UNIVERSIDAD
DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Máster Universitario en Ciencia de Datos e Ingeniería de Computadores

Master's Thesis

Development of Fair Machine Learning Algorithms based on Decision Trees

Presented by:
David Villar Martos

Supervising tutor:
Jorge Casillas Barranquero
Department of Computer Science and Artificial Intelligence

Academic year 2023-2024

Development of Fair Machine Learning Algorithms based on Decision Trees

David Villar Martos

David Villar Martos *Development of Fair Machine Learning Algorithms based on Decision Trees*. Master's Thesis. Academic year 2023-2024.

Supervising Tutor Jorge Casillas Barranquero
Department of Computer Science and Artificial Intelligence

Máster Universitario en Ciencia de Datos e Ingeniería de Computadores

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

University of Granada

Declaration of Originality

Mr. David Villar Martos

I explicitly declare that this work presented as the Master's Thesis (TFM), corresponding to the academic year 2023-2024, is original, understood in the sense that no sources have been used in the preparation of the work without proper citation.

In Granada, on June 30, 2024

Sgd. David Villar Martos

Para ti, mami.
Mi ejemplo de esfuerzo y superación.
Mi pilar y mi guía en este mundo.
Mi eterno recordatorio de que existe la bondad.

Contents

Acknowledgments

I would like to thank my family and my girlfriend for their unwavering support throughout this time. They know better than anyone that the period during which this work was carried out was a very tough time, with many ups and downs and problems, yet it has come through. They have supported me, been there for what I needed, given me motivation, and helped calm me down. Without them, I wouldn't be where I am today. You are part of me. I love you.

I would also like to thank my friends for the affection they have shown me throughout this time. They are special people who have helped me escape from problems and grow as a person. I will always be there for you, as you have always been there for me.

Last but not least, I would like to thank my tutor for the support received throughout this time. Despite the difficult times, he has always been there, and I have enormous affection and respect for him as a professional and as a person, and anyone who truly knows him will agree with me.

Abstract

Fairness in machine learning is a field that has gained great prominence and relevance in recent years. This area deals with the study and quantification of different measures of fairness on various specific problems and decision processes, as well as the development and implementation of solutions to create fairer systems. Unfortunately, at the limits of joint optimization between accuracy and fairness, a trade-off is reached, so that requiring a fairer system will necessarily imply less accuracy and vice versa. Due to this fact, multiobjective optimization emerges as a method that allows finding a wide range of solutions exploring this optimization frontier. Genetic algorithms represent the state of the art in these optimization processes.

This project will be developed within this area. After conducting a review of the context from a mathematical perspective, analyzing various interpretations of mathematical fairness, and exploring the theory behind multi-objective optimization, the creation of 3 novel algorithms based on decision trees for fair binary classification with a binary sensitive attribute will be proposed. The first algorithm, named Fair Decision Tree (FDT), modifies the information gain criterion during decision tree training to incorporate fairness. The second, named Fair Genetic Pruning (FGP), proposes a genetic optimization procedure on prunings in a matrix tree, which will be the tree that perfectly classifies the training sample. The last one, named Fair LightGBM (FLGBM), modifies the loss function of the LightGBM algorithm to incorporate fairness. For the two algorithms that do not inherently use genetic optimization processes, a genetic hyperparameter optimization procedure based on the NSGA-II algorithm will be employed. This will enable finding models on the joint optimization frontier, specifically for accuracy and fairness objectives. A study will be conducted to test these algorithms alongside a baseline algorithm (classic decision tree) on 10 relevant datasets within the field. Each dataset will be split into 10 training, validation, and test partitions using different random seeds. The average results obtained demonstrate how these algorithms are capable of finding a wide range of Pareto-optimal solutions. Overall, FDT and FLGBM algorithms outperform the baseline algorithm across several quality indicators calculated on the average Pareto fronts. The FGP algorithm has also shown promising results but has not surpassed the baseline algorithm on these indicators. However, it is the algorithm that achieved the best processing time results. All the algorithms have found models that perform better on both objectives than classic models like the COMPAS model developed by Northpointe.

The implementation has been made public (`https://github.com/Daalma7/FairTreesAlgorithms`) so that it can be employed for any problem, allowing specialists to select the model that best suits their requirements in terms of accuracy, fairness, or any other implemented objective.

Resumen

La justicia en el aprendizaje automático es un área que ha tenido un gran auge y relevancia en los últimos años. Este área trata sobre el estudio y cuantificación de diferentes medidas de justicia sobre diferentes problemas y procesos de decisión concretos, así como del desarrollo e implementación de soluciones para poder crear sistemas más justos. Desgraciadamente, en los límites de la optimización conjunta entre precisión y justicia se llega a un balance, de manera que si se quiere un sistema más justo obligatoriamente implicará que haya menos precisión y viceversa. Debido a este hecho, la optimización multiobjetivo surge como un método que permite encontrar una amplia gama de soluciones que exploren dicha frontera de optimización. Los algoritmos genéticos suponen el estado del arte en estos procesos de optimización.

Este proyecto se desarrollará en el marco de este área. Tras realizar una revisión del contexto desde una perspectiva matemática analizando distintos tipos de interpretaciones matemáticas de justicia y analizar la teoría detrás de la optimización multiobjetivo, se planteará la creación de 3 algoritmos novedosos, basados en árboles de decisión para clasificación binaria justa con un atributo binario sensible. El primero de ellos, llamado Fair Decision Tree (FDT) modifica el criterio de ganancia de información durante el entrenamiento del árbol de decisión para incorporar justicia. El segundo, llamado Fair Genetic Pruning (FGP) plantea un procedimiento de optimización genética sobre podas en un árbol matriz, el cual será el árbol que clasifique de manera perfecta la muestra de entrenamiento. El último de ellos, llamado Fair LightGBM (FLGBM) modifica la función de pérdida del algoritmo LightGBM para incorporar justicia. En los dos algoritmos que no utilizan de por sí procesos de optimización genética se utilizará un procedimiento genético de optimización de hiperparámetros multiobjetivo basado en el algoritmo NSGA-II. Esto permitirá encontrar los modelos en la frontera de la optimización conjunta, en este caso, de un objetivo de precisión y otro de justicia. Se realizará un estudio probando estos algoritmos junto a un algoritmo base (árbol de decisión clásico) sobre 10 conjuntos de datos relevantes dentro del área y usando 10 particiones para cada uno en entrenamiento, validación y test usando distintas semillas aleatorias. Los resultados medios obtenidos muestran cómo estos algoritmos son capaces de encontrar una amplia gama de soluciones Pareto-optimales. En general los algoritmos FDT y FLGBM superan al algoritmo base con respecto a una serie de indicadores de calidad calculados sobre los frentes Pareto promedio. El algoritmo FGP ha obtenido también buenos resultados, pero no ha conseguido superar al algoritmo base en estos indicadores. Sin embargo es el algoritmo que mejores resultados en tiempo de procesamiento ha obtenido. Todos los algoritmos han encontrado modelos mejores en relación a ambos objetivos que modelos clásicos como el modelo COMPAS de Northpointe.

La implementación se ha hecho pública (https://github.com/Daalma7/FairTreesAlgorithms) de cara a que estos algoritmos puedan ser empleados para cualquier problema, dejando al especialista la opción de seleccionar el modelo que más le convenga en cuanto a sus requisitos de precisión, justicia, o cualquier otro objetivo implementado.

Part I.

Aims of the project

In this part, the main initial goals of the work will be discussed.

1. Main goals of this project

Fairness in Machine Learning (ML) is gaining increasing relevance in the field of Artificial Intelligence, and it is critical to develop techniques that ensure a certain level of fairness in a mathematical sense in our constructed models to avoid emergent biases based on sensitive attributes, which may be potentially dangerous in this context. Among the techniques that exist within this field, those which modify the learning process of the base algorithm to incorporate fairness as a metric to optimize jointly with other classical precision metrics are a good approach to deal with these problems.

This project will consist of the development of three new techniques that incorporate fairness in our classifiers. Our base classifiers will be decision trees, and the modifications to those trees or the learning of them will be based on different principles (namely: modification of the information gain criterion when learning decision trees, genetic pruning of a large matrix tree, and modification of the loss function of a LightGBM model). They will be later applied to some datasets well-known in Fairness for Machine Learning literature.

