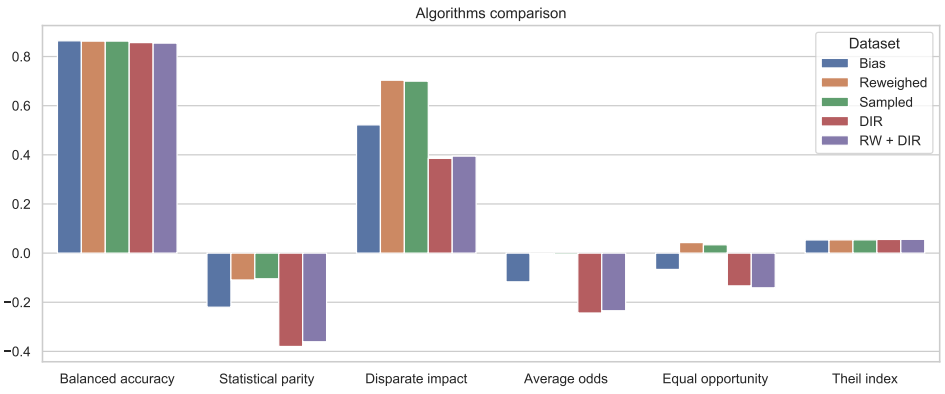


Figure 4.18. Metrics for Bank Dataset

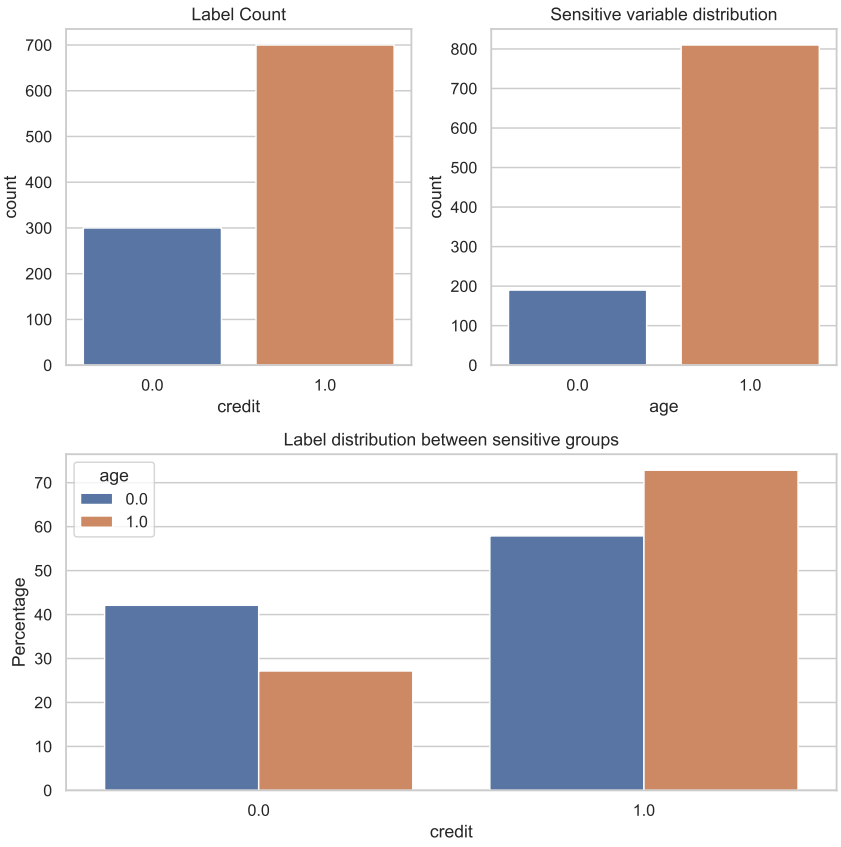


Figure 4.19. Distribution of label and sensitive variable of the German dataset

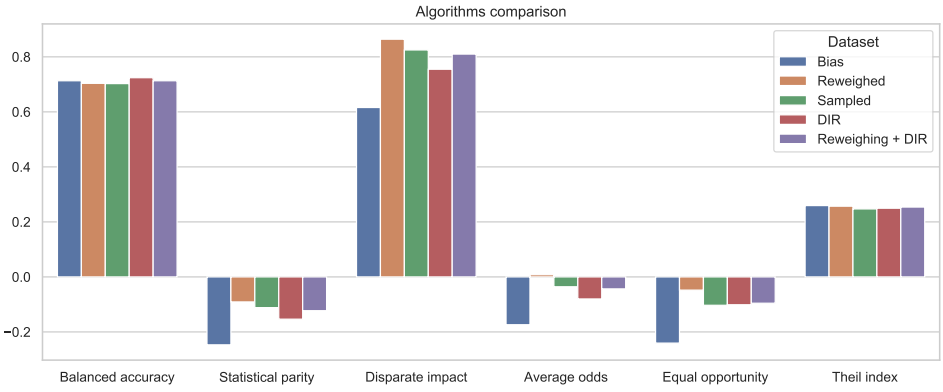


Figure 4.20. Metrics comparison for the German Dataset

Reweighing alone are better. Finally, because the original classifier's bias was not so high, applying bias mitigation methods does not impact the prediction's accuracy.

4.4.3 Overall considerations

These analyses on real datasets confirmed the hypothesis we have made on the synthetic dataset. In particular:

1. All the selected methods, if the dataset is suitable for their application, can improve the fairness of the classifier.

2. All the selected metrics, with the only exception of the *Theil Index*, reflecting the improvement of the classifier's fairness

3. Combining *Reweighing* and *DIR* is of no advantages since the performances we obtain are comparable with those of the single methods.

In addition, we have seen that *DIR* is not able to improve the fairness of the classifier if the dataset is mostly made of categorical bias. Instead, *Reweighing* and *Sampling* are not affected by this characteristic and so can be considered more stable.

