UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND IMPLEMENTATION OF A BIAS ANALYSIS TOOL FOR MACHINE LEARNING ALGORITHMS

MARTÍN GONZÁLEZ CALVO
JULIO 2020

TRABAJO DE FIN DE GRADO

Título:	Diseño e implementación de una herramineta de análisis del sesgo en algoritmos de aprendizaje automático
Título (inglés):	Design and implementation of a bias analysis tool for machine learning algorithms
Autor:	Martín González Calvo
Tutor:	Carlos A. Iglesias
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DESIGN AND IMPLEMENTATION OF A BIAS ANALYSIS TOOL FOR MACHINE LEARNING ALGORITHMS

Martín González Calvo

Julio 2020

Resumen

En los últimos años el uso de algoritmos de aprendizaje automático ha ido creciendo en numerosos campos, siendo una de sus aplicaciones clave la creación de sistemas que toman decisiones de manera autónoma. El gran problema de estas herramientas es que las decisiones que toman pueden tener un gran impacto en difrentes grupos de la población, dado que estos sistemas autónomos basados en aprendizaje automático son utilizados en numerosos campos como la justicia criminal, la educación, la banca, las aseguradoras, la sanidad pública o servicios sociales entre otros.

El dilema sobre si las conclusiones a la que estos algoritmos llegan son realmente justas es una preocupación cada día mayor entre los expertos en la materia, puesto que estas herramientas pueden estar sesgadas y ser injustas. Diferentes definiciones y métricas tanto para justicia como sesgo algorítmico se han propuesto. Sin embargo, a dia de hoy no existe un consenso claro, lo que dificulta la evaluación de estos sistemas.

El propósito principal de este trabajo de fin de grado es la creación de una herramienta que agrupe diferentes métricas y métodos de evaluación para valorar cómo de justos y/o sesgados son los algoritmos de aprendizaje automático auditados de una forma simple y entendible.

Para lograr este objetivo la herramienta hace uso de cuatro librerías (Aequitas, LIME, FairML y ThemisML), las cuales combinadas, proporcionan al usuario un entendimiento preciso de cómo los datos auditados se comportan. De este modo, se puede establecer si realmente el modelo debería funcionar así y ayudar al usuario a determinar si verdaderamente el comportamiento del algoritmo es injusto o sesgado.

La herramienta puede ser utilizada de dos formas diferentes: La primera consiste en un Jupyter Notebook donde el usuario puede modificar todos los valores que desee, pensada para gente con conocimientos técnicos. La segunda, es a través de una aplicación web con una interfaz de usuario intuitiva y fácil de usar donde todo es explicado detalladamente, dirigida a cualquier persona independientemente de sus conocimientos técnicos.

Palabras clave: Aprendizaje Automático, Sesgo, Justicia

Abstract

During recent years, the use of Machine Learning algorithms has grown constantly for multiple purposes. One of its key applications has been the use of automated decision models. The problem is those decisions can have a huge impact on different individuals as the decision-making algorithms are used in a wide range of areas, going from criminal justice, through education, banking, public health or social services.

The dilemma of whether the conclusions that these algorithms make are actually fair or not is a growing concern in this field of engineering, since, due to numerous reasons they can be biased or unfair. Multiple definitions and metrics for both fairness and bias have been proposed, although consensus hasn't been reached yet, so evaluation of these systems is a complicated matter.

The main purpose of this work is to develop a tool to group different metrics and evaluation methods to assess the fairness of our machine learning algorithms in a simple and understanding way.

In order to achieve this, the tool serves from four libraries (Aequitas, LIME, FairML and ThemisML), which combined provide the user with an accurate understanding of how the audited algorithm or data works and whether or not it should work that way, thus helping to determinate if the way the program works is actually unfair.

This tool can be consumed in two different ways. The first one, a Jupyter Notebook where the user can change and modify values, which is oriented for people with technical skills, and secondly, a web platform with a friendly user interface where everything is explained accurately and oriented towards any group of population that wants to use the tool.

Keywords: Machine Learning, Bias, Fairness

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Context

Nowadays, the use of Machine Learning (ML) [10] algorithms has almost become a part of our daily lives. From doing a quick search on the Internet, to getting a credit score by a banking company. From getting scored by an insurance company as high risk, through helping a judge to decide if an inmate deserves parole. In more cases than people actually think, these decisions are powered by ML, and its use is growing constantly for even more purposes [11].

The ever growing use of automated decision models powered by ML algorithms has led to the discovery that sometimes, this decisions aren't quite as fair as they are expected to be [12].

In many cases, this is due to the presence of bias in the decision-making process of the algorithms [2], hence, the need to detect the possible presence of it, quantify it, and tackle it in any way possible. Therefore, the aim of this project will be the development of a system than can perform, at least, some of this tasks.

1.2 Project goals

The main purpose of this project consists on the design and development of a system that groups different metrics to detect bias in ML algorithms.

The development of this software will carried out in the programming language Python, alongside the help of different libraries designed by academics and independent users, which either have the purpose of studying the behaviour of the algorithm or to audit the structure of a dataset that includes the final results.

Afterwards, all these metrics will be grouped in a unique space (something similar to a report) where the presence or lack of bias will be specified, indicating which variables are affected by its presence.

1.3 Structure of this document

In this section a brief overview of the chapters included in this document is provided. The structure is as follows:

- ***Chapter 1: Introduction***: It is the presentation of the project. Its context and the main goals are described.
- ***Chapter 2: Enabling Technologies***: Describes the different technologies used to carry out this project.
- ***Chapter 3: The Problem of Bias in Machine Learning***: Discusses the origin of bias in ML, stating some real cases and describing some tools to detect it.
- ***Chapter 4: Architecture***: Provides a description of the system as a whole.
- ***Chapter 5: Case Study***: Shows a demonstration of the tool working.
- ***Chapter 6: Conclusions***: The conclusions, achieved goals and future works are discussed.

CHAPTER 2

Enabling Technologies

In this chapter, the technologies used during this project will be discussed.

2.1 Machine Learning

Machine Learning (ML) is one of the many applications of Artificial Intelligence (AI) [13], that serves from large amounts of data acquired from observations and interactions with the world to build patterns that are translated into algorithms. That way, predictions about the future can be done in an autonomous way. In other words, as said by Tom M. Mitchell [10] "How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?".

As seen on figure 2.1, we find three differentiated techniques in ML [1]:

- ***Supervised Learning***: The most common of all, it can be understood as guided learning. Normally, a dataset is used as a instructor that trains the model and serves as a guidance source, that way, when new data is given, it can make predictions/decisions based on the previous data that has been given [1].

- ***Unsupervised Learning***: It is the opposite of supervised learning. The model learns by itself, analysing the data and finding structures in it. Later, when a dataset is given, it finds the patterns in it and groups the data according to those patterns. However, it cannot label those clusters created, e.g., it can't understand that something is an apple, and some other thing is an orange, but it will be able to separate apples from oranges in different clusters [1].

- ***Reinforcement Learning***: This discipline produces predictive models following a hit and trial method. The model is given feedback about whether a prediction is right or wrong, and with this feedback the model trains itself. Consequently, at the beginning the model will fail in a very big percentage of the predictions that it makes [1].

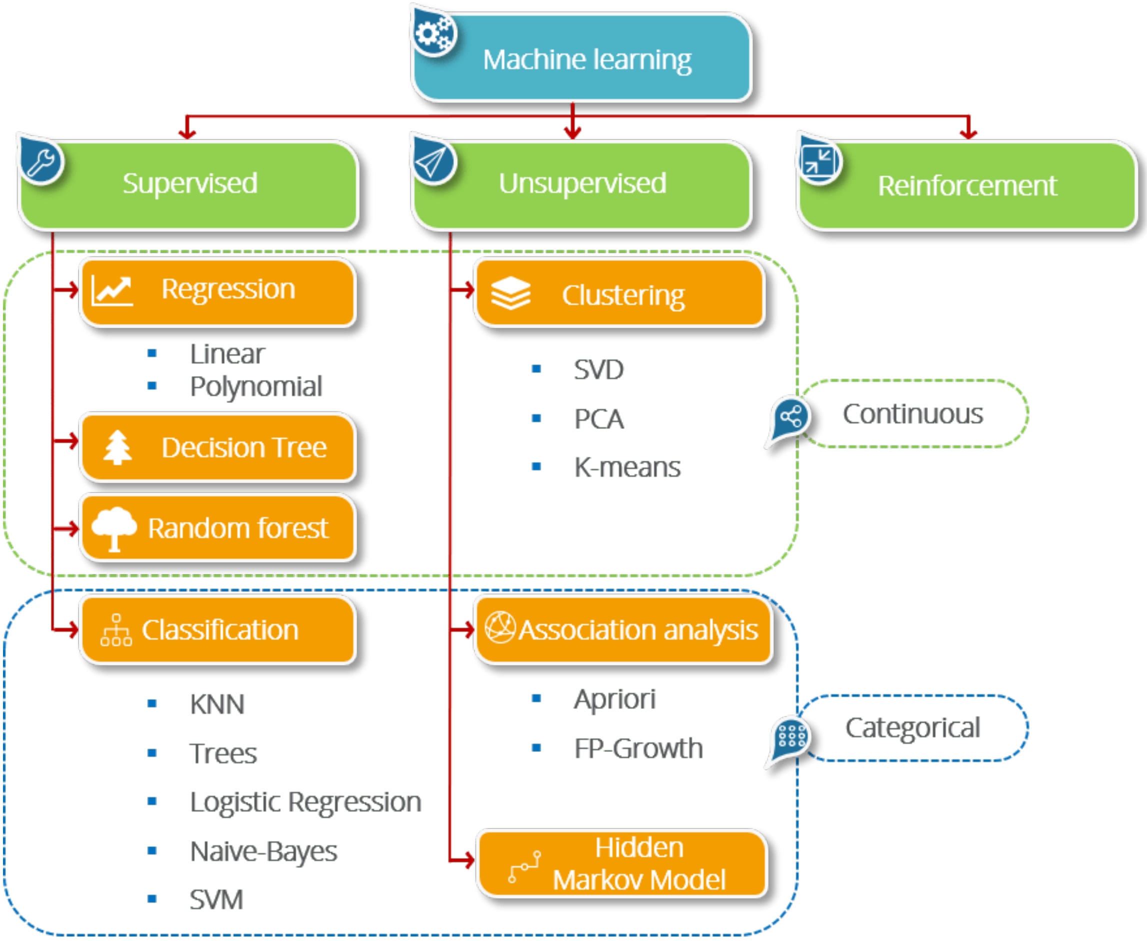


Figure 2.1: Machine Learning Types Tree [1].

While there are many technologies to work with ML (C++, R, Scala and so on) Python has been the programming language selected to carry out this project.

2.2 NumPy

NumPy [14] is the fundamental package for scientific computing with Python. Among all the features it provides, we can highlight [15]:

- ndarray, an efficient and fast multidimensional array object.
- Diverse functions that are capable of performing computations with arrays or mathematical operations between arrays element by element.
- Multiple tools that enable the user to read and write datasets based on arrays to disk.
- Support for linear algebra operations, Fourier transform and random number generation.
- Integration of C, C++, and Fortran code to Python.

Aside form its array-processing capabilities, one of its main purposes is as a container of generic data and the possibility to define any data type the user desires.

2.3 Pandas

Pandas [15] is an open source Python library built for the analysis and modeling of data. It provides rich data structures and functions designed to make working with structured data, the two main data structures are:

- ***Pandas Series***: It consists on a one-dimensional object, much like a one-dimensional array, where each element of the object has its own index.
- ***Pandas DataFrame***: The most common data structure used in Pandas, it consists on a two-dimensional object that can be understood as a database table or a spreadsheet, where each element is indexed.

Pandas works combining the features that Numpy provides along with other data manipulation features from spreadsheets and relational databases. It lets the user to import data from multiple formats like SQL, JSON, CSV and even text files to a pandas data structure, that way, the data can be reshaped, sliced, diced, aggregated and have subsets of it selected in a fast, easy and expressive way.

2.4 Scikit-Learn

Scikit-learn [16] consists on a free software ML library that has the aim to expose a wide range of ML algorithms, both supervised and unsupervised. This is achieved thanks to the use of a consistent, task-oriented interface, which enables the user to easily compare the different methods for a given application. Since scikit-learn counts on the scientific python ecosystem to function, it is a very easy task to implement it later into applications outside of the conventional statistical data analysis spectrum. It is also important to point out that algorithms written in a high-level language can be used as pieces of a larger process that tackles an specific use case, for example, medical imaging. Among the most important tools, scikit-learn includes:

- ***Clustering***: To group unlabeled data (see Unsupervised Learning).
- ***Cross Validation***: To estimate the accuracy of supervised models on unseen models.
- ***Dimensionality Reduction***: To eliminate the number of random attributes in the data.
- ***Preprocessing***: Many utilities to change raw feature vectors into a representation that is more suitable for the downstream estimators. Among the most important ones we find:
  - *Feature Extraction*: As its name indicates, extracts features from objects such as images or text.
  - *Feature Selection*: To identify which attributes are meaningful to create models later.
  - *Standarization & Normalization*: To make data distribution similar to a Gaussian with zero mean and unit variance, since it's a common requirement for many estimators in ML.
- ***Model selection and evaluation***: To get the best possible accuracy between all the models and parameters available.

For this project, scikit-learn has been used to learn how the process of making a ML model works. Also, to create mock models to test the bias auditing tool. However, it does not play an important role on the mechanisms of the auditor itself.

2.5 Matplotlib

Matplotlib [17] is a plotting library for Python designed in its beginnings to imitate MATLAB behaviour. It is built with an object-oriented Application Programming Interface (API) that lets the plots to be embedded. Between all its plotting functionalities the most important ones are: line plots, histograms, scatter plots, polar plots, contour plots, image plots and 3D plots.

2.6 Anaconda

Anaconda [18] is framework designed for Linux, Windows and macOS operative systems that enables the user to quickly download over 7,500 data science libraries. It also enables the creation of virtual environments to quickly operate with all those packages. For this project, its use has been essential since it contains by default almost all the libraries needed, and lets the user operate with them in a safe virtual environment so the operative system itself is safe.

2.7 Jupyter Notebooks

Jupyter Notebook documents [19] consist on files which contain both computer code such as Python and rich text elements like paragraphs, equations, figures and links. Notebook documents are both human-readable documents containing the analysis, description and results presented with multiple components like figures and tables as well as executable documents which can be run to perform data analysis.

These are run in the Jupyter Notebook App, which is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access or can be installed on a remote server and accessed through the Internet.

2.8 Flask

Flask [20] consists on a web framework written in the programming language Python. It is known for its simplicity and the lack of containing extra components that provide the user with common functions. Thanks to its simple way of working, extensions can be added and integrated as they are part of the framework.

Flask communicates with the Python kernel to retrieve data and make operations, and later passes on this data to a template engine to create the HTML files so a browser on the client side can render them (Figure 2.2). In this project, the template engine used is Jinja.

2.9 Jinja

Jinja [21] consists on a template engine for the python programming language developed by the creators of Flask. Since it is text based it is valid for source code and for markup languages like HTML. It is also the default template engine used in Flask.

Jinja main task consists on receiving the data that Flask provides, retrieving an already created HTML template, and fill it with the necessary information so it can be sent to the final user.

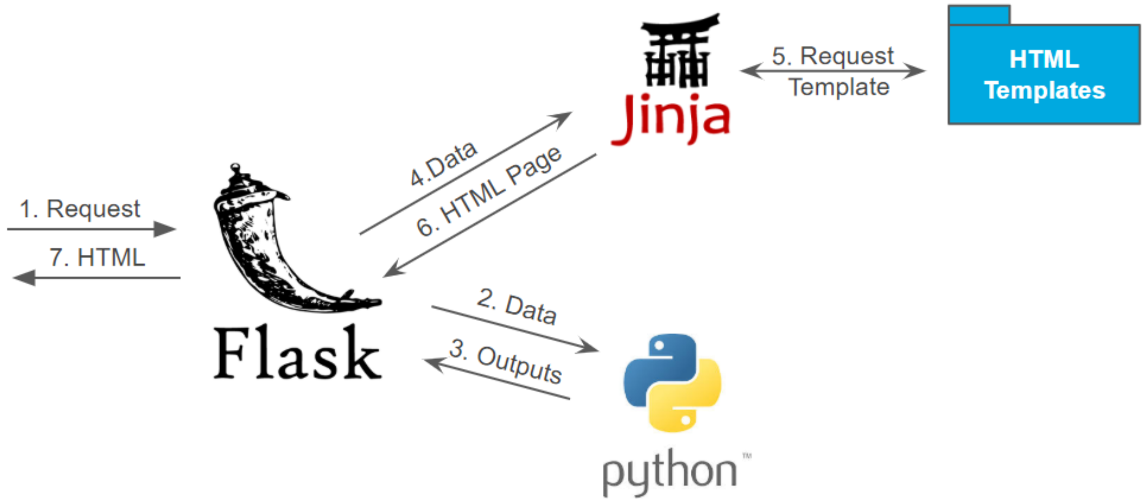


Figure 2.2: Flask & Jinja working scheme.

CHAPTER 3

The Problem of Biased Machine Learning Algorithms

3.1 Introduction

During recent years, the use of Machine Learning (ML) algorithms has grown constantly for multiple purposes. One of its key applications has been the use of automated decision models. The problem is, those decisions can have a huge impact on different individuals as the decision-making algorithms are used in a wide range of areas, going from criminal justice, through education, banking, public health or social services [11].

Several studies, like the one developed by Mehrabi et al. [12], have pointed out that unintended bias is real and affects people in several unfair ways, most of them belonging to specific vulnerable groups. However, we encounter one big question that still remains unanswered: What can be defined as fair or unbiased?

Multiple definitions and metrics for fairness and bias have been proposed [2], although consensus hasn't been reached yet, so evaluation of these systems is a complicated matter. In this chapter, what bias is, its problems, and discussion of some tools and techniques to tackle this issue will be discussed.

3.2 Definition & Origin of Bias in Machine Learning

When taking a look at the dictionary we get a quite precise definition of the meaning of the word bias: “the action of supporting or opposing a particular person or thing in an unfair way, because of allowing personal opinions to influence your judgment” [22].

Extrapolating this definition to a more accurate one in the ML field, bias is translated as predictions or results being skewed towards a certain part of the population, which results in an unfair outcome. We encounter two main reasons for bias to occur in ML [2]:

- ***Dataset Bias***: Related to the data and how it is treated (Figure 3.1). It can be divided in three differentiated subsets:
  - *Sample Bias (also known as Representation Bias)*: The outcome ends up being unfair because the acquired data is not representative of the population, e.g., too many samples of the type “A” whereas not being enough of the type “B”.
  - *Measurement Bias*: Values are distorted because the measurement process is faulty (tools or methods used to acquire the data), leaving important features behind or adding noise to the data.
  - *Prejudice Bias (also known as Historical Bias)*: The data itself is biased, e.g., if an algorithm is trained with multiple images showing men writing code and women in the kitchen, is likely that It will learn that coders are men and homemakers are women. “It is a normative concern with the state of the world, and exists even given perfect sampling and feature selection” [2].
- ***Algorithmic Bias***: When the outcome ends up being unfair because of how the model processes the data, as seen on Figure 3.2, according to Suresh et al. [2] we can break it into three different categories:
  - *Aggregation Bias*: Occurs when the model is built. The reason for it to happen is that the different populations are not combined in a proper way. In most cases, the population is not homogeneous and to fit all subgroups in a unique model leads to errors [2].
  - *Evaluation Bias*: Arises when the model is evaluated. Usually because the performance benchmarks used are not suitable for the different parts of the population, since they are not representative of all of it [2].
  - *Deployment Bias*: Takes place once the model has been deployed, normally because the system in inappropriately interpreted [2].