The main goals of this work are as follows:

- **Review of Machine Learning**: Analysis of basic concepts of machine learning and decision tree models as binary classifiers.
- **Review of Fairness in Machine Learning**: Review of the area of Fairness in Machine Learning. Contextual introduction to the field, brief review of methods for incorporating fairness, analysis of different definitions and mathematical measures of fairness.
- **Review of Multiobjective Optimization**: Review of the multiobjective optimization area. Definitions, analysis quality indicators for solution sets.
- **Development of Algorithms**: Constructing classification techniques that incorporate mathematical fairness objectives using Decision Trees as base classifiers. This will involve the development of three new algorithms, based on different principles:
  - **Fair Decision Tree**: The first algorithm to develop will modify the impurity criterion to include fairness while learning a decision tree. This will involve the incorporation of a new hyperparameter, which controls the balance between the base impurity criterion and the fairness criterion, ensuring consistent splits with a measured balance between precision and fairness. Fairness metrics at node splitting need to be defined in order to calculate them accurately.
  - **Fair Genetic Pruning**: The second algorithm to develop will create the largest optimal possible tree to classify a certain dataset, which will perfectly fit the training data, and then prune it to optimize the objectives given by the user for validation data. Prunings need to be coded in such a way that a multiobjective evolutionary algorithm can be applied.

  - **Fair LightGBM**: The third algorithm to develop will adapt the loss function of a boosting algorithm, this time using LightGBM as it employs decision trees as base classifiers, to incorporate fairness. This will involve adding a hyperparameter in a similar manner to Fair Decision Trees. Additionally, it is necessary to adapt the fairness metrics and extend them continuously for use them in this model.

- **Multiobjective Optimization Framework**: Development of a multiobjective optimization framework to enable the use of these algorithms for this type of problem. Fair Genetic Pruning is inherently adapted for these problems, but for the other two, a hyperparameter optimization procedure needs to be implemented. The framework will be based on the NSGA-II algorithm.

- **Experimental setup**: Definition of an experimental setup, selection of well-known datasets in the context of Fairness in Machine Learning, and other specifications for testing the developed algorithms.

- **Analysis of results**: Analysis of results and graphics to understand and interpret the algorithm's performance, both individually and comparatively. Extraction of conclusions.

Part II.

Foundations: state of the art and theoretical framework.

In this part, the main foundations for the project will be presented. All the necessary concepts will be covered and studied, including the state of the art and related work, to provide a general context for the topic.

2. Background and motivations, state of the art

In this chapter, a general background is provided in the context of Fairness in Machine Learning. The motivations for the increasing importance of this topic in the world of Artificial Intelligence, as well as previous approaches to study and address the problem, will be analyzed.

2.1. Motivation

Machine Learning (ML) is widely used today in a wide range of fields such as medicine, industry, commerce... Despite its undeniable applicability in helping solve problems in multiple social and knowledge areas, it has well-documented negative qualities that have not yet been generally resolved.

One of them is that the decisions made by these systems can not only create inequalities among different population groups characterized by a common value in at least one attribute, in a sense that can be considered discriminatory, but can also enhance these differences over time [O'neil, 2016, Eubanks, 2018]. This fact has been acknowledged in numerous fields: human rights, privacy, employment, health... [pro, 2009, Angwin et al., 2016, Bolukbasi et al., 2016, Zehlike et al., 2017]. As a result, in recent years, there has been a growing awareness in a large part of the community to address this issue, motivating the study of the causes of these biased and discriminatory situations [Selbst et al., 2019, Binns et al., 2018], and the incorporation of techniques to ensure fairness in the decisions made by these methods [Zafar et al., 2017a, Zehlike and Castillo, 2020]. The focus of the study will be oriented towards the construction of one of these methods. It has been demonstrated that there is an inherent tradeoff between the quality of predictions and criteria for achieving fairness [Menon and Williamson, 2018], but, nevertheless, methods that consider definitions of fairness may be able to find solutions with comparable or better quality than current ones, also ensuring a less discriminatory decision [Valdivia et al., 2020].

To mathematically define what is considered to be fair [Verma and Rubin, 2018] is a non-trivial and necessary topic to address, in order to be able to develop methodologies which use this definition. Currently, it is known that there are multiple valid definitions, some of which belong to mutually exclusive families. This directly limits our options, and we may need to change our definition based on what we precisely aim to achieve.

For the implementation of these fairness measures in our models, there are three groups of techniques [Friedler et al., 2019] depending on when do they intervene: preprocessing, inprocessing, and postprocessing. Preprocessing techniques typically involve detecting the existence of bias or partiality in the data before applying them to the machine learning method, and manipulating the data to meet our fairness standards. Inprocessing techniques involve altering the machine learning procedure itself to ensure fair decision-making. They usually alter the learning method, which often occurs by imposing constraints to optimize

within the method, leading to fair solutions. Postprocessing techniques modify the results of an already trained classifier to meet specified definitions of fairness.

The project will focus on developing 3 innovative ideas within algorithm modification methods. These algorithms will be included in a multiobjective optimization procedure, aiming to explore the limits of optimizing conflicting objectives, specifically precision and fairness objectives. All these algorithms will use decision trees as the base machine learning model and they will employ genetic algorithms for the optimization process. The way these genetic algorithms are utilized will vary between algorithms; notably, two out of the three algorithms will utilize a genetic hyperparameter optimization procedure, while another will incorporate a genetic optimization process itself. The use of these algorithms is an excellent choice for conducting a study on multiobjective optimization context, even more when considering an increase in the dimensionality of the objective space (Many-objectives Optimization). The results obtained will be analyzed using different datasets widely employed in the context of fairness in machine learning. Moveover, different definitions of fairness which could be used, as well as measures of quality and performance of the methods, will be mathematically studied.

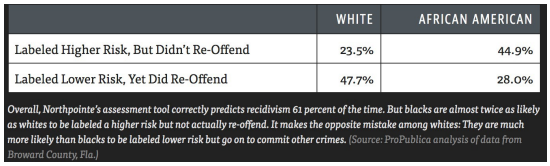
The perspective of this project establishes enormous potential applicability in all areas where to apply machine learning solutions. This is because discriminatory practices have been documented in almost all of these contexts. Furthermore, since optimization yields multiple solutions with different balances in the ranges of fairness, it offers the specialist the possibility to choose the classifier with the most suitable characteristics based on what they aim to achieve.

2.2. Fairness in machine learning background

As time goes by, fairness in machine learning is becoming an increasingly studied topic in the field of artificial intelligence. The issues related to fairness that machine learning could pose were known long before they gained the current significance. However, it was not until high-profile cases emerged that the community truly gave it the attention it deserves.

One of the cases that gained significant media impact was that of COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) in the USA. It was used to assess the risk of recidivism for individuals who had committed a crime, and served as a decision support system for determining variable sentencing periods for them.

In 2016, ProPublica published a data-driven analysis that had a significant impact. Its most striking finding was that COMPAS did not satisfy the false positive rate for both demographic subgroups, Caucasians and African Americans, being quite disparate. It turned out that among the inmates who did not reoffend, African American defendants were twice as likely to be classified as high risk. The study demonstrates how COMPAS has a much higher rate of false positives in the African American population compared to Caucasians (False Positives in Caucasians: 23.5%, in African Americans: 44.9%, False Negatives in Caucasians: 47.7%, and in African Americans: 28%). Based on this and similar findings, they labeled the tool as biased against the African American population.



	WHITE	AFRICAN AMERICAN
Labeled Higher Risk, But Didn't Re-Offend	23.5%	44.9%
Labeled Lower Risk, Yet Did Re-Offend	47.7%	28.0%

Overall, Northpointe's assessment tool correctly predicts recidivism 61 percent of the time. But blacks are almost twice as likely as whites to be labeled a higher risk but not actually re-offend. It makes the opposite mistake among whites: They are much more likely than blacks to be labeled lower risk but go on to commit other crimes. [Source: ProPublica analysis of data from Broward County, Fla.]

Figure 2.1.: Image extracted from ProPublica website, where they showed their results [Julia et al., 2016].

However, Northpointe, now named Equivant, the developer of COMPAS, criticized ProPublica's work and pointed out that COMPAS met the criterion that positive predictive value remained consistent across both demographic groups. This concept means the proportion of those classified as high risk who were subsequently arrested for recidivism. COMPAS also met other fairness measures, such as calibration. Based on this, along with similar findings, it drew significant attention from the community, prompting a thorough examination of the origins, causes, and measures to eliminate or mitigate potential biases and discrimination in machine learning processes [Julia et al., 2016].