4.5 Debiaser for Multiple Variables

Sampling was originally proposed in [41] only for datasets with a single protected variable, in which it was possible to identify the groups DP, DN, FP and FN. As a novel contribution, we have extended this algorithm to a multiple sensitive variable cases, proposing a new method, the *Debiaser for Multiple Variables (DEMV)*. In particular, we consider each group formed by the combination of each possible value of the sensitive binary variables and each value of the label. Formally, let S be the set of all the possible values $s_1, s_2, \ldots, s_n$ of the sensitive variables $S_1, S_2, \ldots, S_n$ and let $+, - \in L$ respectively the positive and negative labels of the dataset D. We define:

$$GP = \binom{S \cup \{+\}}{n+1} \tag{4.10}$$

$$GN = \binom{S \cup \{-\}}{n+1} \tag{4.11}$$

as the sets of all the possible combinations of the n sensitive variables with positive and negative labels respectively. The algorithm, then iteratively adds or removes items from each group $g = \{X \in D | X[S_1, S_2, \ldots, S_n, L] = c, c \in GP \vee GN\}$ until the expected size is reached.

The pseudo-code of the DEMV is given in the listing 1. This function is defined as a recursive function that iteratively calls itself, adding each time a new condition for the definition of the sampling group. This function takes as input the dataset D, the binary sensitive variables $S_1, \ldots, S_n$, the binary label L and some others parameters that are useful for the recursion: a counter i initially set to 0, an array G initially empty and a boolean condition initially set to *true*. Lines from 2 to 9 defines the base condition of the function. In particular, we check if the value of the counter i is equal to the number of sensitive variables. If so, we iterate the possible values of the label (which are 0 or 1) and create a group g made by the condition created during the recursion and the value of L. Then, we compute W_{exp} and W_{obs} like for *Reweighing* and then balance this group (using the algorithm defined in the listing 2). Finally, we add g to the array G (that we use to keep track of all the

sampled groups) and return G. Lines from 10 to 15 defines instead the recursive part of the function. In particular, if the value of i is not equal to the number of sensitive variables, we increment the value of i by one and append to G the result of two different recursive calls. These calls differ from each other only in the condition passed as input. In fact, in one call, we pass a condition equal to the previous one plus the value of S_i equal to 0, while in the second we pass a condition equal to previous one plus the value of S_i equal to one. Since we are dealing with binary attributes, these are the only possible values that the sensitive attributes can have. Defining the conditions in this way allows us to consider all the possible sampling groups. Finally, lines from 16 to 20 defines the returning condition of the functions. In particular, since we are dealing with binary attributes, the maximum number of samples obtainable from the combination of n sensitive variables plus the label is $2^{(n+1)}$. So, if the size of G is exactly equal to $2^{(n+1)}$, then it means that we have considered and balanced all the groups, and so we can return the final sampled dataset D_S. Otherwise, it means that we are in the middle of the recursive tree, and so we simply return G, that will be again merged with the result of other recursive functions.

Figure 4.21 shows how the algorithm builds the conditions for the sampling groups in case on $n = 2$. In particular, each node represents a new condition added to the initially *True* condition. We set the initial condition to *True* because it is unaffected by the *And* conjunctions. Each level of the tree represents a new recursion step where the value of the `i` counter is incremented by one. It is worth noting that the value of the counter is incremented before starting a new recursion call. So, in the root node `i` will be equal to zero and then will be incremented before making the recursive call. The leaf of the tree are the groups that will be balanced and then re-merged to make the unbiased dataset. In particular, each group will be merged with his sibling when the recursive calls end and we start traversing back the tree.

Algorithm 2 is instead the pseudo-code of the *group balancing* function. We show this algorithm separately from the *sampling* because it is independent from the way a group has been created and can be used also with other group creation policies. This function takes as input a group g, the expected size W_{exp} and the observed

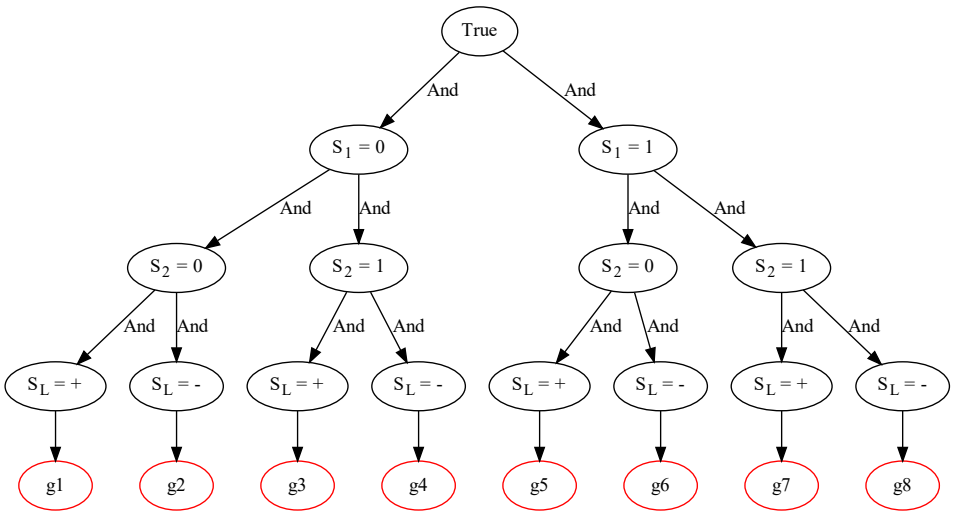


Figure 4.21. Sampling recursion tree for $n = 2$

size W_{obs} and returns the same group g balanced. In particular, the core of this function is the while loop that goes from line 1 to line 6. This loop checks if the ratio between the expected size and the observed size is equal to one (this means that the size of the group is equal to the expected one). If not, then two cases are possible. If the ratio is greater than one, it means that the size of the group is less than the expected one. In this case, we randomly duplicate an item from the same group g. Otherwise, if the ratio is less than one, it means that the size of the group is greater than the expected one. In this case, we randomly remove an item from the group. Finally, when the group is balanced, we simply return it.

Figure 4.22 shows an example application of *DEMV*. In particular, in this case there were two sensitive attributes and we can see that eight groups has been created. These groups are generated by the combination of the values of the two variables and the values of the label (2^{2+1}). The different shape of the curve compared to figure 4.6 is caused by the rounding applied to the ratio between the expected and observed size in order to converge to one.