Nevertheless, bias in ML maintains a close relationship with what we understand as statistical bias. This concept can be understood as the overestimation or underestimation of a population parameter and happens because of many different reasons, some stated above.

To sum up, statistical bias explains the existence of bias in the datasets used to train the algorithms, but it is not the only reason why this phenomenon exists in ML, since the algorithms themselves can also be biased as stated before.

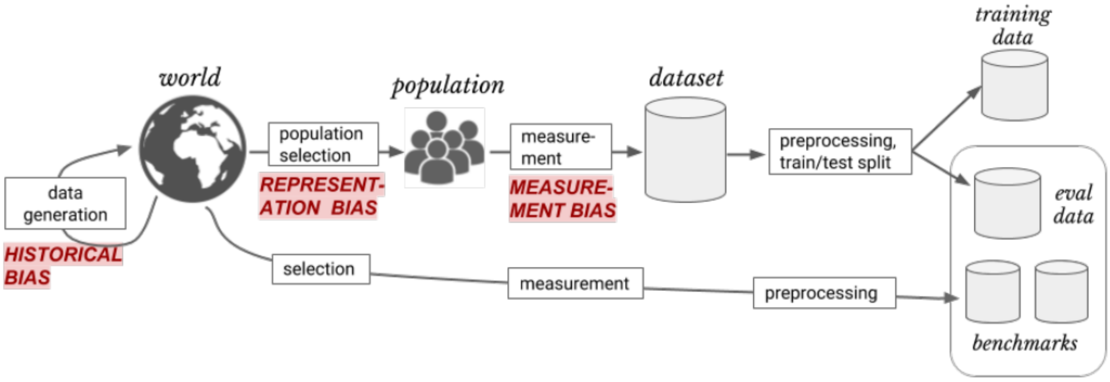


Figure 3.1: Process of Data Generation [2].

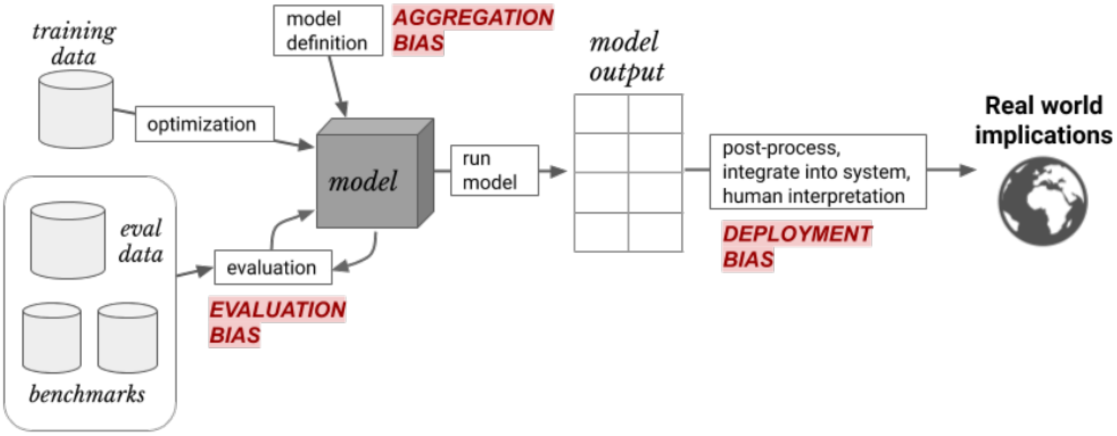


Figure 3.2: Model Building & Implementation [2].

3.3 Real Cases of Biased Machine Learning Algorithms

Amazon's Automated Resumes Tool

According to different media [23], for almost a year, Amazon tried to implement a tool to automatically select resumes between hundreds of applicants. After some time had passed, developers realized that the tool refused to accept applicants that were female. The reason for this was that the algorithm had been trained with a dataset made up by the company last 10 year resumes, which were mostly male applicants. This caused the algorithm to self-learn that male candidates were preferable over female ones, an obvious case of gender bias.

Microsoft's Chat Bot Tay

On March 2016, Microsoft released on Twitter a chat bot called Tay [24]. Only 16 hours after it was released, Microsoft turned off the service due to the bot publishing controversial tweets (denying the holocaust or imitating Donald Trump talking about the infamous wall). It is thought that these tweets were the result of hundreds of trolls talking to the bot and introducing what we know as Prejudice Bias.

Face Recognition on different platforms

MIT researcher Joy Buolamwini, released an study [3] where it was found out that face recognition software sold by Microsoft, IBM and Face++, had an average 0.8% error rate for white males, whereas for black women reached 34.7% in the worst case (Figure 3.3). Again, this was related to the training dataset being populated by white male faces.

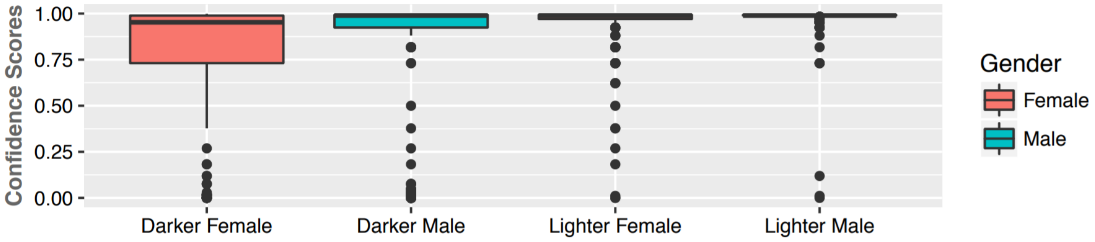


Figure 3.3: Gender classification confidence scores from IBM. Scores are near 1 for lighter male and female subjects while they range from $\sim 0.75 - 1$ for darker females [3].

COMPAS

In the United States of America, numerous algorithms to asses whether a defendant will re-offend are being increasingly used. These algorithms are created in many different ways, going from the different states developing them individually, to researchers and academics across the country collaborating. However, in most cases proprietary software is used, being Northpointe's Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) [25] the most widespread alternative.

ProPublica, an independent, non-profit newsroom that dedicates to investigative journalism did an extensive study [4], on the effectiveness and possible bias of COMPAS. The study highlights many important facts, and between them we can stick out [4]:

- Black defendants who did not re-offend during a time interval of two-years were almost twice as likely to be misjudged as higher risk compared to white defendants (Figure 3.4).
- Figure 3.5 demonstrates that white defendants who did re-offend during a time interval of two-years were misjudged and labeled as low risk almost twice as often as black defendants that actually re-offended (48 percent vs. 28 percent).
- Black defendants were also two times higher than white defendants to be misjudged as a higher risk for a violent re-offense (Figure 3.6). Furthermore, violent re-offenders who were white were 63 percent more likely to be misjudged as a low risk for violent re-offending, compared with violent re-offenders who were black.

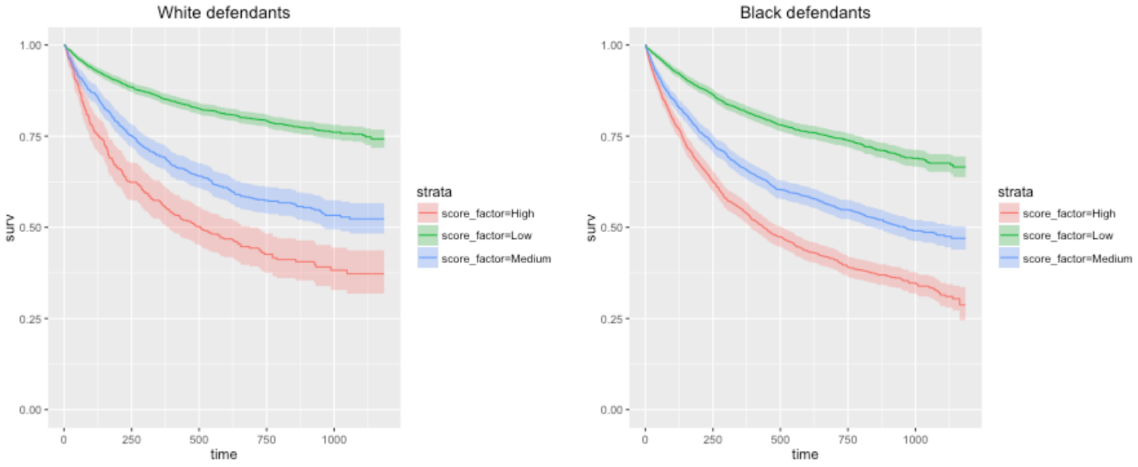


Figure 3.4: Risk of General Recidivism Cox Model (with race-by-score Interaction Term) [4].

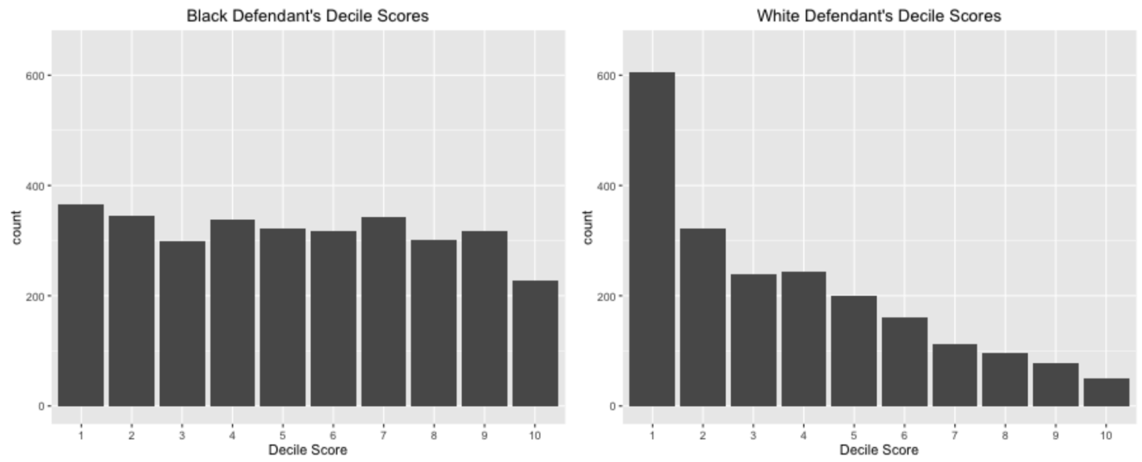


Figure 3.5: Black vs. White Defendants Recidivism Risk Scores [4].

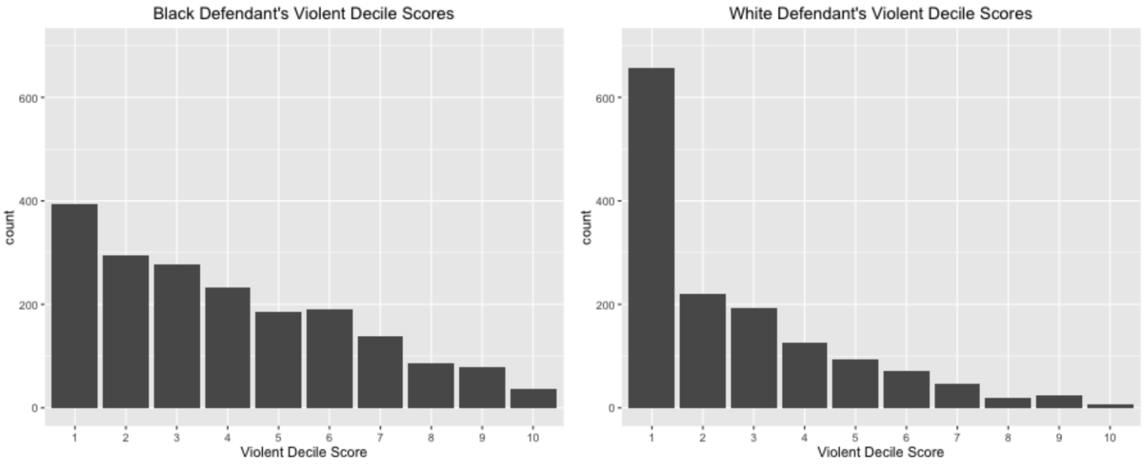


Figure 3.6: Black vs. White Defendants Violent Recidivism Scores [4].

Multiple studies regarding this topic have been developed, for example, the one conducted by Jennifer L. Skeem [26] which ended up determining that the differences could not be attributable to bias. However, other academics like Mark Olver et al. [27] propose that "One possibility may be that systematic bias within the justice system may distort the measurement of 'true' recidivism" [27]. Again we are in front of a case of what we understand as Historical Bias within the dataset.

Allegheny Family Screening Tool

The Allegheny Family Screening Tool [28] is a ML model developed to assist social workers from the Allegheny County's child welfare office. Its main purpose consists on helping the workers to decide whether children could be a victim of abusive circumstances, thus needing to be removed from their families. The tool has been designed in an open and transparent way, with public forums and opportunities to find flaws and inequities within the software.

This tool is a great example of how to mitigate bias and avoid a negative feedback loop in a ML algorithm, since the score is not provided to caseworkers nor to judicial parties should the case continue to move through the system. That way, it is ensured that judicial decisions are not influenced by the score. This would be a huge problem, because those biased algorithm decisions that are later ratified by a jury could go back and be fed back into the tool directly as a means for future assessment.

Other concerns about the tool, like its possible racial disparity were taken into consideration and evaluated, and, as stated in the Frequently-Asked Questions document regarding the tool "The County made the decision not to include race as a factor [...] However, there are other predictors that are correlated with race due to potentially institutionalized racial bias (e.g., criminal justice history) that would imply that race is still a factor." [29]. We are, again, in front of a possible case of Historical Bias, the hardest to detect and mitigate.

Beauty.AI

In the year 2016 Youth Laboratories released the first ever beauty contest evaluated by robots, called Beauty.AI [30]. The main idea behind this invention was to eventually end all beauty contests (What would be as precise as a machine determining something, even beauty?). However, out of the 44 winners, most of them turned out to be white. This began to arise suspicions about the possibility of the algorithm being racist, leading to Beauty.AI to be taken offline. This is thought to happen because the model wasn't fed with enough diverse data. Probably, most of the images from the dataset were white people.

A latter study by Böhlem et al. [30] on building a model that punctuates beauty, was carried out thanks to the University of Hong Kong's large-scale CelebFaces Attributes Dataset, known as CelebA [31]. The study ended up determining that, the attribute 'Attractive' did not maintain a relationship with just a single feature. Thus concluding that the concept of attractiveness is in fact multidimensional.

3.4 Tools & Techniques to tackle bias in Machine Learning

3.4.1 Aequitas

Aequitas [11] is an audit tool developed by the Center for Data Science and Public Policy of The University of Chicago, and, as their web page states, it consists on an open source bias audit toolkit built to be used by ML developers, data scientists, analysts of all kind, and policymakers to audit ML models to find possible discrimination and bias on said models. Thus helping to make informed and equitable decisions around developing and deploying ML models involved in predictive risk-assessment tools.

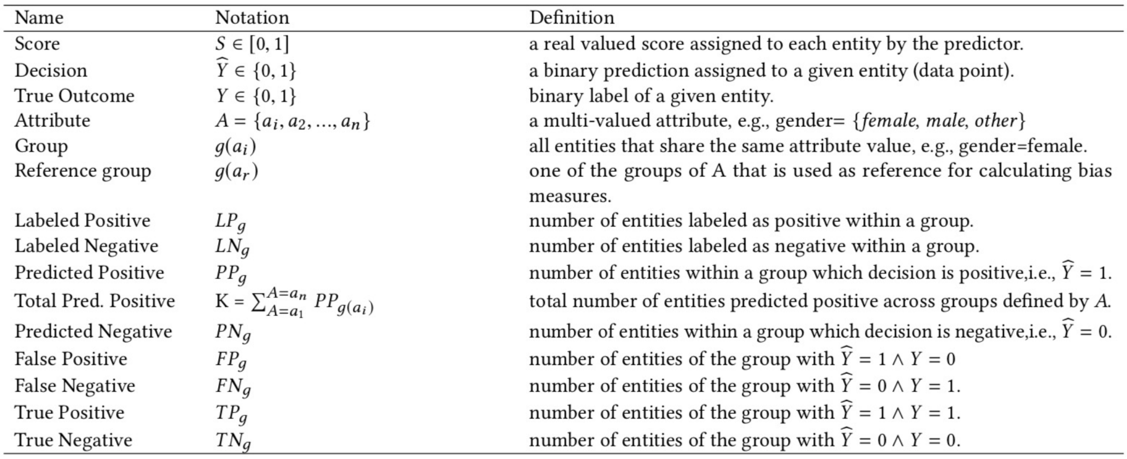
Aequitas bases its functionality in being fed a dataset with three differentiated parts:

- ***score***: This column of the dataset represents the conclusion that a model reaches, it can be binary (0 or 1) or a decimal between 0 and 1, which would require indicating a threshold to denote a binary decision. This decision represents if the subject is apt or not, e.g., being granted parole or being granted a loan.
- ***label_value***: This column of the dataset represents the ground truth data, in other words, if the prediction made by the model was right, e.g., The subject did not re-offend after being granted parole or the subject was able to repay the loan. That is why the model can only be audited after it has been implemented and not before. It is also a binary value, 0 meaning the prediction was incorrect, 1 meaning the prediction was correct.
- ***attributes***: These elements consist on as many columns as the user chooses to have. They are the key element to be audited and are used to decide the fairness of the model. Examples of attributes include race, gender or age.

score	label_value	race	sex	age_cat
0	1	African-American	Male	25 - 45
1	1	Native-American	Female	Less than 25

Table 3.1: Example dataset to be fed to the Aequitas Tool

To understand how the tool works we need to go over some preliminary concepts:

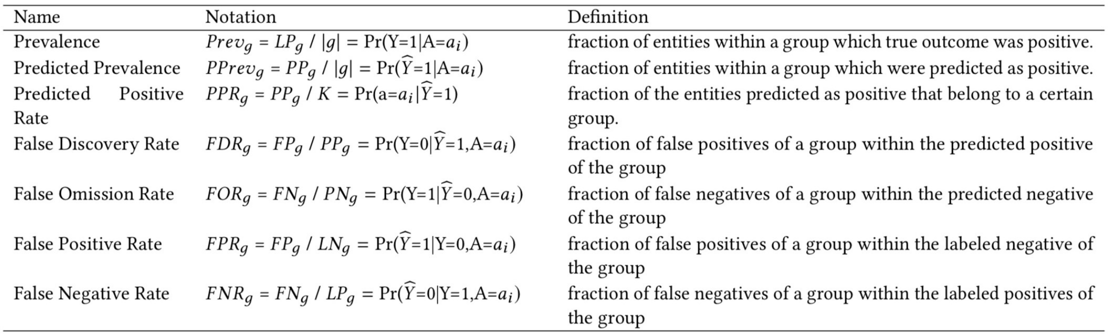


Name	Notation	Definition
Score	$S \in [0,1]$	a real valued score assigned to each entity by the predictor.
Decision	$\widehat{Y} \in \{0,1\}$	a binary prediction assigned to a given entity (data point).
True Outcome	$Y \in \{0,1\}$	binary label of a given entity.
Attribute	$A = \{a_i, a_2, ..., a_n\}$	a multi-valued attribute, e.g., gender= $\{female, male, other\}$
Group	$g(a_i)$	all entities that share the same attribute value, e.g., gender=female.
Reference group	$g(a_r)$	one of the groups of A that is used as reference for calculating bias measures.
Labeled Positive	LP_g	number of entities labeled as positive within a group.
Labeled Negative	LN_g	number of entities labeled as negative within a group.
Predicted Positive	PP_g	number of entities within a group which decision is positive,i.e., $\widehat{Y} = 1$.
Total Pred. Positive	$\mathrm{K} = \sum_{A=a_1}^{A=a_n} PP_{g(a_i)}$	total number of entities predicted positive across groups defined by A.
Predicted Negative	PN_g	number of entities within a group which decision is negative,i.e., $\widehat{Y} = 0$.
False Positive	FP_g	number of entities of the group with $\widehat{Y} = 1 \wedge Y = 0$
False Negative	FN_g	number of entities of the group with $\widehat{Y} = 0 \wedge Y = 1$.
True Positive	TP_g	number of entities of the group with $\widehat{Y} = 1 \wedge Y = 1$.
True Negative	TN_g	number of entities of the group with $\widehat{Y} = 0 \wedge Y = 0$.