The origins and causes of this, as well as other biases in the context of machine learning, are none other than human beings. Humans have prejudices and biases, which lead us to transmit these values to machine learning models. This is quite common to occur in the creation or selection of data for the training of our models, which can happen either directly or indirectly. Indirect bias may occur when sampling from the population or when selecting features to measure from individuals. On the other hand, direct sources of bias occur, for example, if data is labeled based on subjective criteria. The involvement of humans in the dataset creation process, even if they are experts, makes it highly improbable that these have been taken objectively and unbiasedly. Some of the main causes of bias are listed below:

- **Biased sample**: It occurs when samples are taken with partiality. This fact is highly problematic because bias can become even more pronounced over time. An example of this can be observed in police patrols, which tend to patrol more at conflictive zones, leading to obtain more conflict reports over them.
- **Tainted examples**: Since Machine Learning systems use a sample to learn models, there is a problem if chosen examples are mislabeled or biased in any way. For instance, if it were selected by someone with a strong political or racial bias, the system probably will learn that bias and incorporate it into its predictions.
- **Limited attributes**: It may happen that the attributes chosen in the study affect minority groups differently and are not as representative or have a different type of correlation.
- **Disparity in sample size**: Imbalanced issues tend to consistently disadvantage minority groups, and to address them, we can incorporate data imbalance treatment techniques.
- **Proxies**: It may happen that we have certain attributes which serve as proxies or intermediaries for others that could be more sensitive attributes, such as using address instead of nationality.

These causes can be summarized into 3 groups: uncovering initially unnoticed differences in the starting domain (biased sample and tainted examples), problems with the sample taken

while acknowledging differences in behavior (limited attributes and disparity in sample size), and understanding the causes of the disparities found (proxies). However, once bias has occurred, it becomes a really challenging task to discern its origin.

With the aim of avoiding the adverse effects of bias, examples of definition include the rules, norms, and laws against discrimination in some countries. We can see that these definitions essentially fall into two major groups:

- **Disparate treatment**: If the decision-making process is entirely or highly based on the value of a sensitive attribute that is susceptible to being the source of discriminatory behaviors.
- **Disparate impact**: If the consequences based on the decisions made directly harm individuals with a specific value in a sensitive attribute that is susceptible to being the source of discrimination.

2.3. Background

Concerning the establishment of fairness criteria in machine learning algorithms, there has been a clear trend to introduce it, directly or indirectly, through constraints which imply fairness. In this way, joint optimizations have been performed based on both accuracy and fairness criteria [Zafar et al., 2017b].

With respect to classifiers, particularly decision trees, efforts have been made to study ways of altering their training processes to facilitate the incorporation of fairness measures. In [Kamiran et al., 2010], the information gain function is slightly modified to perform operations on the tree that aim to minimize entropy concerning the attribute that may cause discrimination. Various alternatives were explored, such as entropy concerning the class label.

In [Agarwal et al., 2018], it was proposed to reinterpret the problem of fair classification as a series of cost-sensitive classification problems, with the aim of achieving Pareto-optimal solutions with respect to both prediction accuracy and the considered definition of fairness.

In [Balashankar et al., 2019], a Pareto-optimal point is identified, which improves accuracy and also optimizes various precision measures on subgroups. Additionally, the concept of equality of opportunity is satisfied.

In [Zafar et al., 2019], a convex problem with fairness constraints is considered, where the method's accuracy is optimized based on these constraints. During the problem formulation fairness is introduced in terms of a measure defined with respect to fairness at the decision boundary. This measure inherently encapsulates other fairness measures. Disparate treatment leads to a tradeoff between fairness and accuracy in the model, and this tradeoff is determined by a parameter specified by the user based on their preference. The problem formulation also allows adding certain attributes as constraints to the problem.

In [Hu and Chen, 2020], the classic minimization problem with constraints is transformed into a social welfare maximization problem. Support vector machine regularization tech-

niques and other methods were employed, leading to the conclusion that, when applying strict fairness criteria, it may be possible to achieve worse overall welfare for the study groups.

The work proposed in [Valdivia et al., 2020] does address, for the first time, the use of multi-objective optimization using a modification of NSGA-II, and it serves as a base building block for this project. Unlike the majority of literature in this field, this work employs the fairness criterion not as a constraint but as an objective to be minimized. Two objectives are used, one for error through geometric mean and another for fairness in terms of the difference in the false positive rate, a criterion falling within the branch of predictive parity fairness. Complexity is considered only as an auxiliary criterion to establish dominance among individuals. The conclusion of the work is that using this method can lead to better solutions that surpass not only in fairness but also in accuracy, some machine learning methods used in the problems of the datasets employed.

With respect to recent algorithms that incorporate fairness and are based on decision trees: in [Castelnovo, 2022], decision trees are developed where the impurity criterion only considers making splits at a node if that split meets a specific global fairness constraint.

In [Zhang and Zhao, 2020], an algorithm also based on decision trees is proposed, where the impurity criterion is multiplied by the result of a fairness function in certain cases. This algorithm is applied to a data stream mining problem, achieving good results.

In [Ranzato et al., 2021], the Meta-Silvae genetic algorithm is used to optimize decision trees with respect to a convex combination of accuracy and a measure of individual fairness. This genetic algorithm represents each tree with its own structure and applies crossover and mutation to those trees. Crossover is performed substituting one subtree of one parent with one of the other parent, while mutation is done using combinations of growth and pruning.

In [van der Linden et al., 2022], a pruning criterion for decision tree branches is also proposed, where no improvement in terms of fairness is expected. This approach works with the independent problems that remain in the decision tree as it is traversed, using global fairness constraints.

In [Cruz et al., 2023], a further step is taken by including a fairness optimization criterion in a LightGBM model, reformulating fairness objectives so that they can be included in the Lagrangians of the method itself.

Therefore, during this time, the community has shown an interest in conducting simultaneous optimizations in the context of decision quality and fairness in machine learning methods. The project's work takes a novel approach by attempting to explore the entire Pareto front in a multiobjective optimization, which can discern a wide range of solutions, all with optimal characteristics in the Pareto sense. For the first time, a method will be designed capable of efficiently optimizing various criteria of fairness, precision, and complexity simultaneously, allowing not only to obtain a large number of alternative solutions with different fairness profiles but also to understand the reachable non-dominance boundaries in the dataset, thus characterizing the problem in terms of fairness.

Multiobjective problems are very common in the current world, because we often encounter

contradictory objectives that we want to optimize simultaneously. A typical example would be an optimization problem in which we want to maximize the quality in the production of a product and minimize the investment we make in its production. Usually, we work with two objectives. When the number of conflicting objectives increases, it is already referred to as Many-objectives optimization, which, due to its characteristics, requires special techniques for its resolution.

Currently, there is a wide range of multiobjective optimization paradigms [Li et al., 2015], based on various criteria such as relaxing the concept of Pareto dominance, restricting population diversity at specific points, basing the search process on different quality measures, niche-based algorithms, and more. Each algorithm has its own distinct properties regarding structure, diversity, and convergence of solutions, which can be studied in a subsequent analysis. Due to the diverse perspectives, implementing multiple algorithms, combining their solutions, and studying them comparatively will allow us to achieve better reliability when extracting results.

With respect to fairness metrics, they are currently handled in three major groups: group fairness, individual fairness and counterfactual fairness. It has been demonstrated that some definitions of fairness belonging to certain subfamilies are incompatible [Chouldechova, 2017], making the use of multiobjective optimization techniques a justified approach to work in this context.

3. Machine learning and decision trees introduction

In this chapter, a brief introduction to machine learning concepts will be provided. Properties of decision trees that will be used later will be described.

3.1. Machine learning, context and basic concepts

The objective of this project is to implement three new binary classification algorithms based on decision trees that incorporate fairness measures. Subsequently, a multiobjective optimization process will be performed, either on their hyperparameters or within the algorithm itself in order to find the best models with respect to precision and fairness objectives. Before discussing different definitions of fairness in machine learning or the concepts related to multiobjective optimization, which will be covered in Chapters 4 and 2 respectively, we will review theoretical concepts on machine learning, classification problems, and decision trees from a mathematical perspective. This will help anyone unfamiliar to the topic to better understand the basic concepts. A standard notation will be introduced, and we will explore some of the key concepts that will enable us find good classifiers [Abu-Mostafa et al., 2012]. This notation will be further expanded in Sections 3.3 and 4.1.