Algorithm 1: Debiaser for Multiple Variables

Input: (Dataset D, Binary sensitive variables $S_1, S_2, \ldots, S_n$, Binary label L, $i = 0$, $G = []$, condition=$true$)

Output: Sampled dataset D_S

$n = length(\{S_1, S_2, \ldots, S_n\})$
if $i == n$ **then**
  foreach $l \in L$ **do**
    $g = \{X \in D | \text{ condition} \wedge L == l\}$
    $W_{exp} = \frac{|\{X \in D | \text{condition}\}|}{|D|} \cdot \frac{|\{X \in D | L == l\}|}{|D|}$
    $W_{obs} = \frac{|g|}{|D|}$
    balance g using W_{exp} and W_{obs}
    add g to G
  return G
else
  $i = i + 1$
  $G_1 = sample(D, S_1, \ldots, S_n, i, G, \text{condition} = condition \wedge S_i == 0)$
  $G_2 = sample(D, S_1, \ldots, S_n, i, G, \text{condition} = condition \wedge S_i == 1)$
  append G_1 to G
  append G_2 to G
  if $length(G) == 2^{(n+1)}$ **then**
    $D_S =$ merge all $g \in G$
    return D_S
  else
    return G

Algorithm 2: Group balancing algorithm

Input: (Group g, Expected size W_{exp}, Observed size W_{obs})

Output: Balanced group g

```
while W_exp \ W_obs ! = 1 do
    if W_exp \ W_obs > 1 then
        randomly duplicate item i in g
    else if W_exp \ W_obs < 1 then
        randomly remove item i from g
    recompute W_obs
return g
```

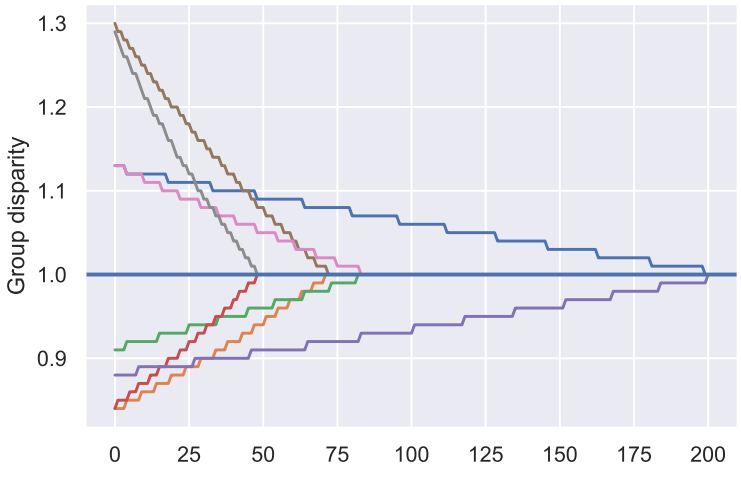


Figure 4.22. Application of *DEMV* on a real dataset

4.5.1 Comparison with established methods

Now, we describe the analysis made on datasets with more protected variables. In particular, we check if the *Debiaser for Multiple Variables* is able to mitigate the bias as well as the other established methods. In addition, we want to evaluate the ability of the method to mitigate the bias towards more complex groups identified by the combination of more sensitive values.

In order to better compare the performances of our method, we also tested the classic *Sampling* algorithm using a modified version of the selected datasets. Specifically, we added a new variable to the dataset called `flag`, which is equal to 0 if all the sensitive variables are equal to 0 and 1 otherwise. Formally, `flag` is defined as follows:

$$flag = \begin{cases} 0 \text{ if } S_1, S_2, \ldots, S_n = 0 \\ 1 \text{ otherwise} \end{cases} \tag{4.12}$$

This variable act as a flag, indicating if we are dealing with a member of the unprivileged group (`flag` $= 0$) or not (`flag` $= 1$). Classic *Sampling* was then applied using this variable as the sensitive variable for creating balancing groups.

The selected datasets are the following: **ProPublica Recidivism (COMPAS)** dataset, an extended version of the **German Credit** dataset which considers also the `sex` variable and an extended version of the **Adult Income** dataset which considers also the `race` variable

COMPAS

Figure 4.23 shows the distribution of the label and the sensitive variables in the dataset and the distribution of the label between the two sensitive groups. As we can see, the probability of recidivism is higher for non-caucasian men. Figure 4.24 shows the performances of the methods for this dataset. Before commenting on them, it is worth noting that, in the case of multiple sensitive variables, [25] suggests applying the *DIR* transformation to the joint probability distribution of the sensitive variables. For this reason, only for *DIR* we have first computed the joint probability distribution of the sensitive variables and then applied the fairness metrics using this new variable as a reference. From the picture, we can see that *Sampling* performs