Table 3.2: Aequitas Preliminary Concepts [5].

Aequitas produces three types of metrics:

- ***Group Metrics***:



Name	Notation	Definition
Prevalence	$Prev_g = LP_g \ / \ \lvert g \rvert = \Pr(\mathrm{Y}{=}1 \vert \mathrm{A}{=}a_i)$	fraction of entities within a group which true outcome was positive.
Predicted Prevalence	$PPrev_g = PP_g \ / \ \lvert g \rvert = \Pr(\widehat{\mathrm{Y}}{=}1 \vert \mathrm{A}{=}a_i)$	fraction of entities within a group which were predicted as positive.
Predicted Positive Rate	$PPR_g = PP_g \ / \ K = \Pr(\mathrm{a}{=}a_i \vert \widehat{\mathrm{Y}}{=}1)$	fraction of the entities predicted as positive that belong to a certain group.
False Discovery Rate	$FDR_g = FP_g \ / \ PP_g = \Pr(\mathrm{Y}{=}0 \vert \widehat{\mathrm{Y}}{=}1,\mathrm{A}{=}a_i)$	fraction of false positives of a group within the predicted positive of the group
False Omission Rate	$FOR_g = FN_g \ / \ PN_g = \Pr(\mathrm{Y}{=}1 \vert \widehat{\mathrm{Y}}{=}0,\mathrm{A}{=}a_i)$	fraction of false negatives of a group within the predicted negative of the group
False Positive Rate	$FPR_g = FP_g \ / \ LN_g = \Pr(\widehat{\mathrm{Y}}{=}1 \vert \mathrm{Y}{=}0,\mathrm{A}{=}a_i)$	fraction of false positives of a group within the labeled negative of the group
False Negative Rate	$FNR_g = FN_g \ / \ LP_g = \Pr(\widehat{\mathrm{Y}}{=}0 \vert \mathrm{Y}{=}1,\mathrm{A}{=}a_i)$	fraction of false negatives of a group within the labeled positives of the group

Table 3.3: Aequitas Group Metrics [5].

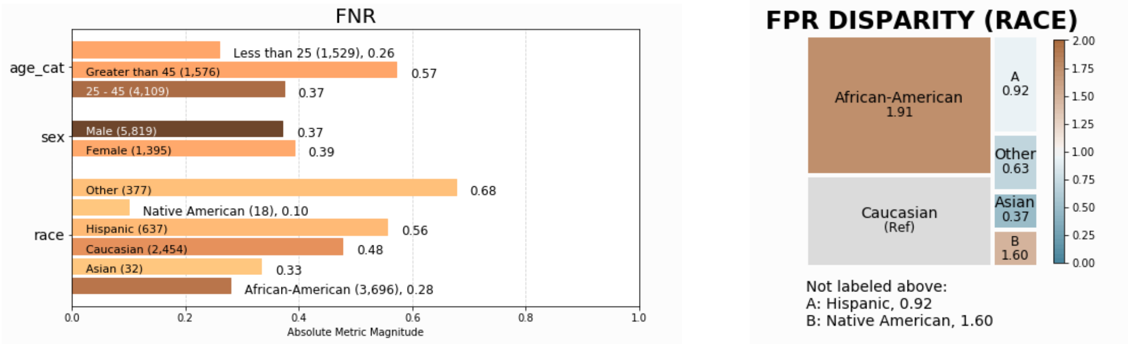
- ***Bias Metrics***: Measures the disparity between a group and the reference group. This disparity is calculated as seen on Equation 3.1.
- ***Fairness Metrics***: Defined in relation to a reference group. A group meets parity if the conditions stated on Equation 3.2 are met (By default $\tau = 0.2$).

$$MetricDisparity_{ProtectedGroup} = \frac{GroupMetric_{ProtectedGroup}}{GroupMetric_{ReferenceGroup}} \tag{3.1}$$

$$(1-\tau) \leq MetricDisparity_{ProtectedGroup} \leq \frac{1}{(1-\tau)} \tag{3.2}$$

Aequitas can be consumed in three different ways:

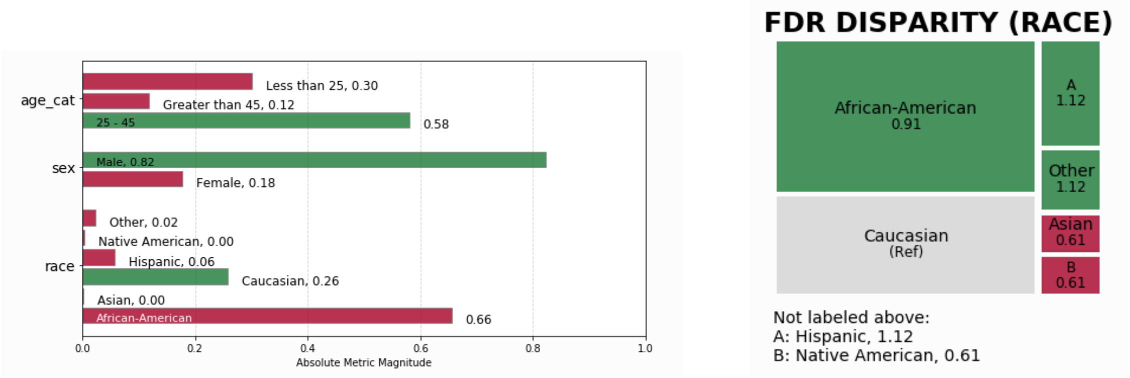
- ***Web App***: It can be ran either locally or through their own website. Once the dataset and some parameters are given to the tool, it delivers a report based on descriptive tables giving group, bias and fairness metrics, but with no graphics.
- ***CLI***: Command line tool mode, works in a similar way to the web app but without its user friendly graphic interface.
- ***Python Environment***: The most powerful out of the three, since aside from getting the calculation tables, offers graphics to understand better the three metrics (Examples include Figures 3.7 and 3.8). For the creation of the tool, this will be the modality used.



(a) Group Graph

(b) Bias Graph

Figure 3.7: Aequitas Bias & Group Graphics Example [5].



(a) Group Fairness Graph

(b) Bias Fairness Graph

Figure 3.8: Aequitas Bias & Group Fairness Graphics Example [5].

3.4.2 FairML

FairML [32] consists on a library that quantifies the relative significance of the model's inputs. This is achieved by slightly perturbing the inputs and seeing how the output changes. In order to do this, orthogonal projection is used as a perturbation scheme. An orthogonal projection consists on a vector projection that maps a vector onto a direction perpendicularly to the reference vector (Figure 3.9).

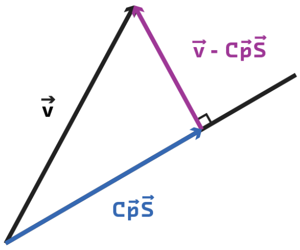


Figure 3.9: Example of an Orthogonal Projection $\vec{v}$ over $C\vec{p}\vec{S}$ (In Euclidean Space) [6].

Given the case that two vectors are orthogonal between each other, it is impossible that a linear transformation of one of them can give as a result the other vector and vice-versa. With this premise, FairML measures the feature dependence: The difference between outputs when the the input is perturbed and when is not shows the dependence of the model for a designated attribute, and because of orthogonality it is assured that there will not be hidden collinearity effects [6].

However, an orthogonal projection is a linear transformation, and it can exist non-linear dependence among attributes. Because of this FairML also uses model compression and other input ranking algorithms. Once the relative significance of the attributes is determined (Figure 3.10), the information is used to assess the fairness of a model.

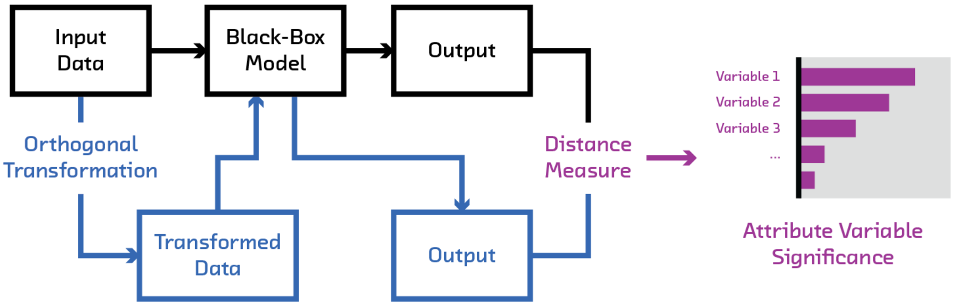


Figure 3.10: FairML Process Diagram [6].

3.4.3 Themis-ML

Themis-ML [33] consists on an open source library developed by Data Scientist Niels Bantilan that implements several fairness-aware methods that comply with the scikit-learn API. The library offers four modules:

- ***Datasets***: Module for loading pre-processed datasets.
- ***Metrics***: Module for Fairness-aware scoring metrics.
- ***Preprocessing***: Module for pre-processing linear models.
- ***Postprocessing***: Module with utilities for computing statistics and doing checks.

For the purpose of creating the tool the function `mean_difference` from the module Metrics will be used, in equation 3.3 the calculations that this function makes can be seen. Where y is an array $(n \times 1)$ containing the binary target variable for all the subjects (1 is the desirable outcome and 0 is the undesirable outcome), and s consists on an array of the same size $(n \times 1)$ indicating which of those subjects belongs to the disadvantaged group. The result will always be a decimal number between 0 and 1.

$$mean\_difference = \frac{1}{n}\sum_{i=0}^{n-1} y(s=0)_i - \frac{1}{m}\sum_{i=0}^{m-1} y(s=1)_i \tag{3.3}$$

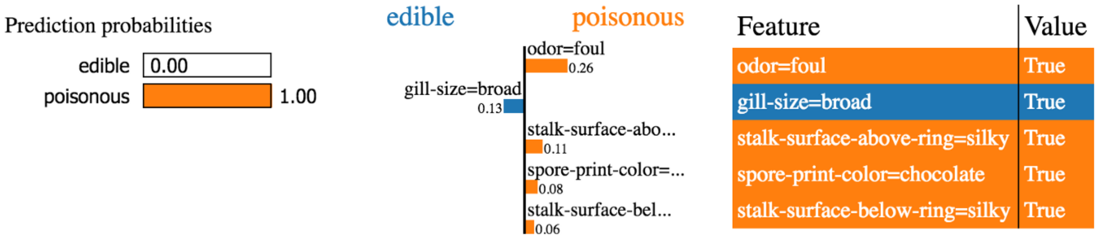
Algorithm 1: mean_difference(y, s) [Binary Target Case]

```
input : y: array containing binary target variable for all the groups, where 1 is
          the desirable outcome and 0 is the undesirable outcome (size n × 1)
input : s: array indicating which subjects belong to the protected group, where 1
          is belongs to the group and 0 is not (size n × 1)
output: Computed mean difference in y with respect to protected class s.
mean_diff = y[s == 0].mean() - y[s == 1].mean()
```

3.4.4 LIME

LIME [7] consists on a model-agnostic explainer, this means that the tool can be applied to any ML model. The biggest difference between the model-agnostic and model-specific approach to explain a model is, that instead of analysing the inner mechanisms of a model, it perturbs the input data and sees how the predictions change. That way, a human can understand how a model works and see if it is biased.

The output that LIME produces consists on a list of explanations that describe the importance that each feature has in order to make a prediction. An example can be seen on figure 3.11.



Feature	Value
odor=foul	True
gill-size=broad	True
stalk-surface-above-ring=silky	True
spore-print-color=chocolate	True
stalk-surface-below-ring=silky	True

Figure 3.11: Example of LIME applied to a classification problem [7].

The process of creating explanations so a human can later interpret them (Figure 3.12) begins with approximating the model to one that is interpretable and local. Examples of interpretable models include strongly regulated linear models or decision trees. Once the interpretable model is created, it is trained with an slightly changed dataset that gives a local approximation. This perturbed data is created with different techniques, like for example, adding noise to continuous features. Since the model has been approximated locally, the task gets to be simplified [9].

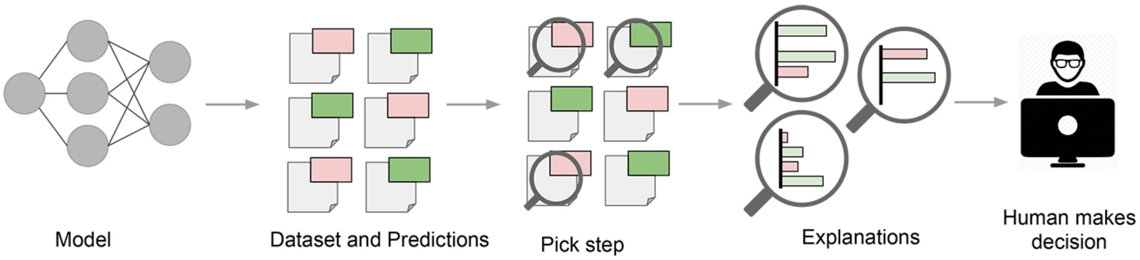


Figure 3.12: Explaining a model to a human decision-maker [8].

However, LIME has two main setbacks [9]:

- ***Linearity***: To obtain the local behaviour, only linear models are used. If only a small section of the data is taken this does not present a problem. Nevertheless, the model can contain some non-linear behaviours and LIME would fail to explain the behaviour of the original model. This can be understood in Figure 3.14.
- ***Data Perturbation***: In some cases, perturbing the data with simple modifications is not enough. The ideal case is that perturbations would be driven by the variation that is observed in the dataset. Also, modifying the values manually would not be a great idea, as it most likely would introduce bias into the model explanations.

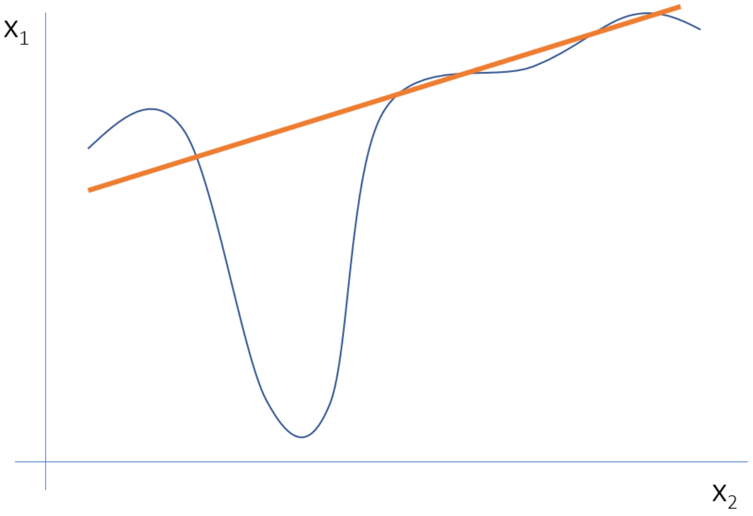


Figure 3.13: A linear approximation of the local behaviour for two features is not a good representation and won't capture the highly non-linear behaviour of the model [9].

CHAPTER 4

Architecture

In this chapter, the general structure of the project and its different phases will be explained. Afterwards, a more in-depth explanation of its internal functioning is provided.

4.1 Introduction

In this project, a tool to detect bias in ML models will be developed. This consists on a mechanism that given a ML model with input data already used or a dataset with the input data and the corresponding results, will provide the final user with insight about the behaviour of the audited model or data.

The results obtained after processing the data will be presented in a report, either explaining how the model works so the user can assess if the decision-making process is biased or with the results of the analyzed dataset confirming whether or not the results acquired are biased.

As a result of this, the overall fairness of an studied system can be obtained and decisions can be made on how to mitigate the possible presence of bias.

The Bias Auditing Tool will be able to be consumed in two different ways:

- ***Jupyter Notebook***: This method is directed towards people with technical skills, all the code can be changed and it offers more flexibility than the web application. In order to use this method is necessary to have an environment that can render Jupyter Notebooks. Also, the different requirements would need to be installed.
- ***Web Application***: Provides a friendly Graphic User Interface (GUI) to use the bias auditor. It is oriented towards any group of population that wants to use the tool. The only requirement to use this method is an up-to-date web browser.

In Figure 4.1 a block diagram of the overall process with the different stages needed to get the results is described. The preprocessing and processing stages are covered by the created classes as a single process, depending on the input data that the final user provides. To achieve this, four Python classes have been created, one for each library that is to be implemented. These classes have later been integrated in a single package called "Bias_Auditer". The libraries implemented are LIME, FairML, Themis-ML and Aequitas.

Once the audit results are generated, they will be rendered as a whole on a unique report-like space. If the tool is executed in a Jupyter Notebook extension (sect. 4.3) the auxiliary classes created for this purpose will render the report with the received results. If the web application (sect. 4.4) is being used, the backend will generate the HTML code with the audit results and present it to the user in the frontend.

The main advantage of using this tool instead of the different libraries separately is that it saves the user all the data munging procedures and the initial steps required to process the models. Furthermore, it offers all the metrics in a single space.

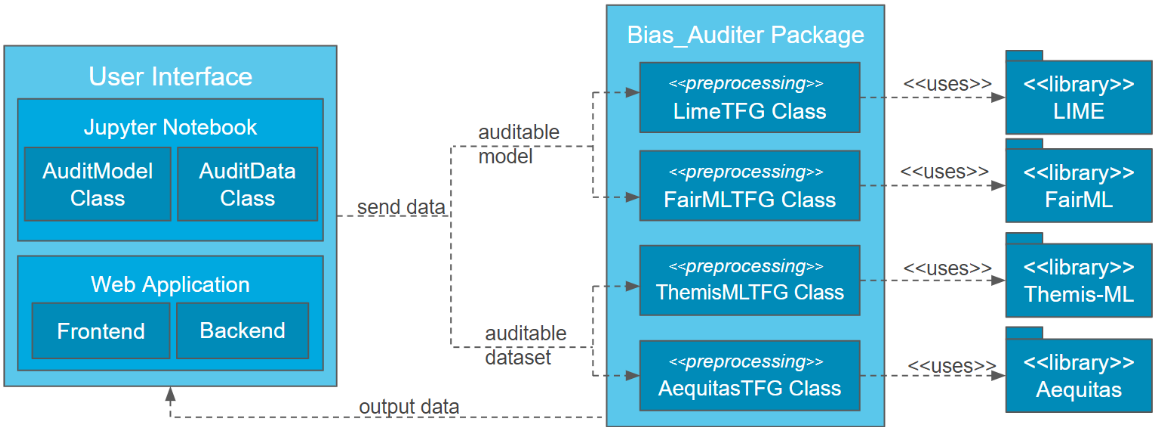


Figure 4.1: General process overview of the bias auditing tool

Input Data

In this step, the initial data is acquired. This inputs can consist on:

- ***Model***: The model to audit with the dataset used to train it.
- ***Dataset***: Containing the output of the model and the different features that want to be audited.

The model and the dataset can be provided, or just one of these elements. However, if both the model and the dataset are provided, the data must be one hot encoded (this means that all values including the output are 1 or 0). Nevertheless, if only the dataset is provided, it can be categorized or one hot encoded. Also some parameters related to the data must be given.

Preprocessing

In this step, depending on the input data, the adequate classes are triggered and the procedure starts. If a dataset is provided, several modifications are made to its structure, so it can properly work with the different libraries. In the case of also having models, the initial procedures to work with the libraries are done, so the outputs can be acquired by the user just by calling the necessary functions.

Processing

In this step, the preprocessed data is given to the different libraries to make diverse calculations and produce graphics and data about the model that help to determine whether the given model is biased following a series of guidelines.