Having considered a population of individuals, represented by a set of characteristics $(X = \{X_1, \dots X_n\}, Y)$, also known as "attributes" or "variables", the objective of machine learning is to learn a function $f\colon X \to Y$ which maps values from the set X (also known as "explanatory variables", "independent variables" or "predictors") to the set Y (also known as "response variable", "dependent variable" or "target variable"). For instance, given a population of students, a machine learning problem could be to learn the function f which relates the final grades they obtained in physics, chemistry, and literature $(X = \{X_1, X_2, X_3\})$ to their final grade in maths Y. We will know the actual X and Y attributes of some of these individuals, and this is known as "supervised machine learning". We will be dealing with binary classification problems, which means that Y can only take on the values 0 or 1.

The function f is unknown, so our goal is to find the best possible approximation to it based on the data we have. We will search for this approximation, denoted as $\hat{p}$, within a specific class of functions denoted as $\mathcal{H}$ (we will use $\hat{p}$ instead of using the notation $\hat{h}$ that appears in [Abu-Mostafa et al., 2012] to maintain a consistent notation throughout this work). This class of functions may not necessarily guarantee that it contains the true function f, but it is necessary to use it to make the search process manageable.

To find the function $\hat{p} \in \mathcal{H}$ such that we can approximate f as closely as possible, we need to define an error function that indicates the quality of our approximation $\hat{p}$ based on the errors it makes. Since we are in a classification context, the error we will use to measure the quality of our model is known as "classification error". However, for learning the function p and evaluating its error, we only have a sample from the total population, denoted as $\mathcal{D}$. The error of the function p can be calculated within the sample $\mathcal{D}$ (E_{in}), although what we aim

to minimize is the error for the entire population (E_{out}). Therefore, we need to find methods that ensure we can bound the error E_{out} based on the error E_{in}. Thus, our objective is to find :

$$\hat{p} \in \mathcal{H} : E_{out}(\hat{p}) \leq E_{out}(p), \forall p \in \mathcal{H} \tag{3.1}$$

Where the error we will use is the classification error. Its expression for E_{in}, which can be calculated for a specific sample $\mathcal{D}$, and E_{out} are be as follows:

$$\begin{aligned} E_{in}(p) &= \frac{1}{N}\sum_{i=0}^{N}[y_i \neq p(x_i)] \\ E_{out}(p) &= P_x[p(x) \neq y] \end{aligned} \tag{3.2}$$

To achieve this, we will introduce a series of properties that will allow us to approximate this better function based on E_{in}.

Lemma 3.1.0.1 (Hoeffding inequality for a finite $\mathcal{H}$)**.** *Given a binary classification problem, where we use classification error and a finite $\mathcal{H}$, with a sample $\mathcal{D}$, then $\forall \epsilon \in (0, 1]$:*

$$P_{h\in\mathcal{H}}[|E_{in}(p) - E_{out}(p)| > \epsilon] \leq 2|\mathcal{H}|e^{-2\epsilon^2 N} \tag{3.3}$$

where ϵ is a constant fixed by the user, and the value of N is the the number of individuals in the training sample $\mathcal{D}$.

This bound is not excessively tight, but it provides us with a means to control the value of E_{out} in terms of E_{in} with a certain probability. This inequality can be equivalently expressed as:

$$E_{out}(p) \leq E_{in}(p) + \sqrt{\frac{1}{2N}\log\left(\frac{2|\mathcal{H}|}{\delta}\right)}, \text{ with at least probability } 1-\delta. \tag{3.4}$$

In the case where the function class $\mathcal{H}$ to be used is not finite, we cannot apply this inequality. To remedy this, we can take one of two different approaches:

- **Generalized Vapnik-Chervonenkis theory**: Use the generalization theory of Vapnik-Chervonenkis. A bound similar to that of the Hoeffding inequality can be established using the Vapnik-Chervonenkis dimension. However there are some classes for which this dimension is infinite, and other methods must be employed, such as algorithms based on Structural Risk Minimization.
- **Discrete class of functions**: Attempt to discretize the class of functions, taking advantage of discrete numerical representations that computers use. This could be useful in some practical contexts, but in general, these methods often result in worse bounds compared to the previous approach.

Let us briefly introduce the theory of Vapnik-Chervonenkis.

Definition 3.1.1 (Growth function)**.** Let $(x_1, \ldots, x_n)$ be a sample of size n, and let $\mathcal{H}$ be a class of functions. Then, the growth function associated with that class of functions $\mathcal{H}$ is a function that counts how many different binary classifiers can generate a function from $\mathcal{H}$, denoted as:

$$m_{\mathcal{H}}(n) = \max_{x_1,\ldots x_n} |\mathcal{H}(x_1, \ldots, x_n)| \tag{3.5}$$

It is clear that $m_{\mathcal{H}} \leq 2^n, \forall \mathcal{H}$. Moreover, the fact that $\exists n \in \mathbb{N} : m_{\mathcal{H}}(n) < 2^n$ will play a big role in error bounding.

Definition 3.1.2 (Break point)**.** A break point for a class of functions $\mathcal{H}$ is a value $k \in \mathbb{N} : m_{\mathcal{H}}(k) < 2^k$.

As an example of function classes $\mathcal{H}$ that have breakpoints, we can mention the class of linear functions ($k = 4$), the class of positive rays on a line ($k = 2$), and the class of intervals on a line ($k = 3$). A class that does not have breakpoints, for example, could be the class of convex polygons in $\mathbb{R}^2$.

Lemma 3.1.0.2 (Break point extension)**.** *If a class of functions $\mathcal{H}$ has a break point k, then $\forall k' > k, m_{\mathcal{H}}(k') < 2^{k'}$.*

Proof. If k is a breakpoint, it means that there does not exist any set of size k that we can binary classify in all possible ways using functions from the class $\mathcal{H}$. By contradiction, if there were a point $k' > k$ such that $m_{\mathcal{H}}(k') < 2^{k'}$, it would imply that there exists a sample of size k' to which we can assign a binary classification in all possible ways using functions from our class $\mathcal{H}$. Taking a subset of this sample of size k, we can observe that we would be able to do the same for this subset as well, contradicting our hypothesis. □

Lemma 3.1.0.3 (Main result of Vapnik-Chervonenkis)**.** *If k is a break point of a class of functions $\mathcal{H}$, then $m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}, \forall N \in \mathbb{N}$, thus $m_{\mathcal{H}} \in \mathcal{O}(N^{k-1})$.*

The proof can be consulted in [Vapnik, 1971]. Therefore, if a function class has a breakpoint, it necessarily implies that its growth function has polynomial growth. This fact allows us to establish a bound in the following manner:

Definition 3.1.3 (Vapnik-Chervonenkis dimension)**.** Vapnik-Chervonenkis dimension, denoted as $d_{vc}(\mathcal{H})$, or simply d_{vc}, is defined as the greatest value of N such that $m_{\mathcal{H}} = 2^N$. If this value does not exist, then $d_{vc}(\mathcal{H}) = \infty$.

Lemma 3.1.0.4 (Vapnik-Chervonenkis bound)**.** *Let $\mathcal{H}$ be a class of functions such that $d_{vc} < \infty$. Considering a classifier $p \in \mathcal{H}$ we can establish the following bound for E_out given E_in:*

$$E_{out}(p) \leq E_{in}(p) + \sqrt{\frac{8}{N} \log\left(\frac{4((2N)^{d_{vc}} + 1}{\delta}\right)}, \text{ with at least probability } 1 - \delta. \tag{3.6}$$

In cases where the Vapnik-Chervonenkis dimension is infinite, we cannot establish a bound using this theory alone. We can resort to the discretization trick, but the bound obtained may be quite imprecise. Therefore, it is often better to make bounds using other procedures.

3.1.1. E_{out} approximations using test sets

The methods discussed earlier provide bounds on the value of E_{out}, but these bounds are often not of high quality. A more effective way to calculate the value of E_{out} involves using a test set to evaluate the quality of our model. This test dataset represents a new sample drawn from the population, independently and identically distributed with respect to the one previously drawn for training. Typically, and as will be the case in our experimentation, we do not have two distinct datasets for training and testing. Therefore, we will split the dataset into two, using one part for training and the other for testing. The error obtained on

the test set will be denoted as E_{test}.