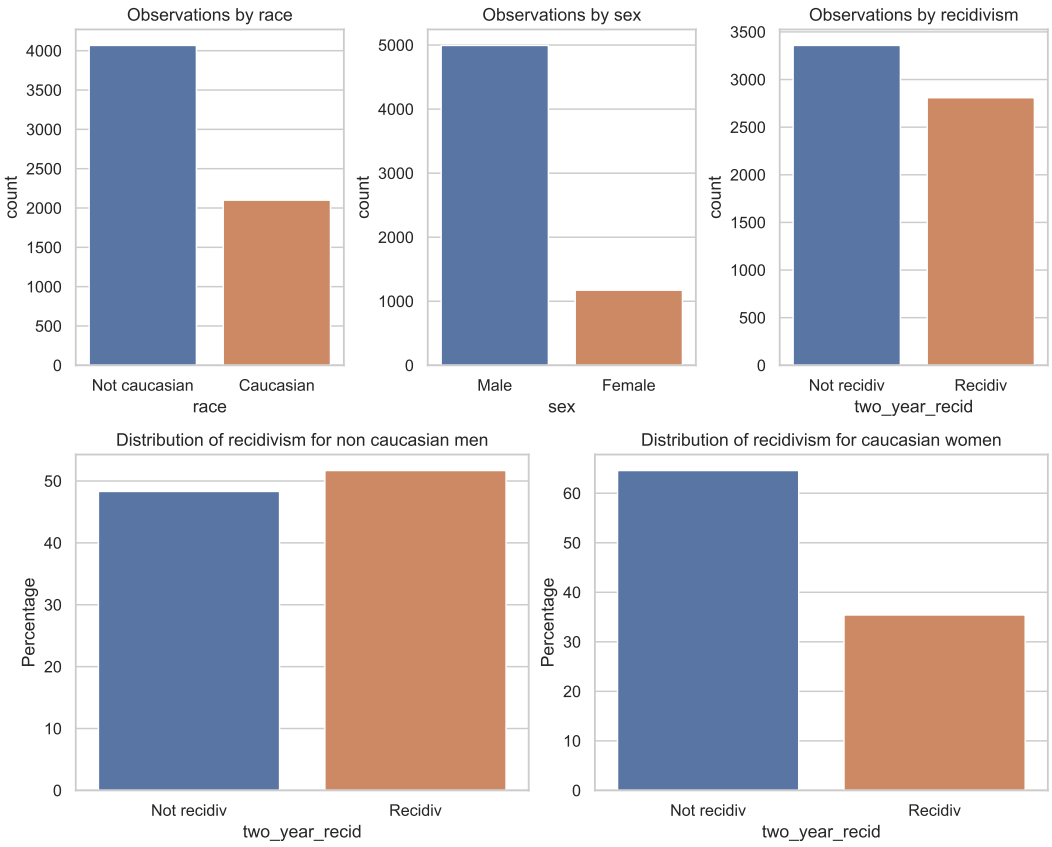


Figure 4.23. COMPAS label distributions

very well in removing bias, while *Reweighing* behaves a little worse. Instead, *DIR* tends to reverse the bias of the classifier favouring the previously unprivileged group, and the same is for the combination of *Reweighing* and *DIR*. Comparing DEMV with the classic *Sampling* version we can see that our method performs better for all the metrics. In fact, DEMV is the method performing better of all, reaching values that are near to the optimal ones. Finally, we can see that since the original bias is not so high, the *Balanced accuracy* is not much affected by all the bias mitigation methods.

German Credit Dataset

Figure 4.25 shows the distribution of labels and sensitive variables for this dataset. From the figure, we can see that there is not a high bias in data. Figure 4.26 shows the performances of the methods. As for the COMPAS dataset, also in this case

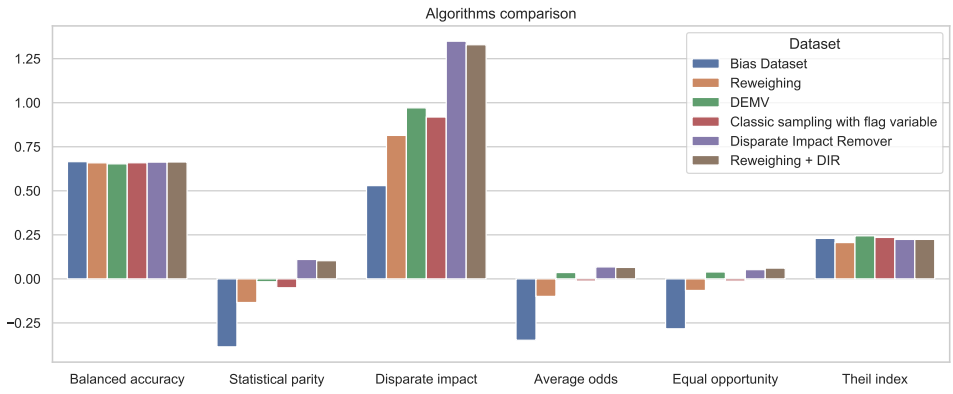


Figure 4.24. COMPAS metrics comparison

we can see that DEMV is the method performing better on all metrics, reaching values that indicate the absence of bias. *Reweighing* and *DIR* are the methods that perform worse and combining them lead us to even worse results. Finally, we can see that, since the bias is not very high, all the methods do not impact the *Balanced Accuracy* much. As for the other cases, since DEMV is the method improving more the fairness, it is also the method that impact more on the *Balanced Accuracy*.

Adult Dataset

From figure 4.27, we can see that the number of black women having an income higher than 50k dollars a year is minimal. This is a symptom of high bias in the data. Figure 4.28 confirms this hypothesis. From this picture, we can see that the bias of the classifier trained with unprocessed data is very high. Like for the single variable case, the bias of this dataset is more difficult to improve. In particular, we can see that all the methods are not able to improve the fairness much. However, we can see that DEMV is the method performing better in all metrics together with *Reweighing*. Specifically, we can see that the values of all the metrics are almost the same. Like for the single variable case, *DIR* is not able to improve the fairness of the classifier, since the dataset is mostly made of categorical variables. As for the other cases, combining *Reweighing* and *DIR* does not help since it gives performances that are at most comparable to the other methods. Finally, comparing our method with the classic version of *Sampling*, we can see that it performs better on all metrics.