Report

In this final step, the outputs from the different libraries are collected. After that, the user is provided with a report made up of different graphics and tables, so the outputs are represented as a whole alongside useful data to determine if there is presence of bias within the audited model or dataset.

4.2 Bias Auditer Python Package

The package contains four classes, one for each library implemented. It will be used in both the Jupyter Notebook and the web application with some minor modifications to be adequated to each environment.

The inner processes and functionalities of the different libraries have already been explained in section 3.4, so no further details about them will be provided in this section. In this section, a small overview of the classes will be given. On Appendix D a more in depth explanation for the constructor parameters and the respective methods of each class is given.

4.2.1 Class LimeTFG

This class has the purpose of implementing the library LIME [7]. With different calculations, LIME gets to explain how a model works and what features are of most importance. It works for many types of ML models. However, the main focus for this implementation is models that work with one hot encoded datasets.

LIME generates an explanation for each individual result that the model produces. This means that every explanation that is rendered will be different. On Figure 4.2 an Unified Modeling Langugage (UML) [34] diagram of the class is presented.

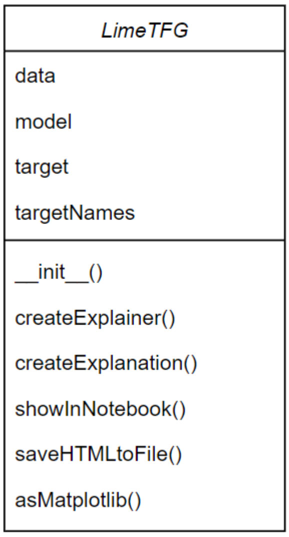


Figure 4.2: LimeTFG Class UML Diagram.

4.2.2 Class FairMLTFG

This class implements FairML [32], a library that gets the different feature dependencies of the audited model. The tool only works for models that process one hot encoded datasets. Unlike LIME, that focuses on explaining each individual prediction, FairML focuses on the model as a whole. On Figure 4.3 an UML diagram of the class is presented.

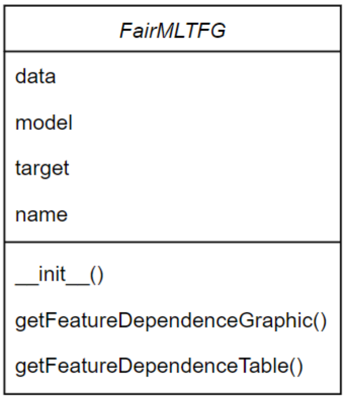


Figure 4.3: FairMLTFG Class UML Diagram.

4.2.3 Class ThemisMLTFG

The main purpose of this class is to implement a single function from the library Themis-ML [33]. This function calculates the mean difference in respect to a protected group (See section 3.4.3). On Figure 4.4 an UML diagram of the class is presented.

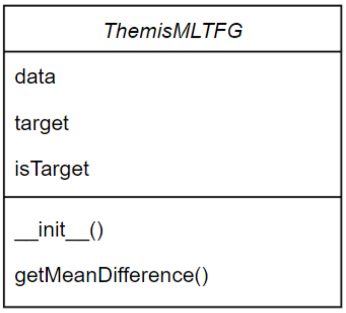


Figure 4.4: ThemisMLTFG Class UML Diagram.

4.2.4 Class AequitasTFG

Aequitas [11] is the most complete library out of the four implemented for this project. The inner working process has already been explained (See section 3.4.1). On Figure 4.5 an UML diagram of the class is presented.

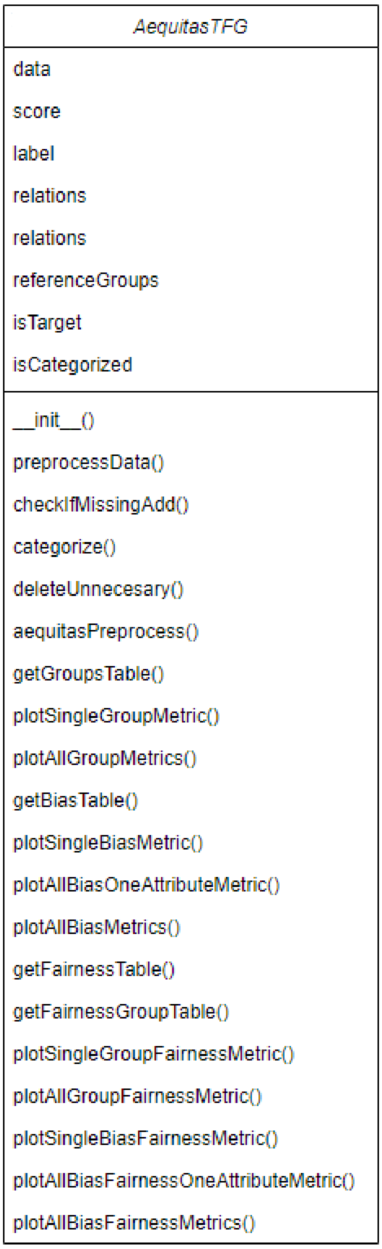


Figure 4.5: AequitasTFG Class UML Diagram.

4.3 Jupyter Notebook

One of the modalities that the tool will be able to be consumed is as a Jupyter Notebook extension. For this purpose, a Notebook file has been created where the code from the classes explained on the previous section can be found.

In addition, to group all the metrics two additional classes have been created. The first one enables the user to create an object to audit models, the second one is used to audit datasets with model results.

4.3.1 Class AuditModel

This class has the purpose of unifying the model explainers LIME and FairML so later the results can be shown as a whole in the Jupyter Notebook. For this purpose it makes use of the classes LimeTFG and FairMLTFG. On Figure 4.6 an UML diagram of the class is presented.

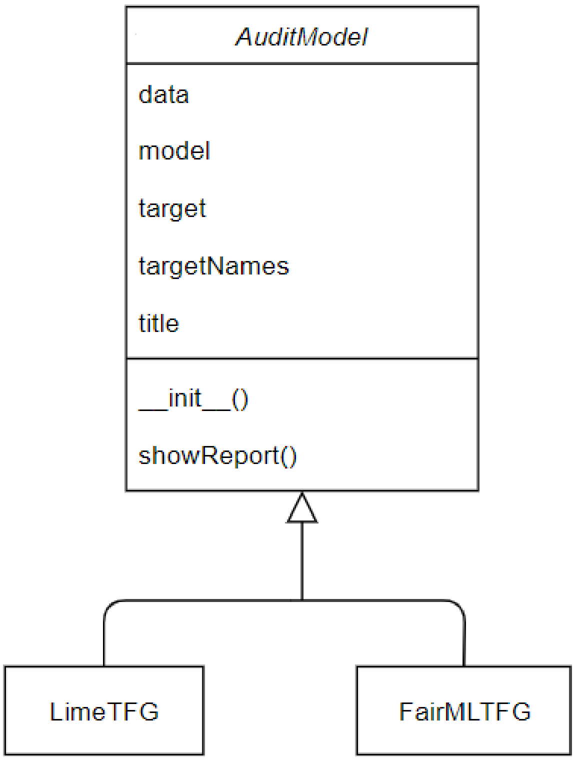


Figure 4.6: AuditModel Class UML Diagram.

4.3.2 Class AuditDataset

This class has to purpose of unifying the metrics provided by Themis-ML and Aequitas so later the results can be shown as a whole in the Jupyter Notebook. For this purpose it makes use of the classes ThemisMLTFG and AequitasTFG. On Figure 4.7 an UML diagram of the class is presented.

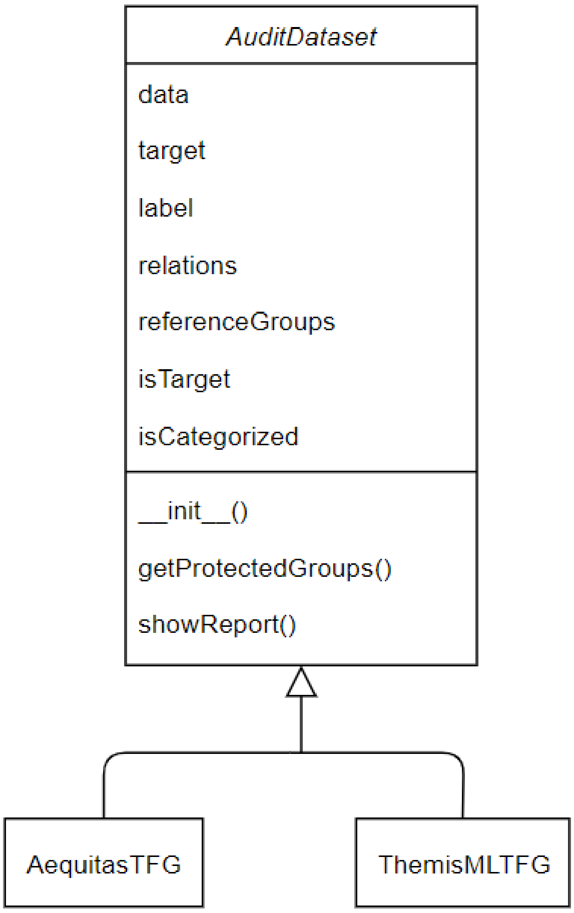


Figure 4.7: AuditDataset Class UML Diagram.

4.4 Web Application

The web application serves from the web framework flask and the already created code. There are two differentiated parts, the frontend, where the user interacts with the application, introducing the different inputs and getting the report as an output, and the backend, where the logic is found and the different operations are made.

In the first place, the final user accesses the web platform via a web browser. There, two options are presented, either audit a dataset, or a model. After selecting the desired option, the necessary input parameters are introduced (this is further explained on Chapter 5), and the report is produced. In order to achieve this, the process (Figure 4.8) uses the following technologies:

- ***Frontend***: HTML5, CSS3, JavaScript, Bootstrap and jQuery to render the views given by the backend.
- ***Backend***: Flask, Jinja2 and Python to produce the code with all the necessary information. Flask receives the input data by the user, and asks the Python kernel to make the operations needed, after receiving the requested data, it communicates with Jinja2 to introduce this data in the HTML templates. Finally, the user receives the code and the browser renders the page thanks to the frontend technologies.

There is no need for a persistance layer since any data from the user will be saved.

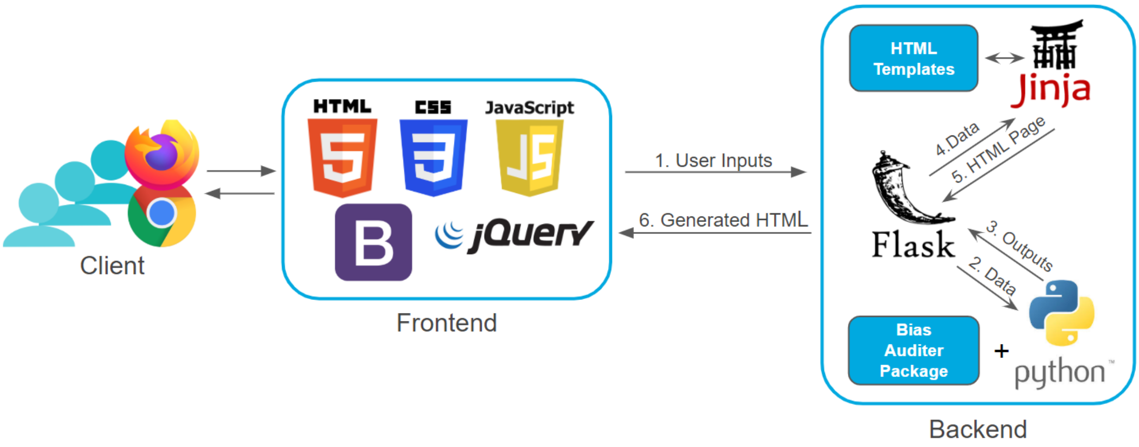


Figure 4.8: Web Application Scheme.

The different parameters are given as inputs via form with a friendly GUI [35]. On Figure 4.9 an UML diagram of the navigation process is described. The web application has three differentiated sections:

- ***Home***: This is the landing page, with a simple introduction to the tools, the libraries used, a contact section and links to audit either a dataset or a model.
- ***Audit Model***: This page serves as a GUI for the class AuditModel.
- ***Audit Dataset***: This page serves as a GUI for the class AuditDataset.

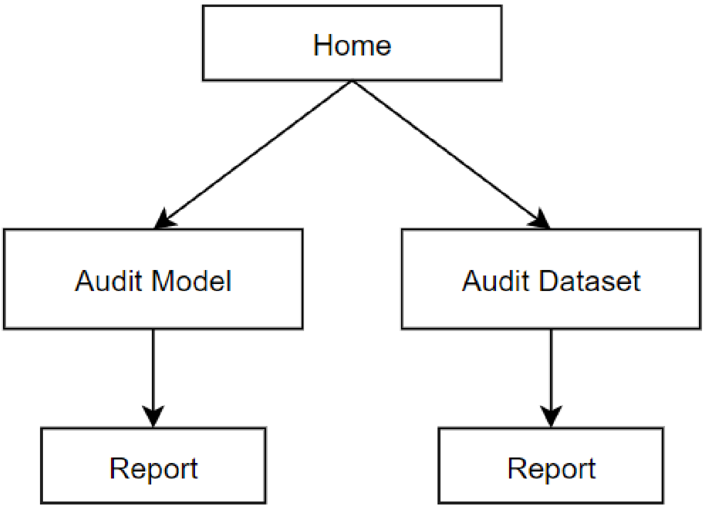


Figure 4.9: Web Application Navigation Diagram.

CHAPTER 5

Case study

5.1 Introduction

In this chapter, the functionalities created and described on chapter four will be shown with examples for both the Jupyter Notebook and the web application. For this purpose, the COMPAS [25] dataset will be used. More specifically, a one hot encoded version that has been taken from the FairML repository [32].

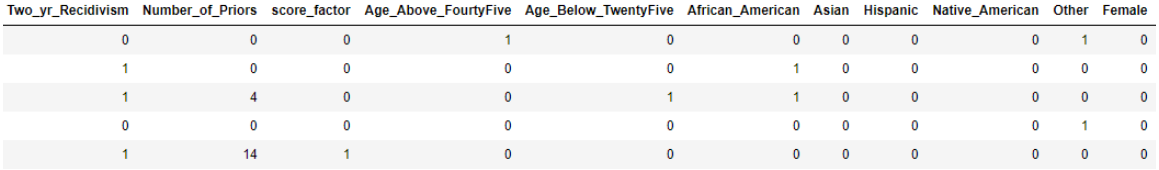
Since the COMPAS model is proprietary software, and only for the purpose of showing all the functionalities that this tool can provide, a very simple Linear Regression Model has been created to mock its behaviour. However, this does not mean that results shown for the model part are accurate in any way.

Nevertheless, the section where a dataset is audited is based on factual data and the results can be interpreted as real and accurate.

5.2 Jupyter Notebook

5.2.1 Use Case 1: Audit Dataset

In this section, the COMPAS dataset from the FairML repository will be audited. It is a non-categorized dataset, in Figure 5.1 its head is shown. The purpose of this case study is to audit the dataset in order to look for possible racial bias and gender bias.



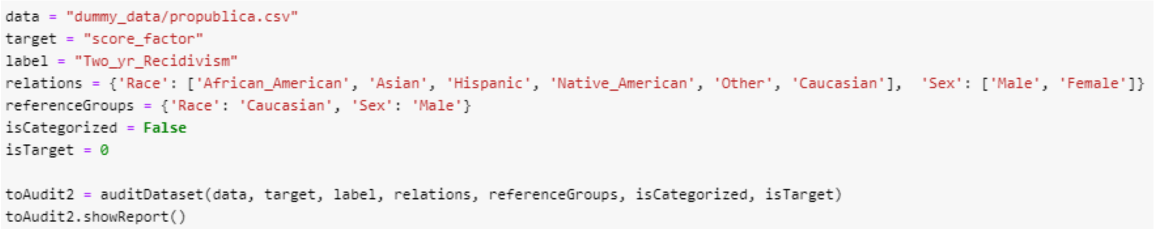
Two_yr_Recidivism	Number_of_Priors	score_factor	Age_Above_FourtyFive	Age_Below_TwentyFive	African_American	Asian	Hispanic	Native_American	Other	Female
0	0	0	1	0	0	0	0	0	1	0
1	0	0	0	0	1	0	0	0	0	0
1	4	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0
1	14	1	0	0	0	0	0	0	0	0

Figure 5.1: Case Study Dataset Head

In order to audit the dataset, the necessary input parameters are given to create an "AuditData" object. After that, the function to request the report is executed (Figure 5.2). These parameters consist on:

- ***data***: A relative route to the CSV file with the dataset used for the model containing the different attributes, the outcome (score) and the ground truth data (label).
- ***target***: A string with the name of the column that stores the results of the model.
- ***label***: A string with the name of the column of the dataset that represents the ground truth data.
- ***relations***: A dictionary including the groups to be audited with its possible values. This parameter is only needed if the dataset is not categorical.
- ***referenceGroups***: A dictionary stating the reference group to audit the bias against.
- ***isTarget***: An Integer. It indicates if the outcome of the model that has a positive connotation is 0 or 1.
- ***isCategorized***: A boolean parameter indicating whether the dataset is categorized or not.

The report's outputs (from figures 5.3 to 5.12) are given in a single notebook cell.



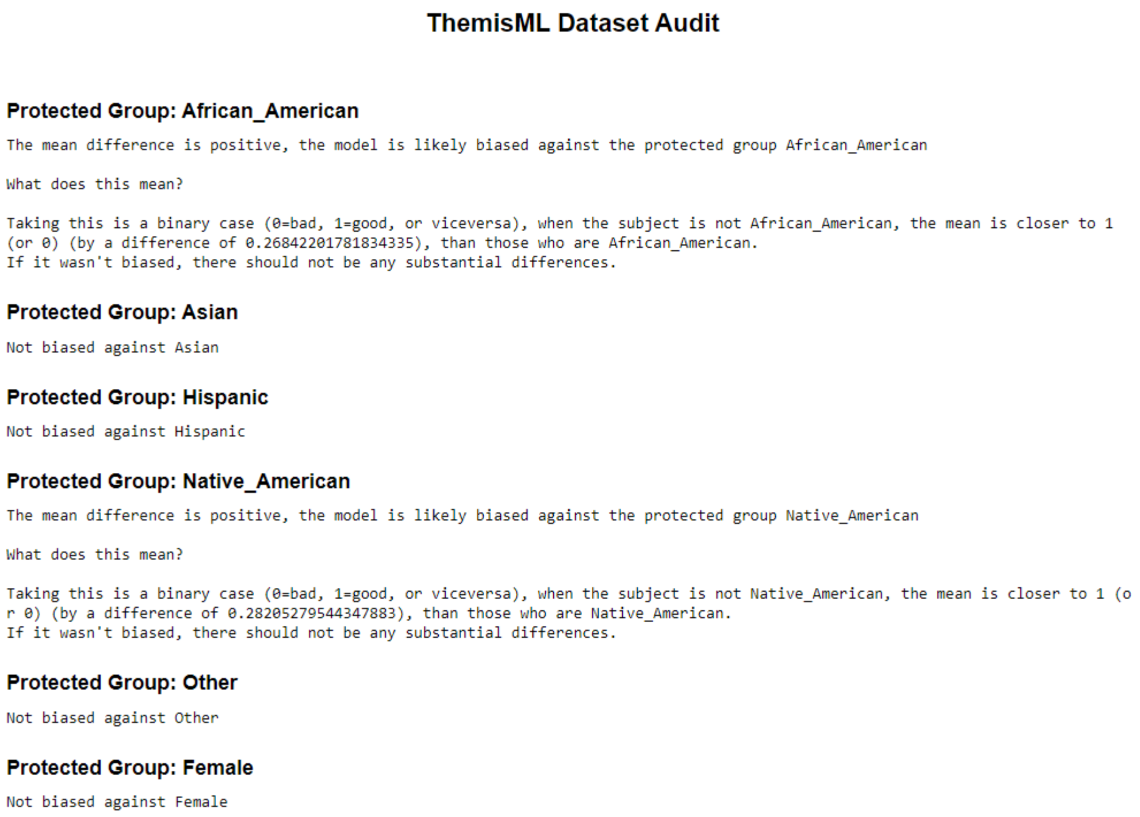
```
data = "dummy_data/propublica.csv"
target = "score_factor"
label = "Two_yr_Recidivism"
relations = {'Race': ['African_American', 'Asian', 'Hispanic', 'Native_American', 'Other', 'Caucasian'],  'Sex': ['Male', 'Female']}
referenceGroups = {'Race': 'Caucasian', 'Sex': 'Male'}
isCategorized = False
isTarget = 0

toAudit2 = auditDataset(data, target, label, relations, referenceGroups, isCategorized, isTarget)
toAudit2.showReport()
```

Figure 5.2: Creation of the AuditDataset object and execution the reporting function

The output that Themis-ML produces is provided on Figure 5.3. It consists on a quick glance of the possible presence of bias for the requested groups. From the output, the following data can be extracted:

- There is not presence of gender bias.
- In respect of racial bias, it has been detected for the groups “African_American” and “Native_American”.