Using a test set provides a significant advantage, as now we can apply the Hoeffding inequality where the class $\mathcal{H}$ consists of a single function, which is the one obtained after training. With this, the following expression is satisfied:

$$P[|E_{test}(p) - E_{out}(p)| > \epsilon] \leq 2e^{-2\epsilon^2 N_t} \tag{3.7}$$

Where in this case, N_t is the number of instances in the test set. This represents a much more accurate bound than the one used when considering only the training set. That is the reason why using a test set to evaluate our models will be practically indispensable. This approximation will be employed to evaluate the models for the experimentation conducted, which will be discussed in Section 9.5.

Keep in mind that allocating more data to training makes our test approximation less precise, while dedicating more data to testing means having fewer data for training, potentially resulting in poorer classifiers. It is important to note that the test dataset will be completely separate and only used to evaluate the models as the final step in our procedures. Processes such as data normalization, if performed, should be done separately for both sets to avoid the phenomenon known as "data snooping", where information from the test set is directly or indirectly used for training, making the procedure theoretically invalid.

3.1.2. Bias-variance tradeoff

The error made by our model can be decomposed into the sum of three distinct components, namely:

- **Bias**: One component is called bias, which indicates how far the average prediction made by the function class is from the real function to be approximated.
- **Variance**: Another component the variance, which indicates the internal variability of the function class with respect to the class mean, and it depends on the size of the taken sample.
- **Random noise**: The final component represents the variance of the noise, which is a component that cannot be controlled.

We can express this tradeoff as: $\mathbb{E}_{\mathcal{D}}[E_{out}(h^{(\mathbb{D})})] = \text{bias} + \text{variance} + \sigma^2$.

This concept of bias-variance tradeoff was initially developed for regression, but it has been shown to hold true for classification contexts as well, as demonstrated in [Domingos, 2000].

3.1.3. Overfitting and regularization

A common phenomenon that can occur during the training of our models is that known as "overfitting". Overfitting happens when models with lower values of E_{in} result in a higher error E_{out}. This phenomenon may be due to stochastic errors, where there might be noisy examples, and to adapt to them, the classifier needs significant modifications that hinder its generalization to examples outside the sample. It can also be attributed to deterministic

errors, as the trained model may not accurately represent the real function f.

To address this issue, regularization is proposed as a technique that limits the function class through soft constraints, making it challenging to train excessively complex models that are prone to overfitting. Common regularization techniques include weight decay for linear models or parameter growth limitations for decision trees, among others.

3.1.4. Validation sets

The validation set involves a similar process to that used with test sets. In this case, the training set is divided into two subsets: the actual training set and the validation set. Models are trained with the pure training set, and after that, their performance is evaluated based on the results when applied to the validation set. This allows us to evaluate the behavior of the models against independently and identically distributed data as those used for training.

The use of a validation set is useful when we want to select a better hypothesis or set of hypotheses during the training of the models. It is possible that our learning models have a series of parameters (known as "hyperparameters") that we need to estimate to build the best possible model. In order to do so, several models with different hyperparameter configurations will be trained, and then evaluate them using the validation set. For our study we will use a genetic hyperparameter optimization process, and to select the best configurations among those tested, we will evaluate the trained algorithms using the validation set.

Currently, sophisticated validation techniques such as Leave-One-Out or K-fold cross-validation are employed. However, for our experimentation, we will use a simpler approach, which involves setting aside a subset of the data explicitly dedicated to validation, similar to what was done with the test set. This chosen validation technique will be explained in Section 9.5.

3.2. Decision trees

In this case, given our context, each individual managed by our genetic algorithms represents a classification function. We need to adjust this function using a machine learning model and subsequently evaluate the quality of its predictions. To demonstrate the potential that can be achieved through the proposed procedure and to ensure that the total execution time of our algorithm is not excessively high due to the significant time invested in training each individual, we will use relatively simple function classes $\mathcal{H}$. These classes should have good properties while maintaining training methods computationally inexpensive. We will use decision trees, which basic properties will now be discussed.

3.2.1. Decision trees as a class of functions

Decision tree classifiers define a class of functions widely use in machine learning. It is a basic class of function to be studied in any basic machine learning course and can be applied to a wide range of problems, such as classification or regression problems, since it is a simple yet powerful class that can greatly adapt to the training data. This adaptability is mainly due to the fact that they can find the combination of features and values that best split the data into similar subgroups. Furthermore, their learning algorithms, such as CART or C4.5 are

relatively simple, intuitive, and fast.

Decision trees classifiers partition the space into different regions, whose boundaries are parallel to the coordinate axes of each variable. Interpreting all our variables as real variables without loss of generality, we can represent the class of classification functions described by decision trees as follows:

$$\mathcal{H}_{\mathcal{DT}} = \left\{ \sum_{i=0}^{N} \alpha_i \mathbb{1}_{\mathcal{X}_i}(x) : N \in \mathbb{N}, \alpha_i \in \mathbb{R}, \mathcal{X}_i \text{ meet the following properties} \right\} \tag{3.8}$$

- **Half-closed intervals**: Each $\mathcal{X}_i$ is a cartesian product of half-closed intervals, not necessarily bounded.
- **Partition**: $\{\mathcal{X}_i : i \in \{1, \ldots, N\}\}$ conform a partition of X, which means:
  - $\cup_{i \in \{1,\ldots,N\}} \mathcal{X}_i = \mathbb{R}^N$
  - $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset, \forall i, j \in \{1, \ldots, N\}, i \neq j$

To perform this partition, trees are structured as a set of hierarchical rules, effectively forming a tree structure, where each non-terminal node represents a rule, and each edge represents a possible outcome for that rule. We will work with binary trees, which means that each rule node has exactly two children. The rules used are inequality comparison rules with respect to a single variable, ensuring that the structure of decision boundaries is parallel to the axes. Thus, for each new example to classify, decision rules are applied consecutively based on the results of previous rule evaluations. The leaf nodes represent the class in which to classify the given example.

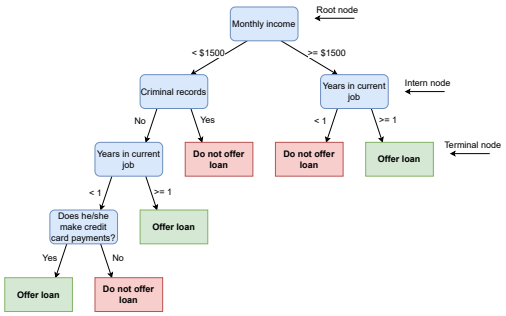


Figure 3.1.: An example of a binary decision tree used for binary classification. The terminal nodes represent the classification made, the non-terminal nodes represent the decision rules to apply, and each edge is directed from nodes of lower to higher depth and represents a possible response to the decision rule. This is a binary tree because each non-terminal node has exactly two children nodes.

3.2.2. CART learning algorithm

The learning algorithm that we will use to fit the best function from the mentioned class $\mathcal{H}_{DT}$ to our training sample is known as "CART algorithm" (Classification And Regression

Trees). The pseudocode of this algorithm can be seen in Algorithm 1.

Algorithm 1: Basic CART algorithm for decision tree classifiers.

Input: Impurity criterion g. Stop criterion.
Data: Training dataset (D_l).
Output: Decision tree classifier trained using D_l.

```
Initialize the tree with a single node that contains all the training data D_l.
while Stop criterion is not met. do
    if All instances that fall onto the current node have the same class, or some stopping
     criterion is met. then
        Convert the current node into a leaf node that assigns the class to all instances,
         and we finish.
    else
        Calculate the attribute and feature that best separates the data falling onto the
         current node based on the node's impurity measure.
        Assign to this node the associated decision rule.
        A cost-complexity pruning process is carried out for the possible child nodes:
         g(n) + α|N| where α is a free parameter subject to optimization, and |N| is the
         number of nodes created up to that point. We create two child nodes if they
         have not been pruned.
return Trained decision tree
```

To discuss the efficiency of the algorithm, we need to introduce the $\mathcal{O}$ notation. Let us provide a brief definition of notation $\mathcal{O}$ and mention some of its most interesting properties.

Definition 3.2.1 ($\mathcal{O}$ notation)**.** The class of functions $\mathcal{O}(g)$, being g a function, is defined as:

$$\mathcal{O}(g) = \{f : \exists x_0, c > 0 : 0 \leq |f(x)| \leq c|g(x)|, \forall x \geq x_0 > 0\} \tag{3.9}$$

Despite $\mathcal{O}(g)$ being a set, to express that a function f belongs to $\mathcal{O}(g)$, it is denoted as $f = \mathcal{O}(g)$ instead of $f \in \mathcal{O}(g)$.