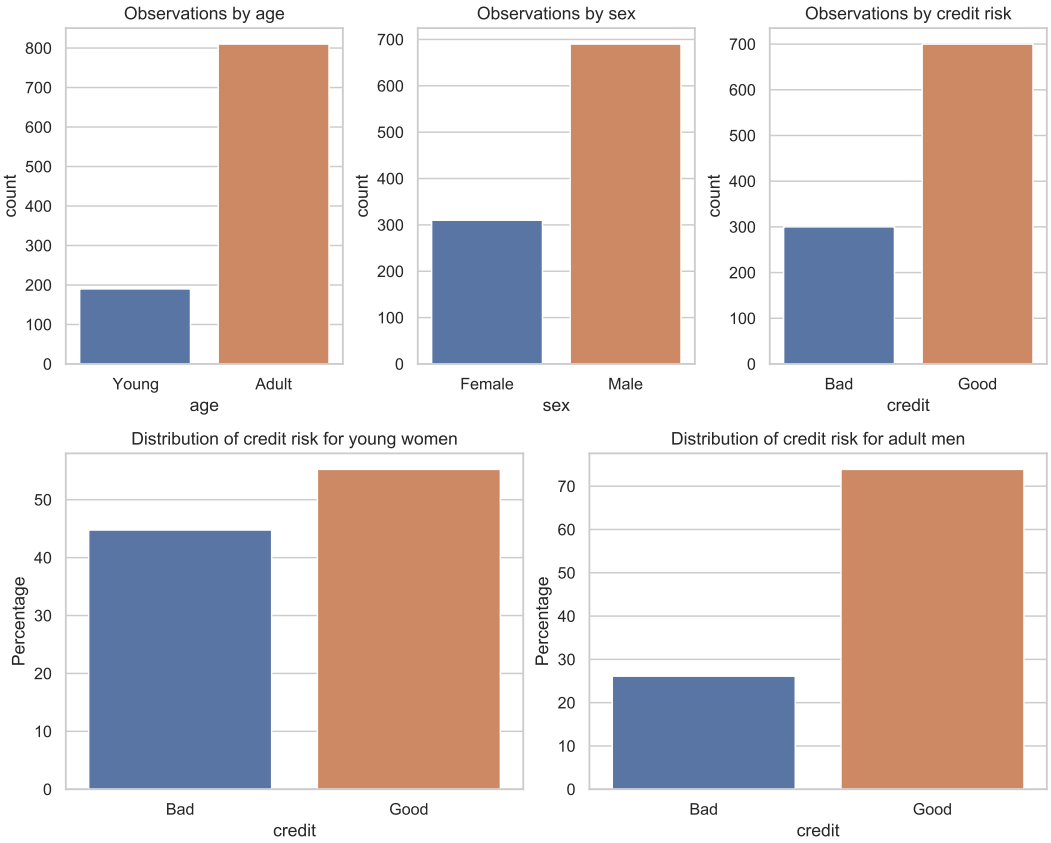


Figure 4.25. Distribution of sensitive variables and label for the German Credit

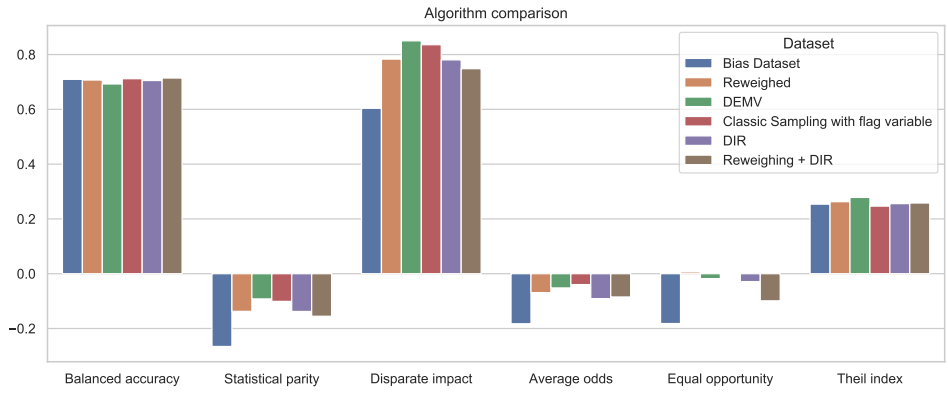


Figure 4.26. Methods performances for German Credit

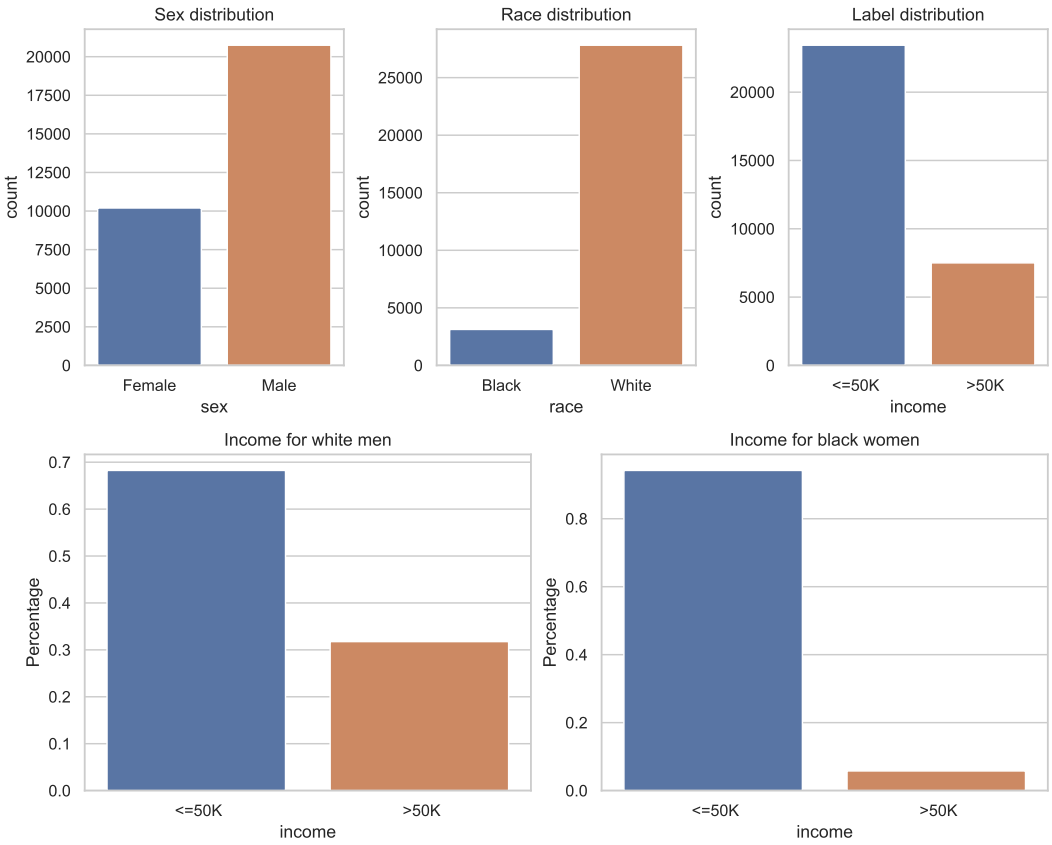


Figure 4.27. Distribution of sensitive variables and label for the Adult Income dataset

In particular, concerning *Statistical Parity* and *Disparate Impact* we see that the difference between the two methods is more marked.