ThemisML Dataset Audit

Protected Group: African_American

```
The mean difference is positive, the model is likely biased against the protected group African_American

What does this mean?

Taking this is a binary case (0=bad, 1=good, or viceversa), when the subject is not African_American, the mean is closer to 1
(or 0) (by a difference of 0.26842201781834335), than those who are African_American.
If it wasn't biased, there should not be any substantial differences.
```

Protected Group: Asian

```
Not biased against Asian
```

Protected Group: Hispanic

```
Not biased against Hispanic
```

Protected Group: Native_American

```
The mean difference is positive, the model is likely biased against the protected group Native_American

What does this mean?

Taking this is a binary case (0=bad, 1=good, or viceversa), when the subject is not Native_American, the mean is closer to 1 (o
r 0) (by a difference of 0.28205279544347883), than those who are Native_American.
If it wasn't biased, there should not be any substantial differences.
```

Protected Group: Other

```
Not biased against Other
```

Protected Group: Female

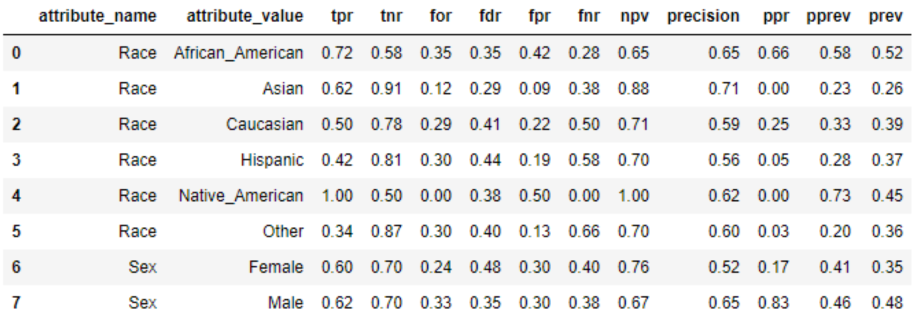
```
Not biased against Female
```

Figure 5.3: Output 1: Themis-ML bias calculations.

From Figures 5.4 to 5.12, the outputs that Aequitas provides for the report are shown. They consist on a more detailed and graphic way of studying the possible presence of bias in the dataset and may detect some parameters that Themis-ML can not. On Figure 5.9 the overall fairness determination is stated and on Figure 5.10 is shown as a plot to be better understood. From the output, it has been learned that:

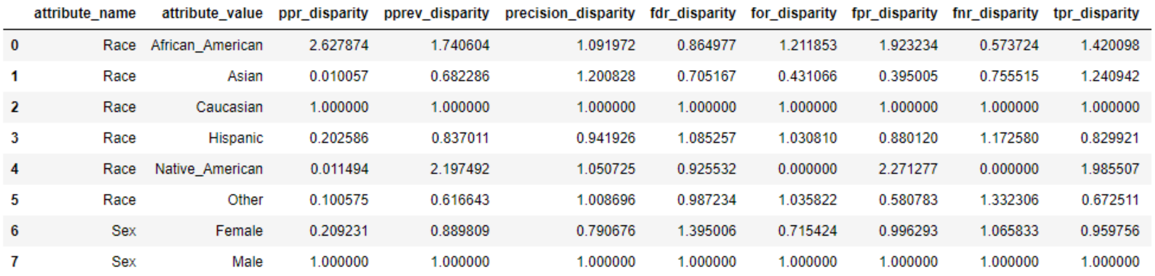
- ***Gender***: Although the model seemed fair for "Female", on Figure 5.12 disparity for this group on the False Omission Rate, False Discovery Rate, Precision Disparity and Predicted Positive Rate is seen. This means that the fairness requirements have not been met, and that is biased against females.
- ***Race***: As Themis-ML already indicated, on Figure 5.11 disparity against the groups "African_American" and "Native_American" is seen for almost every metric. However, not only these groups, but all the others fail to meet the fairness requirement when compared to "Caucasian" in at least one metric. This means that one way or another, the model is racially biased against all groups.

The definition and meaning of the metrics have already been explained on Section 3.4.1 and there is further explanation of them on Appendix D.



	attribute_name	attribute_value	tpr	tnr	for	fdr	fpr	fnr	npv	precision	ppr	pprev	prev
0	Race	African_American	0.72	0.58	0.35	0.35	0.42	0.28	0.65	0.65	0.66	0.58	0.52
1	Race	Asian	0.62	0.91	0.12	0.29	0.09	0.38	0.88	0.71	0.00	0.23	0.26
2	Race	Caucasian	0.50	0.78	0.29	0.41	0.22	0.50	0.71	0.59	0.25	0.33	0.39
3	Race	Hispanic	0.42	0.81	0.30	0.44	0.19	0.58	0.70	0.56	0.05	0.28	0.37
4	Race	Native_American	1.00	0.50	0.00	0.38	0.50	0.00	1.00	0.62	0.00	0.73	0.45
5	Race	Other	0.34	0.87	0.30	0.40	0.13	0.66	0.70	0.60	0.03	0.20	0.36
6	Sex	Female	0.60	0.70	0.24	0.48	0.30	0.40	0.76	0.52	0.17	0.41	0.35
7	Sex	Male	0.62	0.70	0.33	0.35	0.30	0.38	0.67	0.65	0.83	0.46	0.48

Figure 5.4: Output 2: Group Metrics Table.



	attribute_name	attribute_value	ppr_disparity	pprev_disparity	precision_disparity	fdr_disparity	for_disparity	fpr_disparity	fnr_disparity	tpr_disparity
0	Race	African_American	2.627874	1.740604	1.091972	0.864977	1.211853	1.923234	0.573724	1.420098
1	Race	Asian	0.010057	0.682286	1.200828	0.705167	0.431066	0.395005	0.755515	1.240942
2	Race	Caucasian	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
3	Race	Hispanic	0.202586	0.837011	0.941926	1.085257	1.030810	0.880120	1.172580	0.829921
4	Race	Native_American	0.011494	2.197492	1.050725	0.925532	0.000000	2.271277	0.000000	1.985507
5	Race	Other	0.100575	0.616643	1.008696	0.987234	1.035822	0.580783	1.332306	0.672511
6	Sex	Female	0.209231	0.889809	0.790676	1.395006	0.715424	0.996293	1.065833	0.959756
7	Sex	Male	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Figure 5.5: Output 4: Bias Metrics Table.

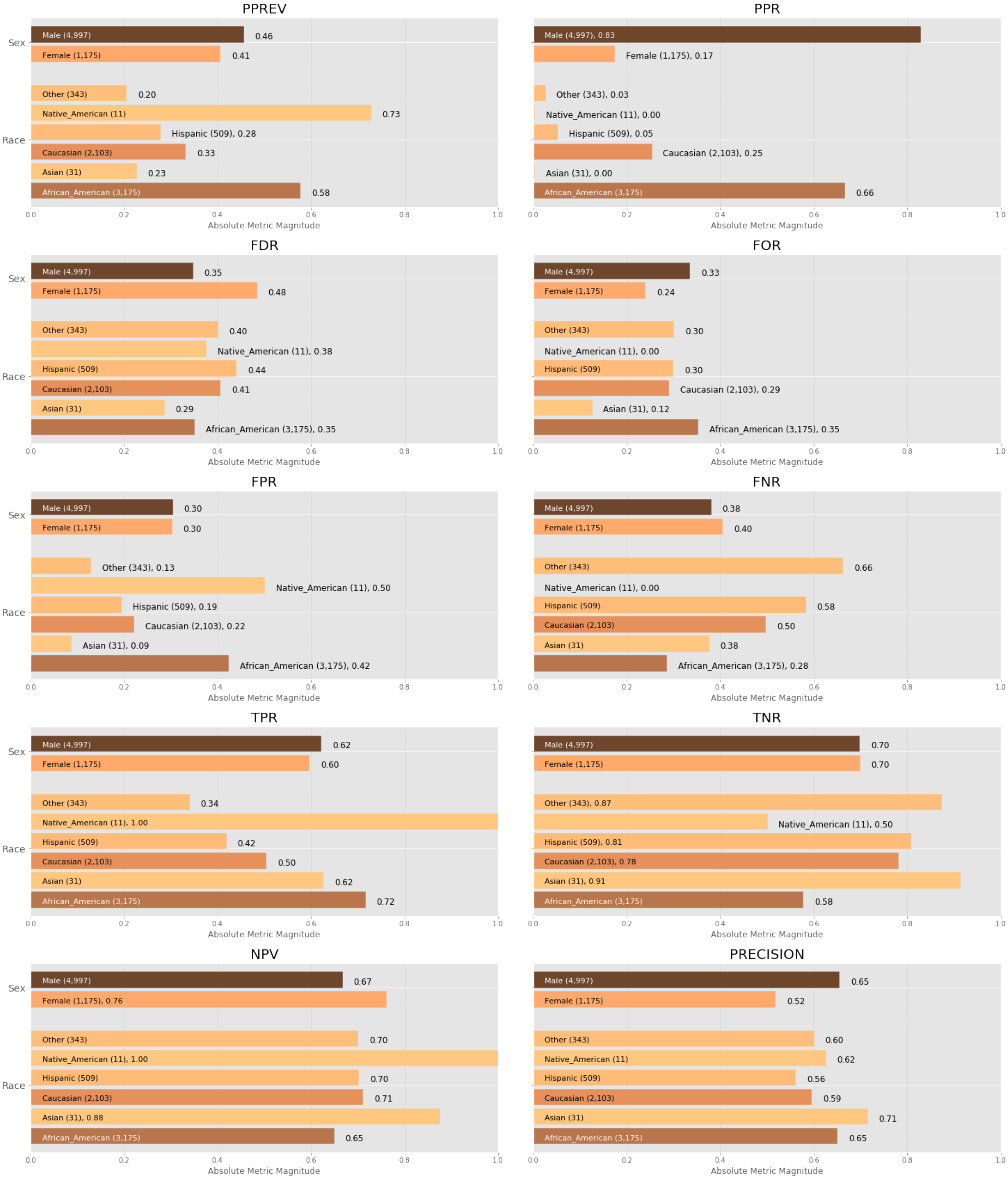


Figure 5.6: Output 3: Group Metrics Graphs.

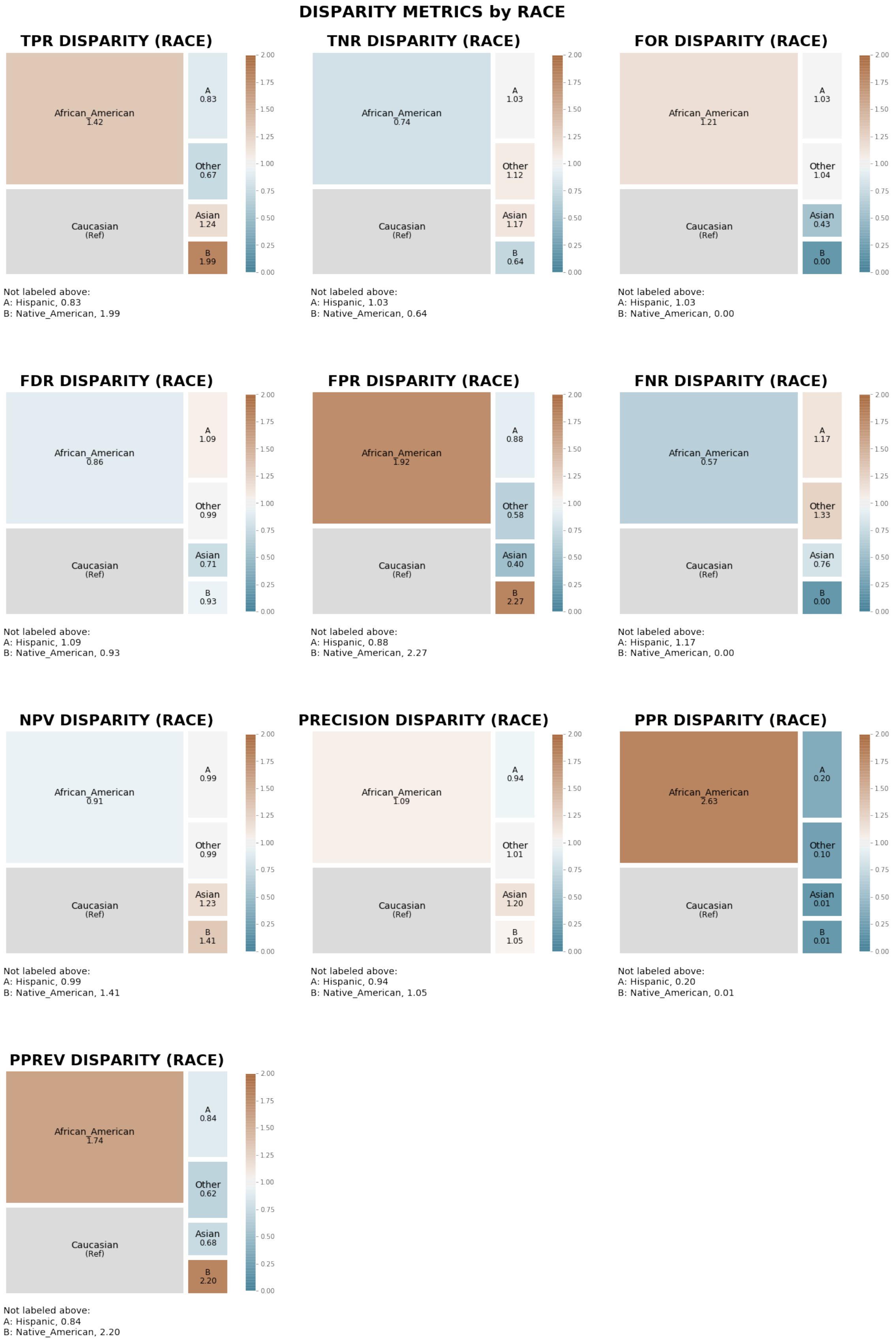


Figure 5.7: Output 5: Bias Metrics Graphs (1).

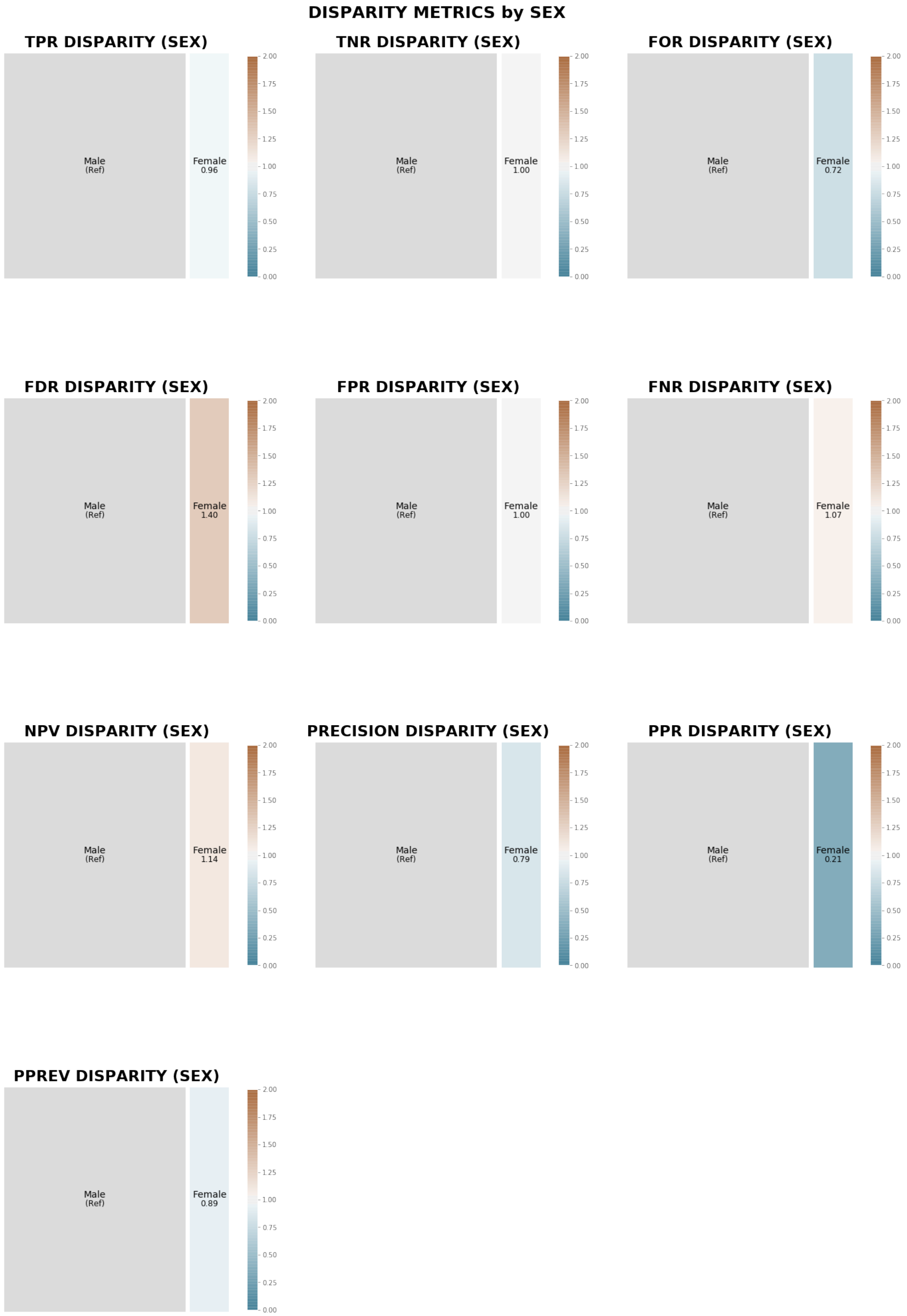
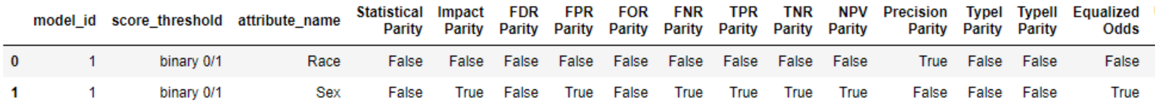


Figure 5.8: Output 6: Bias Metrics Graphs (2).



	model_id	score_threshold	attribute_name	Statistical Parity	Impact Parity	FDR Parity	FPR Parity	FOR Parity	FNR Parity	TPR Parity	TNR Parity	NPV Parity	Precision Parity	TypeI Parity	TypeII Parity	Equalized Odds
0	1	binary 0/1	Race	False	False	False	False	False	False	False	False	False	True	False	False	False
1	1	binary 0/1	Sex	False	True	False	True	False	True	True	True	True	False	False	False	True

Figure 5.9: Output 7: Fairness Determination Table.