A function $f = \mathcal{O}(g)$ if it growth is bounded by function g (except for a constant). Let us now write some simple properties which directly derive from the definition of $\mathcal{O}$ notation:

- **Transitivity**: If $f_1 = \mathcal{O}(g_1)$ and $g_1 = \mathcal{O}(g_2)$, then $f_1 = \mathcal{O}(g_2)$.
- **Product of functions**: If $f_1 = \mathcal{O}(g_1)$ and $f_2 = \mathcal{O}(g_2)$, then $f_1 f_2 = \mathcal{O}(g_1 g_2)$.
- **Product of a class by a function**: $f_2\mathcal{O}(g_1) = \mathcal{O}(f_2 g_1)$.
- **Sum of functions**: If $f_1 = \mathcal{O}(g_1)$ and $f_2 = \mathcal{O}(g_2)$, then $f_1 + f_2 = \mathcal{O}(|g_1| + |g_2|)$.
- **Scalar product of a class**: If $k \neq 0$, then $\mathcal{O}(g_1) = \mathcal{O}(k g_1)$.
- **Scalar product of a function**: If $f_1 = \mathcal{O}(g_1)$, then $k f_1 = \mathcal{O}(g_1)$.
- **Class of a sum of functions of the same order**: If $f_1 = \mathcal{O}(g_1 + g_2)$ and $g_1 = \mathcal{O}(g_2)$, then $f_1 = \mathcal{O}(g_2)$.

3.2.3. Impurity metrics

The choice of the best attribute and node is made through a greedy heuristic procedure (best-first). Since we will work with binary trees, all possible space partitions of X into regions that effectively separate the data in different ways using one value of only one variable at a time will be evaluated. The chosen partition is the one that yields the lowest result with respect to the children's impurity function, maximizing impurity reduction or information gain. With m being the number of attributes and n the number of instances, the time complexity of the algorithm is $\mathcal{O}(mn\log(n))$, as the evaluation of a possible partition can be done in logarithmic time with respect to the number of instances [Sani et al., 2018].

There are several options for measuring the impurity of a partition made by a node. The two most common ones that we will consider are:

- **Gini index**: For leaf nodes, its value is $1 - \sum_{i=1}^{n} p_i^2$, where n is the number of possible clases (different values that Y can take), and p_i is the proportion of elements from class i which fall into that node. For the rest of nodes its value is calculated using the value $1 - \sum_{i=1}^{n} p_i^2$ for each one of its children nodes (being them leaves or not), weighted using the amount of samples which fall into each children node. Possible values for this metric are in the range $[0, 0.5]$.
- **Entropy**: Entropy calculation is done following the same pattern as for Gini index, but using the expression $-\sum_{i=1}^{n} -p_i \log_2(p_i)$ instead. Possible values for this metric are in the range $[0, 1]$.

For both metrics, the closer they are to 0, the less variability there is in elements that fall into that node with respect to their class. A comparison between both Gini and entropy functions is shown in Figure 3.2.

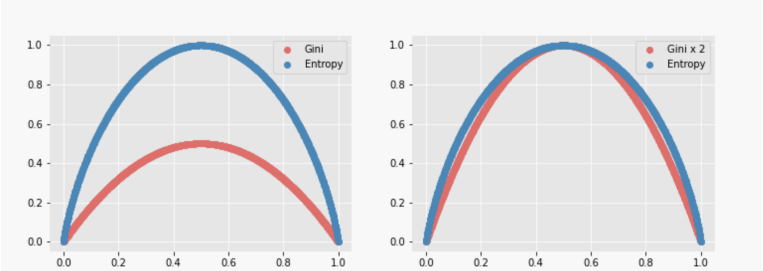


Figure 3.2.: Relation between Gini and entropy impurity functions. Both functions are very similar, and could result in minor differences during the learning process.

3.2.4. Advantages and disadvantages of decision trees

As advantages we can highlight the following:

- **Independent to scale**: The method is invariant to the scale of the data.
- **Data types**: Decision trees can handle both categorical and real-valued data.

- **Explainability**: Decision trees have high explainability, allowing us to understand the process of classifying any individual, as well as the most important variables for splitting the data.
- **Representation**: They have a direct and interpretable representation through their associated set of decision rules.
- **Prediction**: Once learned, prediction using a decision tree is quick as it involves applying the associated set of rules.
- **Missing values**: Decision trees can handle missing values, as they do splits based on the available data.

On the contrary, as disadvantages we can highlight the following:

- **Overfitting**: Decision trees tend to overfit the training sample if their growth is not controlled using hyperparameters.
- **Vapnik-Chervonenkis dimension**: $d_{vc}(\mathcal{H}_{DT}) = \infty$
- **Variance**: The decision tree class has high variance. Given small variations in the training data, the resulting classification tree can vary significantly. To address this issue, the Random Forest model emerged, although it sacrifices the explainability of the model.

3.2.5. Additional considerations: Regularization

Decision trees as a class of functions have $d_{vc}(\mathcal{H}_{DT}) = \infty$. This happens because a decision tree could grow as much as necessary, generating as many rules as needed to perfectly classify any sample. This could result in each individual falling into a different node, providing a tree with a total of n nodes, where n is the number of instances in the training sample. To introduce regularization into the model, one can limit the growth of the tree by restricting the number of leaf nodes it can have, or requiring a certain number of individuals to fall into a node before it can split.

3.3. Additional notation

We will only work with binary classification problems. We will denote p as a binary predictor function, $p\colon X \to Y$, and as Y can only have two values or classes, without loss of generality, we can establish them to be 0 and 1. Class 1 will be denoted as the positive class and class 0 as the negative class. For each individual (x, y) we will consider the abuse of notation $p = 1$ if $p(x) = 1$, and $p = 0$ if $p(x) = 0$. Doing so, we can define the following sets:

- **Positive class (P)**: Individuals for whom $Y = 1$.
- **Negative class (N)**: Individuals for whom $Y = 0$.
- **Predicted Positive (PP)**: Individuals for whom $p = 1$.
- **Predicted Negative (PN)**: Individuals for whom $p = 0$.
- **True Positives (TP)**: Individuals for whom $Y = 1$ and $p = 1$.

- **True Negatives (TN)**: Individuals for whom $Y = 0$ and $p = 0$.
- **False Positives (FP)**: Individuals for whom $Y = 0$ and $p = 1$.
- **False Negatives (FN)**: Individuals for whom $Y = 1$ and $p = 0$.

4. Fairness in machine learning

In this chapter, the main mathematical definitions of fairness in machine learning will be discussed. Certain properties of them will be presented and analyzed. This will help us gain a better and deeper understanding of the topic, as well as establish a notation to follow for further results.

4.1. Mathematical definitions of fairness

It is evident, as shown in Chapter 2, that there is a need to incorporate methods that ensure fair decisions will be made in the context of each specific problem we try to solve. However, there is no defined consensus on what is considered fair within a given context, and multiple conceptions of fairness have emerged from different, equally valid perspectives. The crucial question in our field is how can we define the concept of fairness in a way which can be integrated in our machine learning systems.

There are numerous definitions of fairness in the literature. These can be classified into three major groups, which are not mutually exclusive, but have different approaches in terms of their conception. These three major groups are group fairness, individual fairness, and counterfactual fairness. In order to define each of these families more rigorously, we will introduce a notation.

Extending the notation in Section 3.1, let us consider a probability space $(\Omega, \mathcal{A}, P)$, associated with an observational process of a group of individuals with respect to a certain context. Over this probability space we will define a set of random variables: $X = \{X_1, \cdots, X_n\}, A, Y$. Each of these $X_i \colon (\Omega, \mathcal{A}, P) \to (\mathbb{R}, \mathcal{B}, P_{X_i}), \forall i \in \{1, \ldots, n\}$ are random variables which represent specific observed characteristics of our individuals.