4.5.2 Overall considerations

These analyses on datasets with a more complex bias confirmed some of the single sentive variable dataset's hypotheses. Still, they also provide us with some more points of analysis. In particular, we can confirm that all the metrics, except the *Theil Index*, reflect the improvements in the classifier's fairness after applying all the methods. Again, we can confirm that combining *Reweighing* and *DIR* does not give us performances that are better than the ones obtained by the single methods. However, we have seen that, in the case of multiple sensitive variables, the method performing better in our experiments is the DEMV, which is our proposed extension of the *Sampling* algorithm proposed in [41]. In particular, we have seen that this

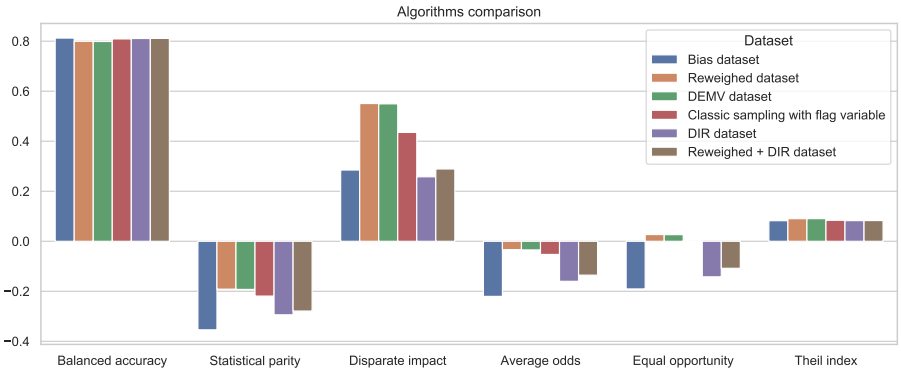


Figure 4.28. Metrics comparison for the Adult Dataset

method performs better than the standard *Sampling* with a flag variable. *Reweighing* is also able to improve the fairness of the classifier, but it is more difficult in case of high bias. Instead, *DIR* confirms not being able to improve the fairness if the dataset is mostly made of categorical variables. Otherwise, if the variables are mostly numerical, *DIR* is able to improve also a high level of bias, but it may also tend to a reverse bias, where the original privileged group became unprivileged and vice-versa.

4.6 Final considerations

These analyses conducted on several heterogeneous datasets allowed us to make the following conclusions:

1. The *Theil Index* is the only metric not able to reflect the improvements in the fairness of the classifier. This behaviour can be explained by the fact that this metric is an *individual fairness* metric. Our methods, instead, consider *group fairness* improvements. Indeed, the other metrics are *group fairness* metrics, and they are all able to reflect the fairness improvements.

2. Concerning the *group fairness* metrics, regardless of their belonging to the *WAE* or *WYSIWYG*, they are all able to reflect the changes in the fairness of the classifier.

3. Concerning datasets with a single sensitive variable, we may conclude that

Reweighing and *Sampling* are more suited than *DIR* for bias mitigation. *DIR* is able to improve the fairness of the classifier only if the dataset is not made mostly of categorical variables. At the same time, *Reweighing* and *Sampling* are not affected by the type of variables of the dataset.

4. Concerning datasets with multiple sensitive variables, DEMV has been proven to be the best method for improving the fairness. At the same time, *Reweighing* has more difficulty in improving the fairness if the original bias of the classifier is very high. *DIR* is confirmed not able to improve the fairness if the dataset is mostly categorical.

5. Our custom extension of *Sampling* (DEMV) is able to improve the fairness of the classifier in case of multiple sensitive variables, also in case of high original bias. This method is also able to keep a good level of accuracy of the classifier.

6. Concerning the dataset's fairness metrics (not shown during the description of the analyses), *Reweighing*, *Sampling* and *DEMV* are able to improve them, while *DIR* does not affect them.

7. For all the datasets, we have seen that the more a method is able to improve the original fairness of the model, the more it lows the accuracy of the predictions. This trade-off can be explained by the fact that, after bias mitigation, the model is trying to classify all the groups with the same probability, but this can lead to a higher number of false positives and false negatives.

8. In general, combining *Reweighing* and *DIR* is of no advantage since we are able to obtain the same or better results with at least one of the other methods.

Reconnecting with the working context of chapter 3, we may conclude that the *group fairness* metrics are all able to describe the fairness of the classifier in a proper way inside the pipeline. *Reweighing*, *Sampling* and *DEMV* seemed to be more stable methods for improving the fairness of the classifier since they are not affected by any particular characteristic of the variables. *Reweighing* has the downside that it requires the classifier to consider weights related to data, while *Sampling* and *DEMV* does not. *Sampling* and *DEMV*, instead, may delete useful items during

the group sampling process, but from the analysis of the accuracy, it seemed that it never affected the *Quality of predictions* QR. For this reason, a user may choose to use *Reweighing* if the classifier accepts weights or *Sampling* (*DEMV* in case of multiple sensitive variables) instead. *DIR* may be considered as well, but only for datasets with mostly numerical variables. Finally, concerning the *Computational complexity* QR, *Reweighing* may be considered linear in the number of items of the dataset since it only applies a value to all the dataset's items. *Sampling* and *DEMV* instead requires iterating each group multiple times in order to re-balance them properly. This process may require much time if the groups are many and are very unbalanced. For this reason, *Reweighing* may be considered a preferable choice if computational complexity is an issue and the machine learning classifier is suitable, otherwise both *Sampling* in case of single sensitive variable and *DEMV* in case of multiple sensitive variables, are able to mitigate the bias in a proper way in any of the cases described above.