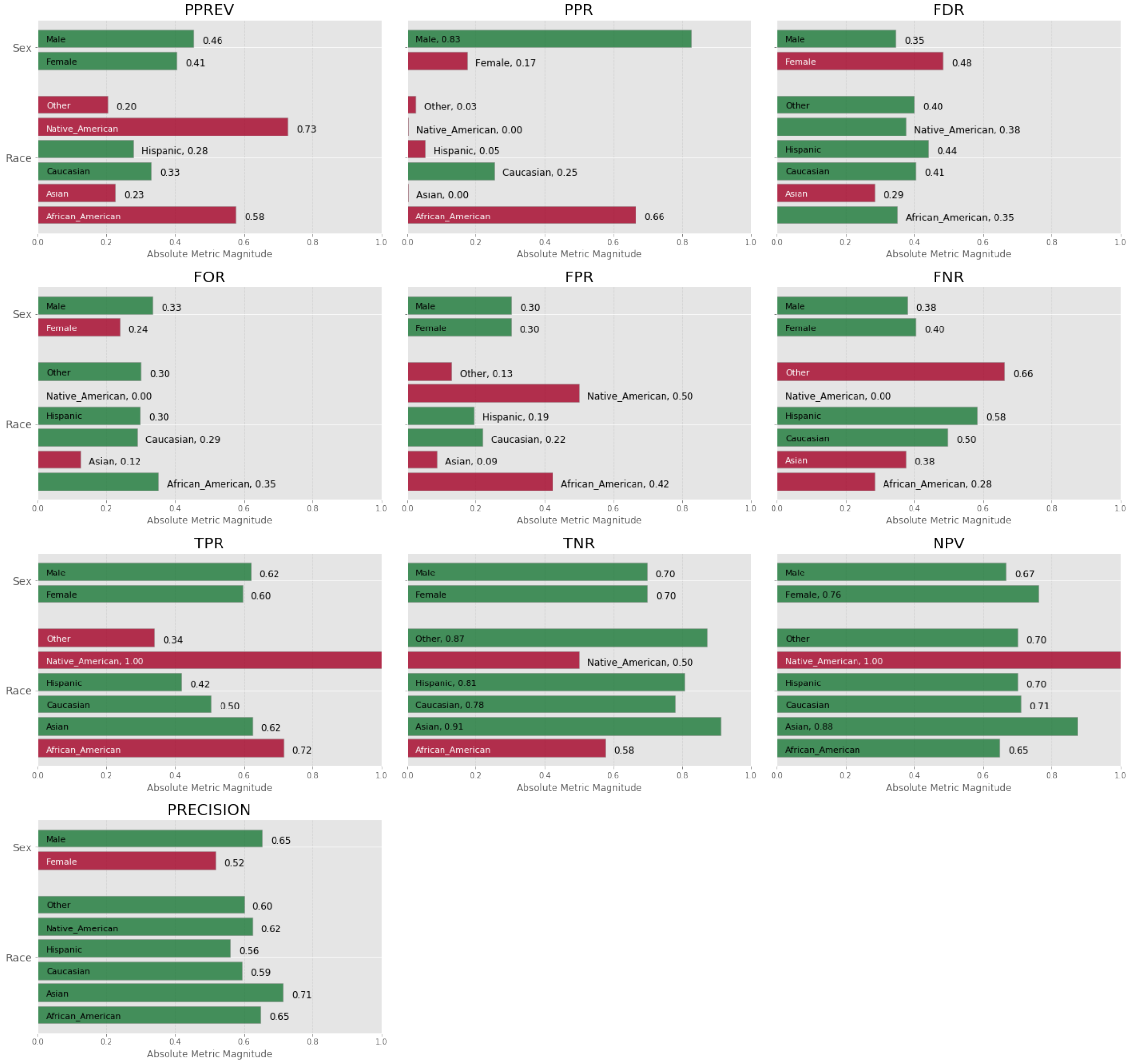


Figure 5.10: Output 8: Group Fairness Determination Graphs.

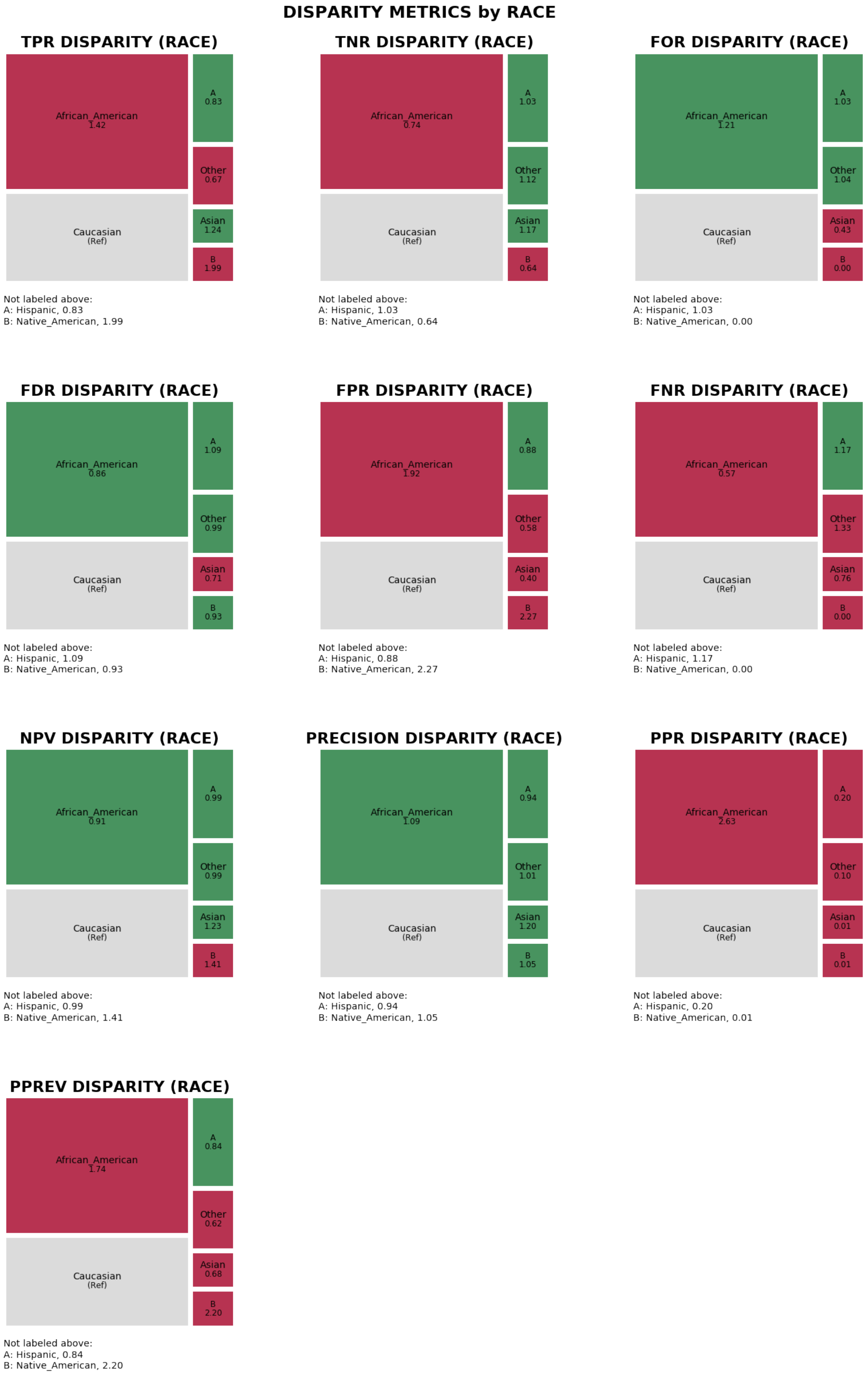


Figure 5.11: Output 9: Single Fairness Determination Graphs (1).

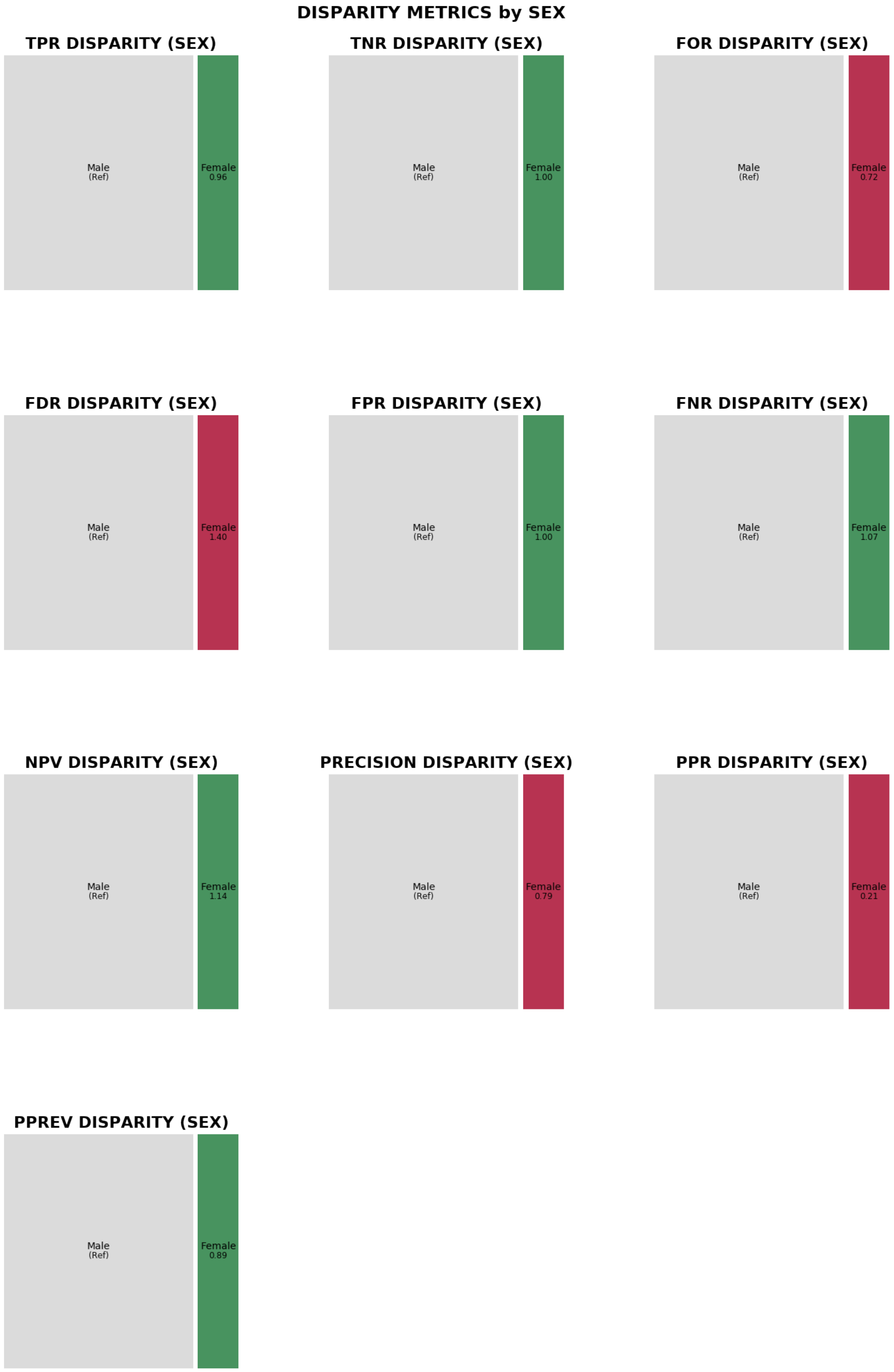
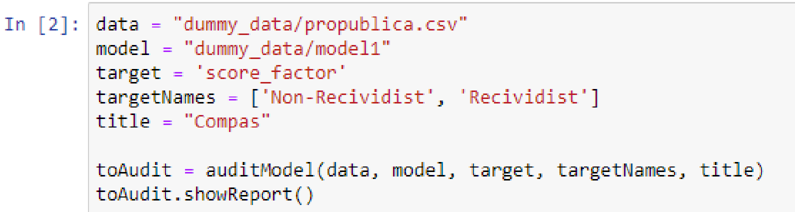


Figure 5.12: Output 10: Single Fairness Determination Graphs (2).

5.2.2 Use Case 2: Audit Model

A Linear Regression mock model has been created to imitate the COMPAS model behaviour. The purpose of this case study is to see what features does the model rely on to make the predictions and debate if it is gender or racially biased. The necessary input parameters are given in order to create the object. After that, the function to request the report is executed (Figure 5.13). The report's outputs (Figures 5.14 and 5.15) are given in a single notebook cell. The parameters needed are:

- ***data***: A relative route to the CSV file with the dataset used to train the model.
- ***model***: A relative route to the joblib export of the model.
- ***target***: Consists on a string that indicates the name of the column that stores the results of the model.
- ***targetNames***: Consists on a string array that indicates the meaning of the binary target values.
- ***title***: A string that indicates the name of the audited model.



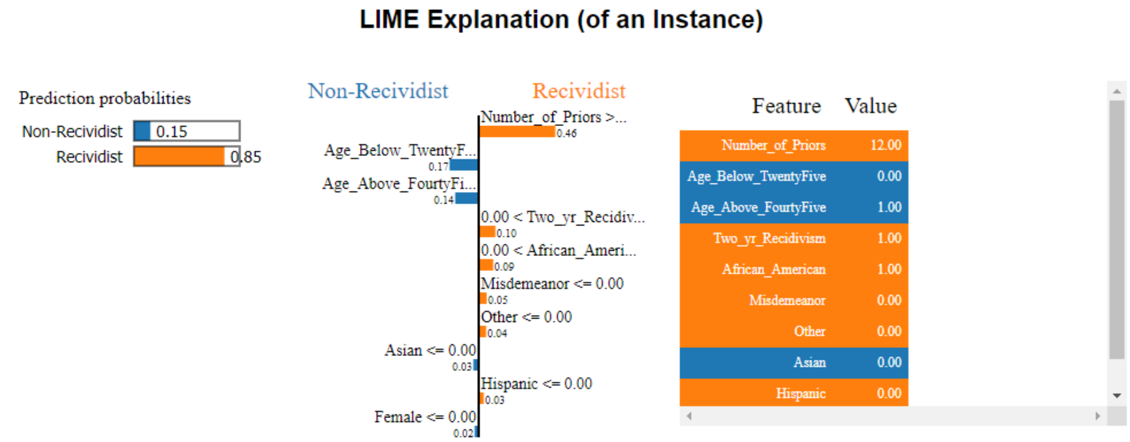
```
In [2]: data = "dummy_data/propublica.csv"
        model = "dummy_data/model1"
        target = 'score_factor'
        targetNames = ['Non-Recividist', 'Recividist']
        title = "Compas"

        toAudit = auditModel(data, model, target, targetNames, title)
        toAudit.showReport()
```

Figure 5.13: Creation of the AuditModel object and execution the reporting function

On Figure 5.14 the explanation of a prediction is made with LIME. It can be seen that the most probable outcome for this values, shown at the table at the right of the figure, is "Recividist". The number of priors and if the subject has been a re-offender the last two years are the most important features towards deciding if a subject is a possible Recividist.

However, the third one is if the subject is African American, coming even before committing a misdemeanor. Just by seeing this, it can be determined that the model is racially biased. Nevertheless, in regard to gender, females is at the bottom of the chart, meaning it hasn't been a key feature in the decision, so it can also be determined that it is not gender biased.

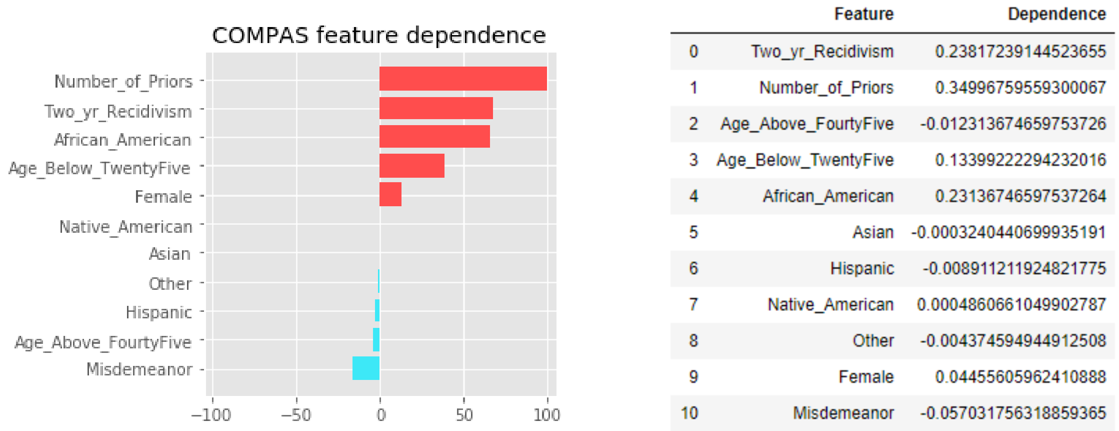


Feature	Value
Number_of_Priors	12.00
Age_Below_TwentyFive	0.00
Age_Above_FourtyFive	1.00
Two_yr_Recidivism	1.00
African_American	1.00
Misdemeanor	0.00
Other	0.00
Asian	0.00
Hispanic	0.00

Figure 5.14: Output 1: LIME explanation of an instance.

On Figure 5.15 the explanation of the feature dependence of the model is made with FairML. In the plot, it can be seen that the third most dependent feature for "Recividist" (right means 1 and left means 0) is if the subject is African American, even coming before age factors.

For the feature Female there is not a big relative significance. From this, the conclusion that the model is in fact racially biased against African Americans can be determined.



	Feature	Dependence
0	Two_yr_Recidivism	0.23817239144523655
1	Number_of_Priors	0.34996759559300067
2	Age_Above_FourtyFive	-0.012313674659753726
3	Age_Below_TwentyFive	0.13399222294232016
4	African_American	0.23136746597537264
5	Asian	-0.00032404406999935191
6	Hispanic	-0.008911211924821775
7	Native_American	0.00048606610499902787
8	Other	-0.004374594944912508
9	Female	0.04455605962410888
10	Misdemeanor	-0.05703175631885936 5

(a) FairML Feature Dependence Graph

(b) FairML Feature Dependence Table

Figure 5.15: Output 2: FairML feature dependence table and plot output.

5.3 Web Application

For the case of the web application, once the user has arrived to the landing page, two options, audit data and audit model, will be presented (Figure 5.16). Depending on the button clicked, either a form asking for the necessary parameters to audit a model (Figure 5.18) or a dataset (Figure 5.19) will be shown. The fields of the forms are changed dynamically depending on the user's inputs. After clicking the respective "Audit" buttons, a report just like in the Jupyter Notebook will appear. The web page also contains some information referencing the libraries used and an about section (Figure 5.17).