The random variable $A \colon (\Omega, \mathcal{A}, P) \to (\mathbb{R}, \mathcal{B}, P_A)$ represents the value an individual takes on a sensitive attribute, i.e., one that may lead to discrimination. This is also known as a "protected attribute". In this study, A will be a one-dimensional binary random variable taking values 1 and 0, but there could be more than one sensitive attribute among those studied. Social groups will be considered as collections of individuals who share the same value in the random variable A, thus describing two exhaustive and mutually exclusive social groups, known as "protected groups" or "demographic groups". Typically, in cases where two groups are considered, one of them is known as the "privileged group" (individuals for whom $A = 1$), and the other one is known as "unprivileged group" (individuals for whom $A = 0$). It is important to note that the sensitivity of this attribute is context-dependent: there may be attributes considered sensitive in one context but not in others. For example, significant differences in the number of granted mortgage loans between individuals of different biological sexes may indicate discrimination, whereas when assessing an individual's suitability for medical treatment, biological sex may not be a sensitive attribute due to relevant

physiological differences.

Finally, $Y\colon (\Omega, \mathcal{A}, P) \to (\mathbb{R}, \mathcal{B}, P_Y)$ is a random variable representing a characteristic of interest. For the current study case, we will also consider that the random variable Y is binary, $Y \in 0, 1$, thus discrete. Therefore, our study will focus on supervised binary classification problems.

Our objective is to find a predictor variable $p\colon (\Omega, \mathcal{A}, P) \to (\mathbb{R}, \mathcal{B}, P_Y)$, that can approximate the true relationship between variables X, A and variable Y as accurately as possible. This variable can also be described as a predictor function $p\colon X, A \to Y$. Depending on whether we consider the probabilistic or functional notation, we will write $p = y$ or $p(x_1, \dots x_n, a) = y$ ($p(X, A) = Y$), although it is common to abuse the notation and directly write $p = y$. In this chapter we will mostly use the probabilistic context. This variable p will also be binary, taking the same values as Y. In the current study case these will be 0 and 1.

Let us assume that all variables follow a joint distribution $(X, A, Y) \sim D$. However, in most cases, we may not be able to sample data from that exact distribution; instead, the sampled examples will be biased. This is why, in a real-world scenario, we will sample from a biased distribution D', different from the theoretical true distribution D.

If we did not consider anything about fairness and wanted to obtain a perfect classifier, we would achieve it if we could find a predictor p such that $p(X, A) = Y, \forall (X, A, Y) \sim D$. However, as we have seen, there is a high likelihood of some form of bias. Therefore, we need a method that allows us to ensure that we are minimizing the potential harmful effect that bias could have on the learning process.

We will now define useful concepts that will help us continue developing the theoretical framework for fairness definitions:

Definition 4.1.1 (Independence of components of a random vector)**.** Let X be a random vector: $X = (X_1, \dots, X_n), X_i\colon (\Omega, \mathcal{A}, P) \to (\mathbb{R}, \mathcal{B}, P_{X_i}), \forall i \in \{1, \dots, n\}$. The components $X_1, \dots, X_n$ of X are independent if they meet:

$$F_X(x) = F_{X_1}(x_1) F_{X_2}(x_2) \dots F_{X_n}(x_n) \tag{4.1}$$

Where F_X is the distribution function of the random vector X, and F_{X_i} is the distribution function of each one-dimensional random variable $X_i, \forall i \in \{1, \dots, n\}$.

Lemma 4.1.0.1 (First characterization of independence of components of a random vector)**.** *Let X be a discrete random vector, where $X = (X_1, \dots, X_n), X_i\colon (\Omega, \mathcal{A}, P) \to (E_i, \mathcal{B}, P_{X_i}), \forall i \in \{1, \dots, n\}$. The components X_1, X_2 are independent if and only if $\forall (x_1, \dots, x_n) \in E_1 \times \dots \times E_n$,*

$$P_X(x_1, \dots, x_m) = p_{X_1}(x_1) \dots p_{X_n}(x_n) = P[X_1 = x_1] \dots P[X_n = x_n] \tag{4.2}$$

Where p_{X_i} is the probability mass function of the random variable $X_i, \forall i \in \{1, \dots, n\}$.

Lemma 4.1.0.2 (Characterization of independence of the components of a two-dimensional discrete random vector)**.** *Let X be a random two-dimensional discrete random vector: $X = (X_1, X_2)$,*

$X_i\colon (\Omega, \mathcal{A}, P) \to (E_i, \mathcal{B}, P_{X_i}), \forall i \in \{1,2\}$. *Random variables* X_1, X_2 *are independent* $\Leftrightarrow P[X_1 = x_1 | X_2 = x_2] = P[X_1 = x_1], \forall (x_1, x_2) \in E_1 \times E_2$

Proof. $\boxed{\Rightarrow}$ As X_1 and X_2 are independent, $P[X_1 = x_1, X_2 = x_2] = P[X_1 = x_1]P[X_2 = x_2]$. It is known that $X_1|X_2$ is a two-dimensional random variable, with probability mass function $p_{X_1|X_2}(x_1, x_2) = \frac{P[X_1=x_1, X_2=x_2]}{P[X_2=x_2]} = \frac{P[X_1=x_1]P[X_2=x_2]}{P[X_2=x_2]} = P[X_1 = x_1]$.

$\boxed{\Leftarrow}$ It is known that $P[X_1 = x_1 | X_2 = x_2] = p_{X_1|X_2}(x_1, x_2) = \frac{P[X_1=x_1, X_2=x_2]}{P[X_2=x_2]} = P[X_1 = x_1] \Rightarrow P[X_1 = x_1, X_2 = x_2] = P[X_1 = x_1]P[X_2 = x_2]$, therefore X_1 and X_2 are independent. □

In order to abbreviate, we will use the notation $P[X_1|X_2] = P[X_1]$ to refer to the fact that $P[X_1 = x_1 | X_2 = x_2] = P[X_1 = x_1], \forall (x_1, x_2) \in E_1 \times E_2$.

Lemma 4.1.0.3. *Let* X *be a two-dimensional discrete random vector,* $X = (X_1, X_2), X_i\colon (\Omega, \mathcal{A}, P) \to (E_i, \mathcal{B}, P_{X_i}), \forall i \in \{1,2\}$, *where additionally,* $E_2 = \{e_\alpha, e_\beta\}$. *Random variables* X_1, X_2 *are independent* $\Leftrightarrow P[X_1 = x_1 | X_2 = e_\alpha] = P[X_1 = x_1 | X_2 = e_\beta], \forall x_1 \in E_1$

Proof. $\boxed{\Rightarrow}$ Both X_1 and X_2 are independent, so using the previous lemma, $P[X_1 = 1 | X_2 = e_\alpha] = P[X_1 = x_1 | X_2 = e_\beta] = P[X_1 = x_1], \forall x_1 \in E_1$.

$\boxed{\Leftarrow}$ Now $P[X_1 = x_1 | X_2 = e_\alpha] = P[X_1 = x_1 | X_2 = e_\beta], \forall x_1 \in E_1$. The events $X_2 = e_\alpha$ and $X_2 = e_\beta$ constitute an Ω partition, and additionally $P[X_2 = e_\beta] = 1 - P[X_2 = e_\alpha]$, so we can apply the law of total probability the following way:

$$\begin{aligned} P[X_1 = x_1] &= P[X_1 = x_1 | X_2 = e_\alpha]P[X_2 = e_\alpha] + P[X_1 = x_1 | X_2 = e_\beta]P[X_2 = e_\beta] \\ &= P[X_1 = x_1 | X_2 = e_\alpha](P[X_2 = e_\alpha] + P[X_2 = e_\beta]) \\ &= P[X_1 = x_1 | X_2 = e_\alpha](P[X_2 = e_\alpha] + 1 - P[X_2 = e_\alpha]) \\ &= P[X_1 = x_1 | X_2 = e_\alpha] = P[X_1 = x_1 | X_2 = e_\beta], \forall x_1 \in E_1. \end{aligned} \tag{4.3}$$

And using Lemma 4.1.0.2, X_1 and X_2 are independent. □

The following notation abbreviation will be used $P[X_1|X_2 = e_\alpha] = P[X_1|X_2 = e_\beta]$ to denote the fact that $P[X_1 = x_1 | X_2 = e_\alpha] = P[X_1 = x_1 | X_2 = e_\beta], \forall x_1 \in E_1$

4.1.1. Group fairness

Group fairness is based on the premise that for fairness to prevail, there should be no significant probabilistic differences in the classifications made among different demographic groups. We will address decision problems involving two distinct demographic groups, one that can be considered privileged in social or historical terms, and the other, unprivileged, aiming to minimize differences between them. The way we consider these probabilities will lead to the emergence of multiple measures, which will be discussed below.