Chapter 5

Conclusions and future works

In this thesis, we have analysed the quality characteristics of DS workflows and made an in-depth analysis of the *Bias and Fairness* of machine learning systems. First of all, we have identified the QR and shown how they can be used to define our concept of *Quality* in DS pipelines. Then, we gave a high-level example of how the selected requirements can actually influence the user's decisions in all the steps of the workflow. Then, we made an in-depth study of one of the selected QR, i.e. the *Bias and Fairness* of machine learning models. First, we have made a survey of the different definitions, metrics and mitigation approaches existing in the literature. Then, we selected three established pre-processing methods to improve the fairness of the classifiers, namely *Reweighing*, *Sampling* and *Disparate Impact Remover (DIR)*, and compared their performances on several datasets using some selected fairness metrics. We made an extensive experimental analysis of such methods in order to identify the strengths and limits of all of them. From this analysis, we concluded that *Reweighing* and *Sampling* are the more suited methods for bias mitigation in case of single sensitive variable since they are more stable and are not affected by any characteristic of the dataset. As a novel contribution, we have then proposed and tested the *Debiaser for Multiple Variables (DEMV)*, an extension of the *Sampling* algorithm for datasets with multiple sensitive variables. We provided a mathematical formulation and the pseudo-code of it. Finally, we have compared our new method with the other established algorithms obtaining comparable or even better results.

5.1 Future works

The future works are manifold and can be distinguished in works related to the *High Quality* and works related to *Bias and Fairness*. Concerning the *High Quality* of the DS pipelines, from the analysis of such workflows, we can conclude that they can be profitably modelled as product line architectures. Over the last two decades, software product lines architectures have been successfully used in industry for building families of systems of related products, maximizing reuse, and exploiting their variable and configurable options [15]. Product line architectures act as a blueprint for building software, allowing stakeholders to define a common set of *features* and *variation points* [8]. A *feature* can be defined as a *component*, a building block that performs a specific task, that can be combined with other components to create a more complex system. *Variation points* are the elements that distinguish a final product from another. Based on the user's requirements, *variation points* may add, change or remove a *feature* from the product line architecture. DS workflows have a similar structure in terms of general tasks (features) to accomplish. At the same time, the final implemented pipeline can be very different depending on the QR, defined by the user, that will impose extra steps (variation points) needed to satisfy them. From this observation, it is natural to model DS workflow as a product line architecture. In fact, using this formalism allows us to define some features and variation points driven by the defined requirements. As a long term work, we aim to implement the described modelling framework. The required working steps are many: from an in-depth study of all the requirements and metrics to the definition of a DS workflow formalism that extends the product line architecture and embeds the defined quality metrics in the variation points. The final step will be the implementation of the described DS pipelines using model-driven techniques. The final modelling framework will help experts to build their pipelines satisfying, high-quality requirements.

Concerning *Bias and Fairness*, we plan to extend our experimental analysis to a larger set of cases. First of all, we want to see how the size of the dataset impacts the *Reweighing* and *Sampling* methods since both algorithms relies on the dimensions of both privileged and unprivileged groups. Then, we aim to see how feature reduction

techniques (such as *PCA* [40]) may improve the performances of *DIR* on datasets with mostly categorical variables. Finally, we want to extend our analysis also on tasks different from classification, such as regression or natural language processing. Our final goal is to use the results of these analyses for the implementation of our modelling framework in order to select the best methods to measure and improve the fairness for any type of task.

Bibliography

[1] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. Kepler: an extensible system for design and execution of scientific workflows. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.*, pages 423–424, 2004.

[2] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica, May*, 23(2016):139–159, 2016.

[3] Ricardo Baeza-Yates. Bias on the web. *Commun. ACM*, 61(6):54–61, May 2018.

[4] Ricardo Baeza-Yates. Bias on the web. *Communications of the ACM*, 61(6):54–61, 2018.

[5] Writuparna Banerjee. Train/test complexity and space complexity of logistic regression, Aug 2020.

[6] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *Nips tutorial*, 1:2, 2017.

[7] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *Calif. L. Rev.*, 104:671, 2016.

[8] Don Batory. Product-line architectures. In *Smalltalk and Java Conference*, volume 12. Citeseer, 1998.

[9] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy,

John T. Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. AI fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *CoRR*, abs/1810.01943, 2018.

[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*, page 0049124118782533, 2018.

[11] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. Knime - the konstanz information miner: Version 2.0 and beyond. *SIGKDD Explor. Newsl.*, 11(1):26–31, November 2009.

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

[13] Leonardo Candela, Valerio Grossi, Paolo Manghi, and Roberto Trasarti. A workflow language for research e-infrastructures. *International Journal of Data Science and Analytics*, pages 1–16, 2021.

[14] Longbing Cao. Data science: A comprehensive overview. *ACM Comput. Surv.*, 50(3), June 2017.

[15] Rafael Capilla, Jan Bosch, Pablo Trinidad, Antonio Ruiz-Cortés, and Mike Hinchey. An overview of dynamic software product line architectures and techniques: Observations from research and industry. *Journal of Systems and Software*, 91:3–23, 2014.

[16] Simon Caton and Christian Haas. Fairness in machine learning: A survey, 2020.

[17] Lianping Chen, Muhammad Ali Babar, and Bashar Nuseibeh. Characterizing architecturally significant requirements. *IEEE Software*, 30(2):38–45, 2013.

[18] Giovanni Luca Ciampaglia, Azadeh Nematzadeh, Filippo Menczer, and Alessandro Flammini. How algorithmic popularity bias hinders or promotes quality. *Scientific reports*, 8(1):1–7, 2018.

[19] Bo Cowgill and Catherine Tucker. Algorithmic bias: A counterfactual perspective. In *Workshop on Trustworthy Algorithmic Decision-Making*. NSF Trustworthy Algorithms, Arlington, VA, 2017.

[20] Derek Doran, Sarah Schulz, and Tarek R Besold. What does explainable ai really mean? a new conceptualization of perspectives. *arXiv preprint arXiv:1710.00794*, 2017.

[21] Flavio du Pin Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Data pre-processing for discrimination prevention: Information-theoretic optimization and analysis. *IEEE Journal of Selected Topics in Signal Processing*, 12(5):1106–1119, 2018.