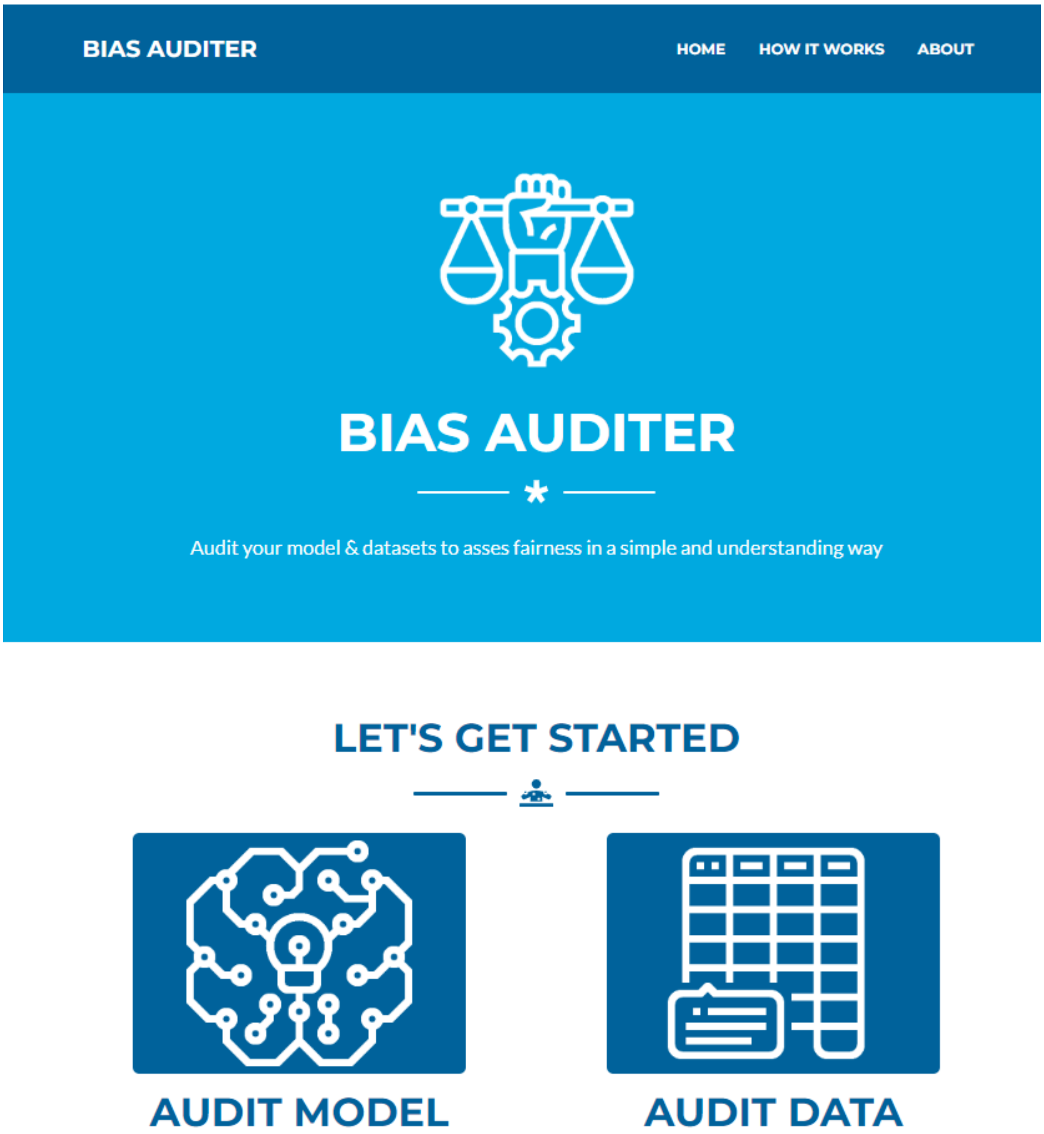
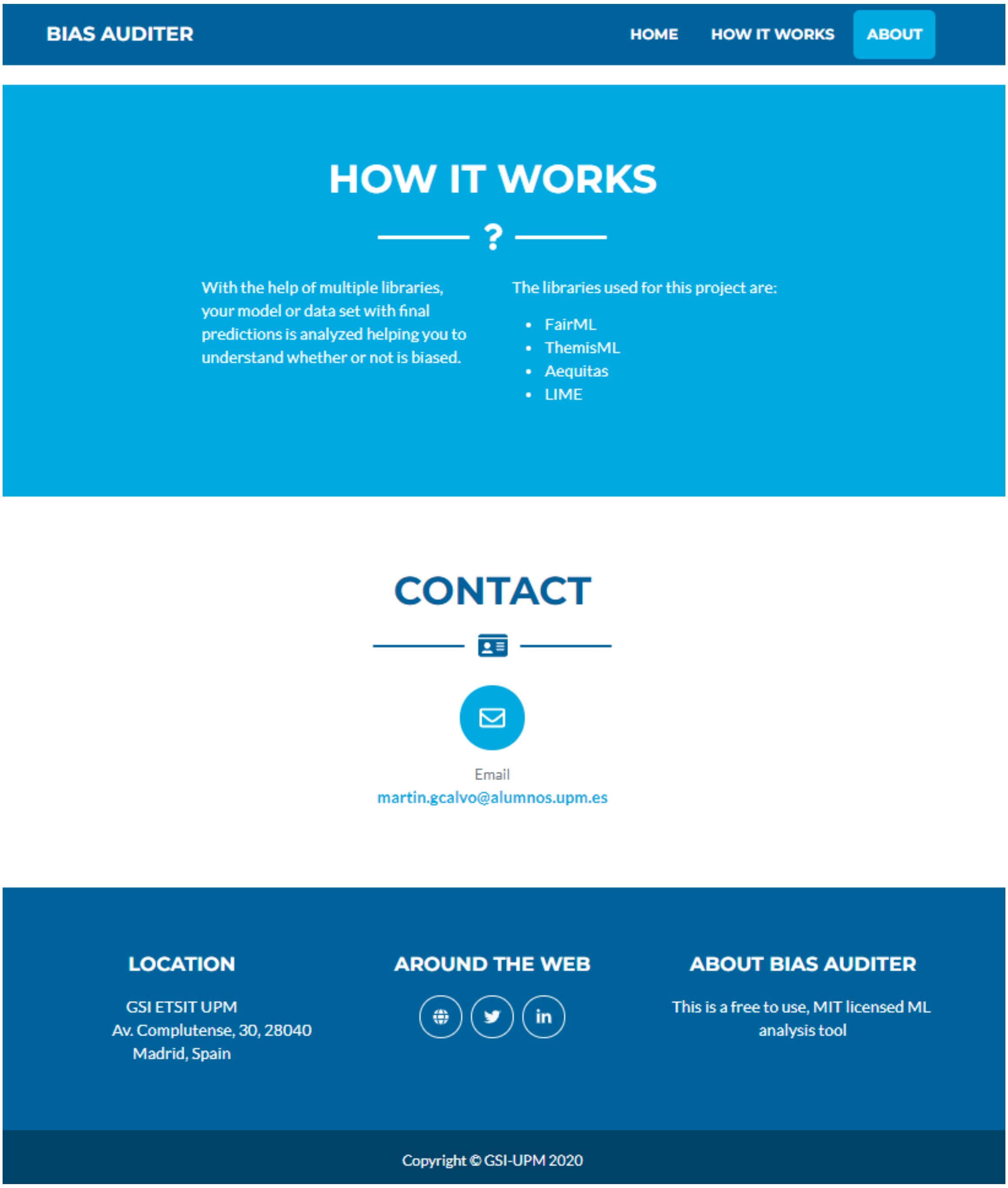


Figure 5.16: Web Application Landing Page (1).



HOW IT WORKS

?

With the help of multiple libraries, your model or data set with final predictions is analyzed helping you to understand whether or not is biased.

The libraries used for this project are:

- FairML
- ThemisML
- Aequitas
- LIME

CONTACT

Email
martin.gcalvo@alumnos.upm.es

LOCATION

GSI ETSIT UPM
Av. Complutense, 30, 28040
Madrid, Spain

AROUND THE WEB

ABOUT BIAS AUDITER

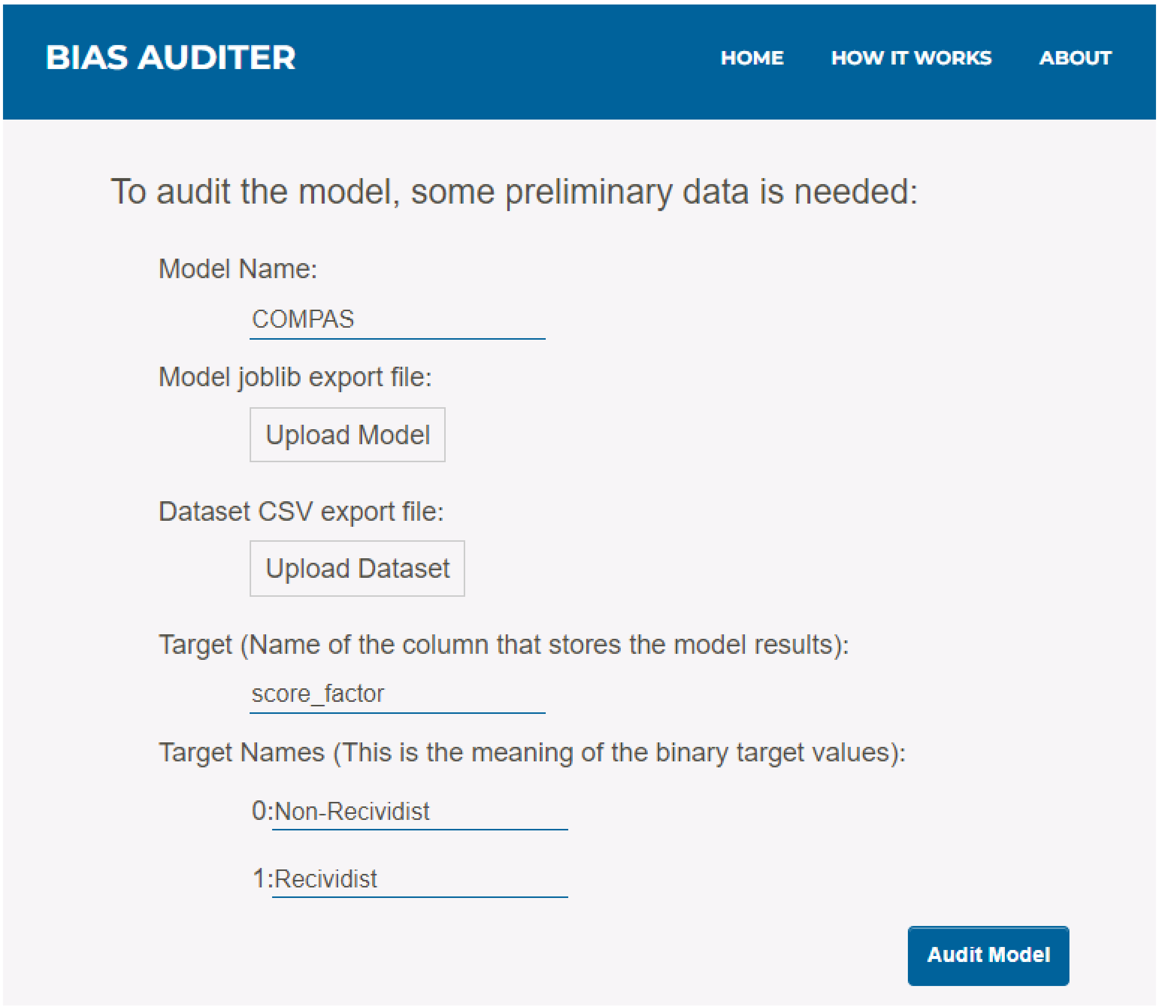
This is a free to use, MIT licensed ML analysis tool

Copyright © GSI-UPM 2020

Figure 5.17: Web Application Landing Page (2).

The parameters needed to audit a dataset or a model are asked in the respective forms. The inputs in the “Audit Dataset” form may be different depending on the case, so the fields will change dynamically depending on the options chosen by the user.

Once the different parameters are introduced, the “Audit” button will be clicked and the user will be taken to a page presenting the same results shown on Section 5.2.1 for the case of Figure 5.19 and the results on Section 5.2.2 for the case of Figure 5.18.



To audit the model, some preliminary data is needed:

Model Name:

COMPAS

Model joblib export file:

Dataset CSV export file:

Target (Name of the column that stores the model results):

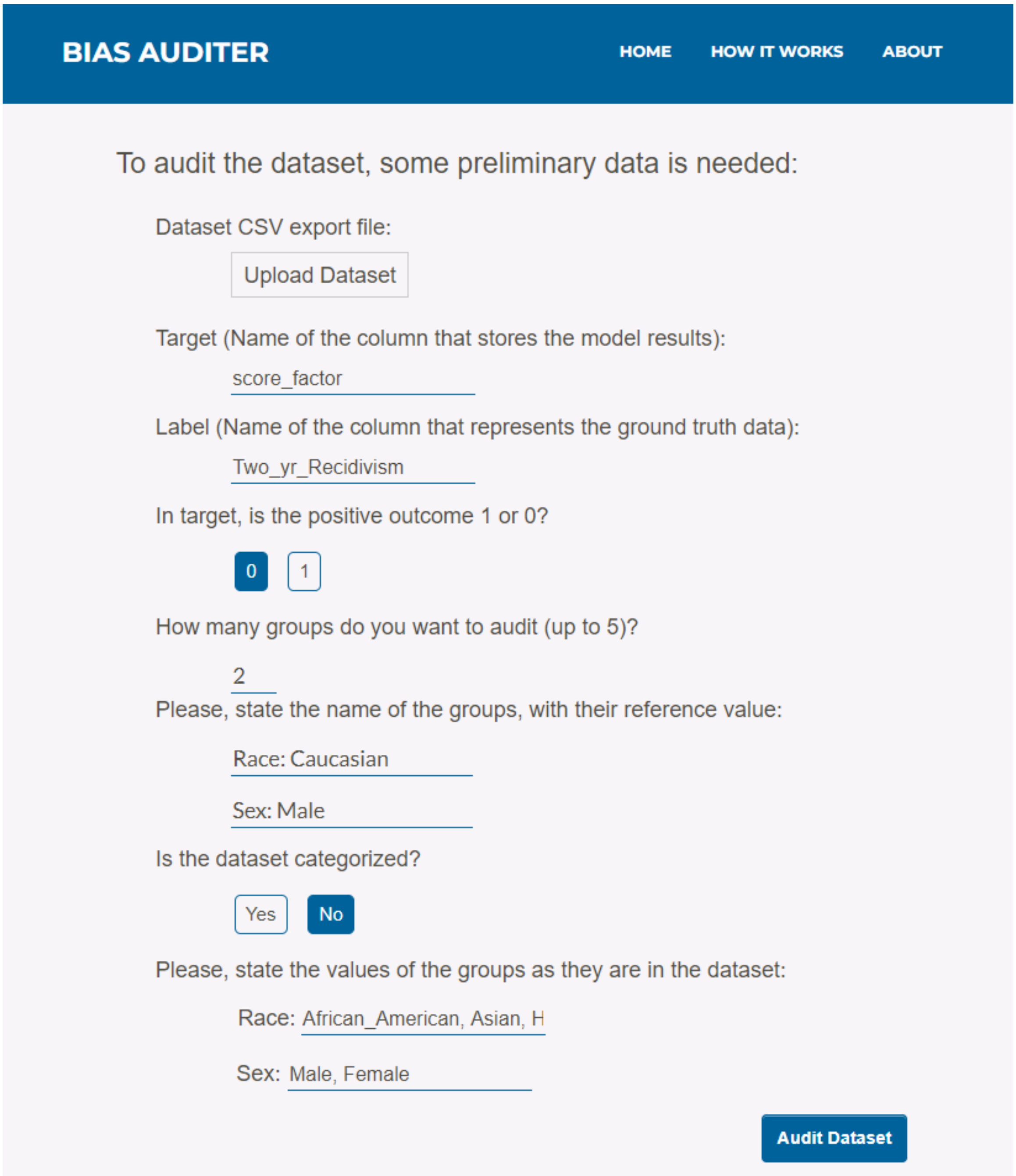
score_factor

Target Names (This is the meaning of the binary target values):

0:Non-Recividist

1:Recividist

Figure 5.18: Audit Model Form.



To audit the dataset, some preliminary data is needed:

Dataset CSV export file:

Target (Name of the column that stores the model results):

score_factor

Label (Name of the column that represents the ground truth data):

Two_yr_Recidivism

In target, is the positive outcome 1 or 0?

[✓ 0] 1

How many groups do you want to audit (up to 5)?

2

Please, state the name of the groups, with their reference value:

Race: Caucasian

Sex: Male

Is the dataset categorized?

Yes [✓ No]

Please, state the values of the groups as they are in the dataset:

Race: African_American, Asian, H

Sex: Male, Female

Figure 5.19: Audit Dataset Form.

CHAPTER 6

Conclusions and future work

In this chapter the conclusions extracted from this project, and the thoughts about future work will be described.

6.1 Conclusions

The main purpose of this project was to build a bias detection tool for ML algorithms. In the end, the tool does not only analyze the output that a model produces (the section described as "Audit Dataset") but it also takes care of analyzing the inner mechanisms of a model and how it affects to the final result ("Audit Model"). For this purpose four libraries have been implemented:

- ***Model Explainers***: LIME and FairML.
- ***Data Analyzers***: Aequitas and Themis-ML.

To sum up, the development of a bias analysis tool has been achieved, but also, the possibility for the tool to explain how a model works has been introduced. This can provide more insight to the final user about how a model works, and specially, if it should work that way, which could help to reach the conclusion that in fact, the model is biased.

6.2 Achieved Goals

The goals achieved for this project are:

- ***Development of a bias detection tool***: The main goal, the development of a bias detection tool for ML algorithms, has been achieved. Not only as a Jupyter Notebook for people with technical skills, also as a web platform oriented to all kinds of people.
- ***Development of a ready to deploy web application***: A web platform has been developed so the tool can be deployed and be of easy access for everyone that wishes to use it. There, guidance for the different inputs is given.

6.3 Problems Faced

During the course of this project, the main problems encountered have been:

- ***Learning ML from scratch***: Thanks to the documentation provided by the supervisor of this work and multiple online resources, the learning curve has been exponential.
- ***Standardize libraries as a whole***: The process of having all the libraries working as a whole and being integrated as a unique tool, has been troublesome.

6.4 Future Work

The possible next steps to improve the tool are:

- ***Categorical Models***: The tool only has full functionality for models that work with one hot encoded data. Whereas is true that some parts can function with categorical data, not all of it does. Model explainers for categorical data could be introduced.
- ***Image Classification Models***: This discipline of ML is one of the most widespread. However, the auditor is not able to work with this type of data because of its nature. New functionalities related to image classification models could be introduced.
- ***Add more libraries***: Four libraries have been implemented. Nevertheless, there are plenty of more libraries related to analyzing bias in ML models. Their functionality could be researched and be added of the tool without overlapping current ones.

APPENDIX A

Impact of this project

The purpose of this appendix is to explain the possible implications from a social, environmental and ethical point of view.

A.1 Social Impact

The platform developed for this project could conduct to fairer machine learning models. As it has been stated at the begining of this document, algorithmic fairness is a growing problem. Offering a tool to assess fairness makes the process easier and accessible for more people. To sum up, the existence of this tool can represent a very positive social impact

A.2 Environmental Impact

The current environmental impact of the project is very low, since only the contaminating parts of the computer that has been used are to take into account. However, if the service is to be deployed as a Platform as a Service, the possible pollution derived from power consumption of the data center where the service is deployed should be taken into account.

A.3 Ethical Implications

Regarding the privacy, in the case of the web being deployed, the processed data would not be stored in any way, so users can be assured that their model or data will be safe. Also in the case of not being able to upload it for different reasons, the Jupyter Notebook version can be used.

Regarding the fairness assessment, it should always be supervised by a person with enough knowledge to understand the different outputs and never be taken as a single point of truth, since errors may take place.

APPENDIX B

Economic budget

The purpose of this appendix is to detail the budget for the project, going from the structure followed to carry it out to all the resources used.