[22] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[23] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[24] Inc. Elder Research. Statistical cognitive biases in data science: What is bias?, Jul 2017.

[25] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.

[26] Ian Foster, Rayid Ghani, Ron S Jarmin, Frauke Kreuter, and Julia Lane. *Big Data and Social Science: Data Science Methods and Tools for Research and Practice*. CRC Press, 2020.

[27] Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. On the (im)possibility of fairness, 2016.

[28] Batya Friedman and Helen Nissenbaum. Bias in computer systems. *ACM Trans. Inf. Syst.*, 14(3):330–347, July 1996.

[29] Randall Fulton and Roy Vandermolen. *Airborne Electronic Hardware Design Assurance: A Practitioner's Guide to RTCA/DO-254*. CRC Press, 2017.

[30] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4), June 2010.

[31] Yolanda Gil, Varun Ratnakar, Jihie Kim, Pedro Gonzalez-Calero, Paul Groth, Joshua Moody, and Ewa Deelman. Wings: Intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems*, 26(1):62–72, 2010.

[32] Yolanda Gil, Pedro Szekely, Sandra Villamizar, Thomas C Harmon, Varun Ratnakar, Shubham Gupta, Maria Muslea, Fabio Silva, and Craig A Knoblock. Mind your metadata: Exploiting semantics for configuration, adaptation, and provenance in scientific workflows. In *International Semantic Web Conference*, pages 65–80. Springer, 2011.

[33] Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):1–13, 2010.

[34] P. M. Goncalves Jr. and R. S. M. Barros. Automating data preprocessing with dmpml and kddml. In *2011 10th IEEE/ACIS International Conference on Computer and Information Science*, pages 97–103, 2011.

[35] Sara Hajian and Josep Domingo-Ferrer. A methodology for direct and indirect discrimination prevention in data mining. *IEEE transactions on knowledge and data engineering*, 25(7):1445–1459, 2012.

[36] Harlan Harris, Sean Murphy, and Marck Vaisman. *Analyzing the analyzers: An introspective survey of data scientists and their work.* " O'Reilly Media, Inc.", 2013.

[37] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[38] Vasileios Iosifidis, Besnik Fetahu, and Eirini Ntoutsi. Fae: A fairness-aware ensemble framework. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1375–1380. IEEE, 2019.

[39] Heinrich Jiang and Ofir Nachum. Identifying and correcting label bias in machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 702–712. PMLR, 2020.

[40] Ian Jolliffe. *Principal Component Analysis*. American Cancer Society, 2005.

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

[42] Ron Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

[43] Emmanouil Krasanakis, Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, and Yiannis Kompatsiaris. Adaptive sensitive reweighting to mitigate bias in fairness-aware classification. In *Proceedings of the 2018 World Wide Web Conference*, pages 853–862, 2018.

[44] Paritosh Kumar. Time complexity of ml models, Jun 2021.

[45] Matt J Kusner, Joshua R Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *arXiv preprint arXiv:1703.06856*, 2017.

[46] Maximilian A. Köhl, Kevin Baum, Markus Langer, Daniel Oster, Timo Speith, and Dimitri Bohlender. Explainability as a non-functional requirement. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 363–368, 2019.

[47] Chee Sun Liew, Malcolm P Atkinson, Michelle Galea, Tan Fong Ang, Paul Martin, and Jano I Van Hemert. Scientific workflows: moving across paradigms. *ACM Computing Surveys (CSUR)*, 49(4):1–39, 2016.

[48] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2021.

[49] Ji Liu, Esther Pacitti, Patrick Valduriez, and Marta Mattoso. A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, 13(4):457–493, 2015.

[50] Octavio Loyola-González. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE Access*, 7:154096–154113, 10 2019.

[51] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system. *Concurrency and computation: Practice and experience*, 18(10):1039–1065, 2006.

[52] Binh Thanh Luong, Salvatore Ruggieri, and Franco Turini. k-nn as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 502–510, 2011.

[53] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*, 2019.

[54] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[55] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[56] Randal S. Olson and Jason H. Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Proceedings of the Workshop on Automatic Machine Learning*, volume 64 of *Proceedings of Machine Learning Research*, pages 66–74, New York, New York, USA, 24 Jun 2016. PMLR.

[57] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Emre Kıcıman. Social data: Biases, methodological pitfalls, and ethical boundaries. *Frontiers in Big Data*, 2:13, 2019.

[58] Mauno Rönkkö, Jani Heikkinen, Ville Kotovirta, and Venkatachalam Chandrasekar. Automated preprocessing of environmental data. *Future Generation Computer Systems*, 45:13–24, 2015.

[59] Nripsuta Ani Saxena, Karen Huang, Evan DeFilippis, Goran Radanovic, David C Parkes, and Yang Liu. How do fairness definitions fare? examining public attitudes towards algorithmic definitions of fairness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 99–106, 2019.

[60] Clare Sloggett, Nuwan Goonasekera, and Enis Afgan. BioBlend: automating pipeline analyses within Galaxy and CloudMan. *Bioinformatics*, 29(13):1685–1686, 04 2013.

[61] Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. A unified approach to quantifying algorithmic unfairness: Measuring individual &group unfairness via inequality indices. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2239–2248, 2018.

[62] Harini Suresh and John V. Guttag. A framework for understanding unintended consequences of machine learning. *CoRR*, abs/1901.10002, 2019.

[63] Ivor W Tsang, James T Kwok, Pak-Ming Cheung, and Nello Cristianini. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(4), 2005.

[64] Rattapoom Tuchinda, Craig A. Knoblock, and Pedro Szekely. Building mashups by demonstration. *ACM Trans. Web*, 5(3), July 2011.

[65] Sahil Verma and Julia Rubin. Fairness definitions explained. In *2018 ieee/acm international workshop on software fairness (fairware)*, pages 1–7. IEEE, 2018.

[66] Samuel Yeom and Michael Carl Tschantz. Avoiding disparity amplification under different worldviews. *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, Mar 2021.

[67] Jiaming Zeng, Berk Ustun, and Cynthia Rudin. Interpretable classification models for recidivism prediction. *arXiv preprint arXiv:1503.07810*, 2015.