B.1 Project Structure

The project structure is shown in the following Gantt chart:

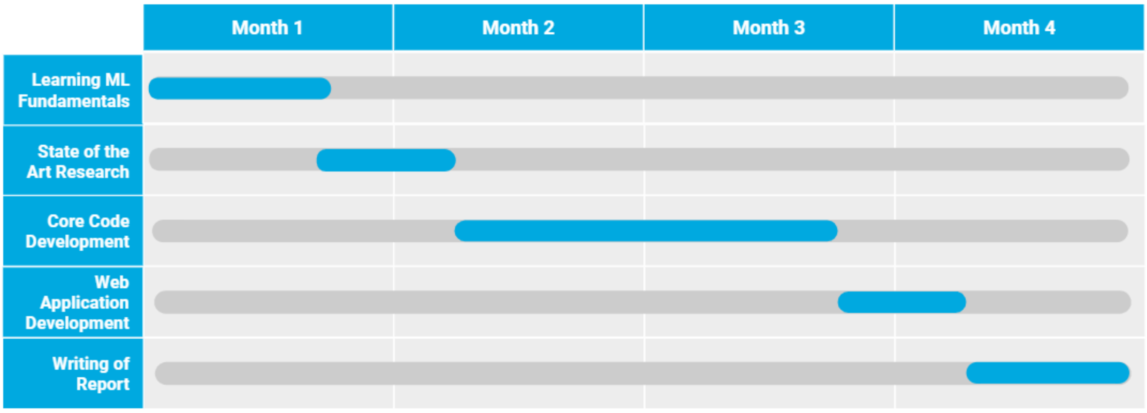


Figure B.1: Project's Structure Gantt's Chart

B.2 Physical resources

To carry out this project the resources that have been used are:

- ***Software***: Regarding this matter, only open source software has been used. The operative system used is the open source distribution Linux Mint and all the development tools are included in Anaconda Individual Edition, which is also open source, so there is no cost associated to the software for this project.
- ***Hardware***: The only hardware used is a laptop computer, the model is ASUS F541U and has the following characteristics:
  - CPU: Intel Core i7-6500U
  - RAM: 16 GB
  - HDD: 1 TB
  - SDD: 500 GB

  The total cost of this equipment, since the SDD has been an enhancement after the purchase is an approximate of **940€**.

B.3 Human Resources

Regarding the cost related to human resources, only one person has been needed to complete the project during a 4 month period. Taking into account that the average salary for a machine learning engineer in Spain is 35.000€ per year, Spanish taxes already included.

With this information the total cost related to human resources is: **11.667€**.

B.4 Taxes

This project is not meant for sale, is meant to be used as an open source tool, therefore there are not any tax considerations to take into account.

B.5 Conclusion

The total cost of the project is 21.307€ and has a total duration of four months.

APPENDIX C

Making Aequitas Work

The biggest problem with Aequitas is that, to properly work, we need to regroup the variables we want to audit that are one hot encoded into categorical data.

C.1 Categorizing the dataset

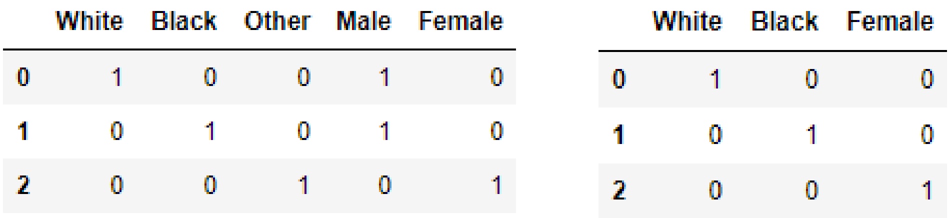
For example, in the case of having three columns, each of them to indicate whether a subject is white, black or other race, this data would need to be grouped in a single column indicating so.

The case where there is not a column to specify a certain subject is something that could also be encountered (e.g. we have one column Female, so is comprehensible that those rows with value 0 equal Male).

In order to do this only two inputs are needed.

- ***Data***: The dataset that is to be audited
- ***Relations***: A dictionary indicating the different relations between the different variables. (e.g. relations = ‘Race’: [‘Black’, ‘White’, ‘Other’], ‘Sex’: [‘Male’, ‘Female’])

To sum up, there is a need for a process to convert these:



	White	Black	Other	Male	Female
0	1	0	0	1	0
1	0	1	0	1	0
2	0	0	1	0	1

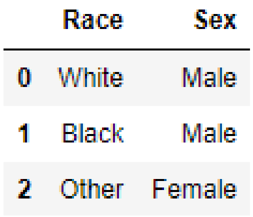
(a) No missing columns

	White	Black	Female
0	1	0	0
1	0	1	0
2	0	0	1

(b) Missing columns

Figure C.1: Examples of one hot encoded input datasets for Aequitas

Into this:



	Race	Sex
0	White	Male
1	Black	Male
2	Other	Female

Figure C.2: Example of a categorical dataset

For this purpose, some auxiliary functions have been created.

C.2 Auxiliary Preprocessing Functions

A total of two functions have been created in order to process the datasets:

- ***checkIfMissingAdd(df, relations)***: This function takes as inputs the two variables stated above and checks for the case of the dataset not having all the necessary columns (Figure C.1 (b)). In the case of not having them, it adds them.

- ***categorize(df, relations):***: This function takes as inputs the two variables stated above and returns a processed, categorized dataset.

APPENDIX D

Python Classes

In this appendix, a more in depth explanation of the classes implemented in this project is given.

D.1 Class limeTFG

This class has the purpose of implementing the library LIME [7]. In order to create the object four inputs are needed:

- ***data***: Consists on a CSV file with the dataset used for the model.
- ***model***: Consists on a joblib export of the model.
- ***target***: Consists on a string that indicates the name of the column that stores the results of the model.
- ***targetNames***: Consists on a string array that indicates the meaning of the binary target values ([“meaning of 0”, “meaning of 1”]).

The following methods have been created:

- ***__init__(self, data, model, target, targetNames)***: Acquires the model, the data, the target column name, and the meaning of the target values. After that, the LIME explainer is created and a first instance of an explanation.
- ***createExplainer(self)***: Creates the LIME explainer. This function is not to be called since it is invoked by the *__init__()* function.
- ***createExplanation(self)***: Acquires the explanation of an instance, there are as many explanations as rows of input data. For that purpose, a random number is generated first, and an explanation is chosen later, so every explanation rendered later can be different if the user chooses so.
- ***showInNotebook(self)***: Shows in a Jupyter Notebook the HTML explanation of the last instance created (There is an example of this in section 3.4.4).
- ***saveHTMLtoFile(self, path)***: Saves in the specified path an HTML file the explanation of the last instance created.
- ***asMatplotlib(self)***: Renders the explanation of the last instance created as a Matplotlib figure.

D.2 Class fairMLTFG

This class implements FairML [32]. In order to create the object four inputs are needed:

- ***data***: A CSV file with the dataset used for the model.
- ***model***: A joblib export of the model.
- ***target***: A string with the name of the column that stores the results of the model.
- ***name***: A string that indicates the name of the audited model.

Three methods have been created:

- ***__init__(self, data, model, target, title)***: Acquires the model, the data, the target column name and the desired title. After that the FairML auditing function is invoked and the results are saved into an internal variable within the class.
- ***getFeatureDependenceGraphic(self)***: Plots a matplotlib figure with the data.
- ***getFeatureDependeTable(self)***: It treats the data through some string operations and returns a pandas dataset of the feature dependencies.

D.3 Class themisMLTFG

This class implements the library Themis-ML [33]. In order to create an object, the following parameters are needed:

- ***data***: A CSV file with the dataset used for the model.
- ***target***: A string with the name of the column that stores the results of the model.
- ***isTarget***: This parameter must be 0 or 1. It indicates if the outcome of the model that has a positive connotation is 0 or 1.

The class implements the following methods:

- ***__init__(self, data, target, isTarget)***: Acquires the data, the target column name and the variable that indicates if the outcome of the model that has a positive connotation is 0 or 1.
- ***getMeanDifference(self, protectedAttribute)***: Calculates the mean difference in respect to a protected group. This means that if the model is not biased against a certain protected group, the mean difference of the result between the overall population and a protected group (for example, women, or black people) should not be substantial.

  For example, if it was biased against a protected group, taking the model result is a binary output (0=bad, 1=good, or vice-versa), when the subject does not belong to the designated protected group, the mean is closer to 1 (or 0) by a substantial difference than those who belong to the protected group. If it wasn't biased, there should not be any substantial differences.

  After making the necessary calculations, the final result is shown as a printed trace alongside with some minor explanations to interpret the result.

D.4 Class aequitasTFG

This class implements the library Aequitas [11]. To create an object, the following parameters are needed:

- ***data***: A CSV file with the dataset used for the model containing the different attributes, the outcome (score) and the ground truth data (label).
- ***score***: A string with the name of the column that represents the results of the model.
- ***label***: A string with the name of the column of the dataset that represents the ground truth data.
- ***relations***: A dictionary including the groups to be audited with its possible values (e.g. relations = 'Race': ['Black', 'White', 'Other'], 'Sex': ['Male', 'Female']). This parameter is only needed if the dataset is not categorical (For more information refer to Appendix C).
- ***referenceGroups***: A dictionary stating the reference group to audit the bias against (e.g. referenceGroups = 'Race': 'Caucasian', 'Sex': 'Male').
- ***isTarget***: This parameter must be 0 or 1. It indicates if the outcome of the model that has a positive connotation is 0 or 1.
- ***isCategorized***: A boolean parameter indicating whether the dataset is categorized or not.

In the class, two types of methods can be distinguished:

- ***Data Processing Methods***: This methods are meant for processing the data and are only to be called by the initializing function of the class.
- ***Data Displaying Methods***: This methods have the purpose of showing the results either with tables or different plots.

D.4.1 Data Processing Methods

- ***__init__(self, data, score, label, relations, referenceGroups, isCategorized)***: Stores the variables and calls the necessary preprocessing functions.
- ***preprocessData(self)***: Changes the dataset column names so Aequitas can understand them, also, if the dataset is not categorized, calls other methods to do so.
- ***checkIfMissingAdd(self)***: See Appendix C.
- ***categorize(self)***: See Appendix C.
- ***deleteUnnecesary(self)***: Deletes all the columns that are not of value for Aequitas.
- ***aequitasPreprocess(self)***: Makes all the calculations that are necessary for the library to show the different tables and graphs.

D.4.2 Data Displaying Methods

The different metrics that Aequitas provides are explained on section 3.4.1, and depending on the method, they are to be used as an input parameter. Here is a list of all the different metrics and how to use them as input parameters:

- ***tnr***: True Negatives Rate
- ***fpr***: False Positives Rate
- ***fnr***: False Negatives Rate
- ***for***: False Omission Rate
- ***npv***: Negative Predicted Value
- ***fdr***: False Discovery Rate
- ***precision***: Model Precision
- ***ppr***: Predicted Positive Rate
- ***pprev***: Predicted Prevalence

Also, the variable *attribute* can be used as an input parameter in some methods, this refers to the an specific attribute of the population (e.g. Race, Sex...). The following data displaying methods have been created:

- ***getGroupsTable(self)***: Calculated absolute metrics table for all population groups.
- ***plotSingleGroupMetric(self, metric)***: Plots the specified group metric.
- ***plotAllGroupMetrics(self)***: Plots group graphs for all metrics.
- ***getBiasTable(self)***: Calculated disparity metrics table for all groups.
- ***plotSingleBiasMetric(self, attribute, metric)***: Plots the specified disparity metric for the specified population attribute.
- ***plotAllBiasOneAttributeMetric(self, attribute)***: Plots all the disparity metrics for the specified population attribute.
- ***plotAllBiasMetrics(self)***: Plots all the disparity metrics for all population groups.
- ***getFairnessTable(self)***: Calculated disparity and fairness metrics table for all population groups.
- ***getFairnessGroupTable(self)***: Calculated fairness metrics table for all population groups.
- ***plotSingleGroupFairnessMetric(self, metric)***: Plots group fairness metric for specified metric.
- ***plotAllGroupFairnessMetrics(self)***: Plots group fairness metrics for all attributes.
- ***plotSingleBiasFairnessMetric(self, attribute, metric)***: Plots bias fairness metric for a single metric and attribute.
- ***plotAllBiasFairnessOneAttributeMetric(self, attribute)***: Plots all bias fairness metrics for specified attribute
- ***plotAllBiasFairnessMetrics(self)***: Plots bias fairness metrics for all attributes and groups.

D.5 Class auditModel

This class makes use of the classes limeTFG and fairMLTFG. To create an object, the following parameters are needed:

- ***data***: Consists on a CSV file with the dataset used for the model.
- ***model***: Consists on a joblib export of the model.
- ***target***: Consists on a string that indicates the name of the column that stores the results of the model.
- ***targetNames***: Consists on a string array that indicates the meaning of the binary target values ([“meaning of 0”, “meaning of 1”]).
- ***title***: A string that indicates the name of the audited model.

The class auditModel has two methods:

- ***__init__(self, data, model, target, targetNames, title)***: With the given input parameters, creates an object limeTFG and an object fairMLTFG and stores them as inner class variables.
- ***showReport(self)***: Displays in the notebook cell the data generated by LIME (the explanation of an instance) and Fair ML (Table and graph of model feature dependence) using HTML displaying functions and calling the necessary functions of both classes.

D.6 Class auditDataset

This class makes use of the classes themisMLTFG amd aequitasTFG. To create an object, the following parameters are needed:

- ***data***: A CSV file with the dataset used for the model containing the different attributes, the outcome (score) and the ground truth data (label).
- ***target***: A string with the name of the column that stores the results of the model.
- ***label***: A string with the name of the column of the dataset that represents the ground truth data.
- ***relations***: A dictionary including the groups to be audited with its possible values (e.g. relations = 'Race': ['Black', 'White', 'Other'], 'Sex': ['Male', 'Female']). This parameter is only needed if the dataset is not categorical (For more information refer to Appendix C).
- ***referenceGroups***: A dictionary stating the reference group to audit the bias against (e.g. referenceGroups = 'Race': 'Caucasian', 'Sex': 'Male').
- ***isTarget***: This parameter must be 0 or 1. It indicates if the outcome of the model that has a positive connotation is 0 or 1.
- ***isCategorized***: A boolean parameter indicating whether the dataset is categorized or not.

The class auditDataset has three methods:

- ***__init__(self, data, target, label, relations, referenceGroups, isCategorized, isTarget)***: With the given input parameters, creates an object themisMLTFG and an object aequitasMLTFG and stores them as inner class variables.
- ***getProtectedGroups(self, relations, referenceGroups)***: Auxiliary method called by initializer function for latter proper display of results in case of the data not being categorized.
- ***showReport(self)***: Displays in the notebook cell the data generated by Themis-ML and Aequitas using HTML displaying functions and calling the necessary functions of both classes.

APPENDIX E

Acronyms and Abbreviations

AI: Artificial Intelligence

API: Application Programming Interface

COMPAS: Correctional Offender Management Profiling for Alternative Sanctions

GUI: Graphic User Interface

ML: Machine Learning

UML: Unified Modeling Langugage

Bibliography

[1] Edureka E-Learning Platform. What is Machine Learning? Machine Learning For Beginners. Online; accessed March 28, 2020 at https://www.edureka.co/blog/what-is-machine-learning/.

[2] Harini Suresh and John V. Guttag. A framework for understanding unintended consequences of machine learning. February 2020. Available at http://arxiv.org/abs/1901.10002/.

[3] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA, 23–24 Feb 2018. PMLR.

[4] Jeff Larsson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. May 2016. Available at https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

[5] Aequitas documentation. Online; accessed March 30, 2020 at https://dssg.github.io/aequitas/metrics.html.

[6] Fairml: Auditing black-box predictive models. Online; accessed April 2, 2020 at https://blog.fastforwardlabs.com/2017/03/09/fairml-auditing-black-box-predictive-models.html.

[7] marcotcr/lime: Lime: Explaining the predictions of any machine learning classifier. Online; accessed April 2, 2020 at https://github.com/marcotcr/lime.

[8] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Local interpretable model-agnostic explanations (lime): An introduction - o'reilly. 2016. Online; accessed May 23. Available at https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/.

[9] Lars Hulstaert. Understanding model predictions with lime - towards data science. 2018. Online; accessed May 23. Available at https://towardsdatascience.com/understanding-model-predictions-with-lime-a582fdff3a3b.

[10] Tom Mitchell. The discipline of machine learning. Technical Report CMU ML-06 108, 2006. Available at https://www.bibsonomy.org/bibtex/255bb706fdd667a0d0a76b78b46f6fabe/organikproject.

[11] Aequitas – center for data science and public policy. Online; accessed March 30, 2020 at http://www.datasciencepublicpolicy.org/projects/aequitas/.

[12] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. September 2019. Available at http://arxiv.org/abs/1908.09635v2/.

[13] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 3 edition, 2010.

[14] Numpy official website. Online; accesed April 24, 2020 at https://numpy.org/.

[15] Wes McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.* O'Reilly Media, 1 edition, February 2013.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] Matplotlib official site. Online; accessed March 30, 2020 at https://matplotlib.org/index.html.

[18] Anaconda — the world's most popular data science platform. Online; accessed April 23, 2020 at https://www.anaconda.com/distribution/.

[19] What is the jupyter notebook? Online; accessed March 30, 2020 at https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html.

[20] *Flask — The Pallets Projects*, 2020 (accessed May 5, 2020). https://palletsprojects.com/p/flask/.

[21] *Jinja — The Pallets Projects*, 2020 (accessed May 5, 2020). https://palletsprojects.com/p/jinja/.

[22] Colin McIntosh. *"Bias" - Cambridge Advanced Learner's Dictionary.* Cambridge University Press, 4th edition, 2013.

[23] Reuters. Amazon scraps secret ai recruiting tool that showed bias against women. Online; last accessed April 24, 2020 at https://reut.rs/2Od9fPr.

[24] The verge - twitter taught microsoft's ai chatbot to be a racist asshole in less than a day. Online; last accessed April 24, 2020 at https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist.

[25] Eugenie Jackson and Christina Mendoza. *Setting the Record Straight: What the COMPAS Core Risk and Need Assessment Is and Is Not - Harvard Data Science Review*, 2020 (accessed May 5, 2020). https://hdsr.mitpress.mit.edu/pub/hzwo7ax4/release/3.

[26] Jennifer L. Skeem and Christopher Lowenkamp. Risk, race, & recidivism: Predictive bias and disparate impact. June 2016. Available at https://ssrn.com/abstract=2687339.

[27] Mark Olver, Keira Stockdale, and J. Wormith. Thirty years of research on the level of service scales: A meta-analytic examination of predictive accuracy and sources of variability. *Psychological assessment*, 26, 11 2013.

[28] The allegheny family screening tool – allegheny county. Online; accessed May 15, 2020 at `https://www.alleghenycounty.us/Human-Services/News-Events/Accomplishments/Allegheny-Family-Screening-Tool.aspx`.

[29] Asft frequently-asked questions – allegheny county department of human services. Online; accessed May 15, 2020 at `https://www.alleghenycountyanalytics.us/wp-content/uploads/2019/05/FAQs-from-16-ACDHS-26_PredictiveRisk_Package_050119_FINAL-8.pdf`.

[30] Marc Böhlen, Varun Chandola, and Amol Salunkhe. Server, server in the cloud. who is the fairest in the crowd? November 2017. Available at `https://arxiv.org/abs/1711.08801`.

[31] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[32] Github - adebayo/fairml. Online; accessed April 2, 2020 at `https://github.com/adebayoj/fairml`.

[33] A fairness aware machine learning library - themis-ml 0.0.2. Online; accessed April 2, 2020 at `https://themis-ml.readthedocs.io/en/latest/index.html`.

[34] Russ Miles and Kim Hamilton. *Learning UML 2.0.* O'Reilly Media, Inc., 2006.

[35] P. Christensson. Gui definition. 2006. Retrieved 2020, Jun 9, from https://techterms.com.

[36] An introduction to seaborn. Online; accessed March 30, 2020 at `https://seaborn.pydata.org/introduction.html`.