One of the major advantages of fairness definitions within this field is that they are independent of the problem to be applied, as specific measures or considerations for the problem at hand do not need to be established in their definition. This is because they are entirely probabilistic, unlike the other two fairness models. A primary disadvantage could be that these measures generally face detection issues when multiple protected groups are considered

simultaneously, and new protected groups are generated as intersections of several of these [Binns, 2020]. Another main drawback is that some measures falling under this interpretation of fairness cannot be simultaneously obtained, as we will see later on.

4.1.2. Unawareness

Definition 4.1.2 (Unawareness)**.** A predictor p satisfies the condition of unawareness if $p(X, A) = p(X)$, or in other words, the value of the sensitive attribute A is not used during the prediction process.

What we can observe here is that, in theory, we completely eliminate disparate treatment, as we treat individuals indifferently regardless of the value they take on the sensitive attribute. However, this is not entirely the case, as some random variable considered in X may be highly correlated to the attribute A, and therefore, we may still indirectly take the sensitive attribute into account in our predictions, not solving the problem. These attributes are called proxy attributes. A combination of X attributes can also be a proxy for attribute A (for example, the combination of address and income can be a proxy for ethnicity, while each of them considered separately may not be proxies for that attribute). Even in the absence of proxies for attribute A, removing A results in a loss of information that could be crucial both in the learning process and in the final classification.

For these reasons, although it may be a very simple and initially effective approximation, it will not be genuinely useful in most cases. The preference is given to the use of other measures that more effectively ensure this fact, as will be seen in counterfactual fairness.

4.1.3. Demographic parity

Definition 4.1.3 (Demographic parity)**.** A predictor p satisfies the condition of demographic parity if p and A are independent. In our case, where the attribute A can only take the values 0 and 1, the condition can be expressed as $P[p = \alpha | A = 0] = P[p = \alpha | A = 1], \forall a \in \{0, 1\}$, which will be expressed as $P[p|A = 0] = P[p|A = 1] = P[p]$.

If we were dealing with a classification problem involving a non-binary sensitive attribute, it would be preferable to define demographic parity using independence rather than opting for the exhaustive definition. However, in the current case, it will be convenient and later necessary to demonstrate the fairness impossibility theorems. Since our predictor is binary, it is also clear, using this definition, that $P[p = \alpha | A = 0] = P[p = \alpha | A = 1] \Leftrightarrow P[p = 1 - \alpha | A = 0] = P[p = 1 - \alpha | A = 1]$, which justifies the simplification in the notation.

This measure aims to assign the same probability of classifying as positive/negative to an individual regardless of the value of their sensitive attribute. There is an extension to this definition known as "conditional demographic parity", where we add to the conditioning the value of an attribute or a set of attributes considered relevant, known as "legitimizing factors" (L). Therefore, the adapted definition would be $P[p = \alpha | L = l, A = 0] = P[p = \alpha | L = l, A = 1], \forall a \in \{0, 1\}$ [Verma and Rubin, 2018].

This way of incorporating fairness is commonly used, but often through the use of relaxed measures. This is because the condition is quite restrictive, and we could significantly lose

accuracy if we had to ensure it holds. Some common ways in which this measure can be relaxed include the following:

- **Relative difference**: As $P[p|A=0] = P[p|A=1] \Leftrightarrow \frac{P[p|A=0]}{P[p|A=1]} = 1$, we can relax the measure incorporating a constant $\epsilon \in [0,1]$ so that $\frac{P[p|A=0]}{P[p|A=1]} \geq 1-\epsilon$.
- **Absolute difference**: As $P[p|A=0] = P[p|A=1] \Leftrightarrow P[p|A=0] - P[p|A=1] = 0$, we can relax the measure incorporating a constant $\epsilon \in [0,1]$ so that $|P[p|A=0] - P[p|A=1]| \leq \epsilon$.

These relaxation rules have been part of significant measures, for example, in the Uniform Guidelines for Employee Selection Procedures, where the 80% rule for adverse impact was employed in the context of employee hiring [Biddle, 2006]. It has been studied and endorsed [Hu and Chen, 2018].

Among other advantages, it helps to increase the importance of the disadvantaged class over time. This is because, in the decision-making process, the probabilities of predicting an individual as positive or negative, conditioned on the value of the sensitive attribute, are forced to be equal or similar. These short- or medium-term decisions can make a significant difference in the long run. Additionally, this measure is independent of the true value of the variable to be predicted (Y), making it useful in contexts where this measure is difficult to quantify.

However, this latter fact can also be considered a disadvantage. For example, it ignores the possible correlation between A and Y. With high probability, the perfect predictor for this sample will not satisfy this condition since, in most cases $P[Y=0|A=0] \neq P[Y=0|A=1]$. Let us consider an example where we are in a job interview where there is an equal number of Caucasians and Asians, and by chance, there are more prepared Asians than Caucasians. Regardless of this fact, we would end up hiring an equal number of Caucasians and Asians, potentially assuming hiring an unprepared Caucasian, as we only care about the probability conditioned on the sensitive attribute.

4.1.4. Equalized odds

Definition 4.1.4 (Equalized odds)**.** A predictor p satisfies the condition of equalized odds if the probability p conditioned on the value of Y is independent of the value of A. In our context we can express this condition as: $P[p=r|Y=z, A=0] = P[p=r|Y=z, A=1] = P[p|Y], \forall r, z \in \{0,1\}$, which will be expressed as $P[p|Y, A=0] = P[p|Y, A=1] = P[p|Y]$.

This measure is also known as "separation". Similar to demographic parity, this definition can be relaxed, allowing for non-compliance up to a certain margin. The primary way to relax it is often to use the measure known as "equality of odds", whose definition is as follows:

Definition 4.1.5 (Equality of odds)**.** A predictor p satisfies the condition of equality of odds if $P[p \neq Y|A=0] = P[P \neq Y|A=1]$.

The issue with this relaxed definition is that there can be an imbalance between false positives $(P[p=1|Y=0])$ and false negatives $(P[p=0|Y=1])$. Another way to relax it is to exclusively consider the true positive rate, which has traditionally been more relevant in

some applications than true negatives. It is known as "equality of opportunity" and is defined as follows:

Definition 4.1.6 (Equality of opportunity)**.** A predictor p satisfies the condition of equality of opportunity if $P[p = 1|Y = 1, A = 0] = P[p = 1|Y = 1, A = 1]$. It is also known as "true positive parity".

Analogously, true negative parity, false positive parity, and alse negative parity can be defined. One way to relax this equalized odds condition is to use one or some of these parities.

As positive highlights, we can say that errors are reduced uniformly for each of the demographic subgroups. Additionally, contrary to what happened in the case of demographic parity, a perfect predictor always satisfies the condition of equalized odds, since $p = Y$.

One of the main problems of using this fairness measure is that over the long term, disparities may increase between the two demographic groups. For example, consider two groups of 100 people each, where the first group has 50 qualified individuals for a job position, while the second group has only 1. If we need to choose 26 people using equalized odds, we will select 25 indiviuals from the first group and 1 from the second. This results in individuals from the first group having more opportunities to secure the position than those from the second group, potentially creating a feedback loop. A better job leads to improved family conditions, enabling the children of the first group to access better education, and consequently, they become better qualified for the position compared to the children of the second group.

We will now provide a necessary and sufficient condition for the equalized odds condition to be satisfied [Tang and Zhang, 2020]:

Theorem 4.1.1 (Characterization of equalized odds)**.** *Let $\mathcal{X}, \mathcal{Y}$ and $\mathcal{A}$ the domains of the random variables X, Y and A respectively. Let us suppose that A and Y are independent, and inside $X = (X_1, \ldots, X_n)$ there are variables which are not independent of Y, expressing $S_A^{(y)} = \{a \in A : \exists x \in \mathcal{X} : p(a, x) = y\}$, and also $S_{X|a}^{(y)} = \{x \in \mathcal{X} : p(a, x) = y\}$, then the condition of equalized odds will be satisfied if and only if the next two conditions are met:*

I $S_A^{(y)} = \mathcal{A}, \forall t \in \mathcal{Y}$

II $\sum_{x \in S_{X|a}^{(y)}} P[X = x|Y = y, A = a] = \sum_{x \in S_{X|a'}^{(y)}} P[X = x|Y = y, A = a'], \forall y \in \mathcal{Y}, \forall a, a' \in \mathcal{A} : a \neq a'$

Proof. For the equalized odds conditions to be satisfied, the following expression needs to hold:

$$P[p = \alpha|Y = y, A = a] = P[p = \alpha|Y = y], \forall a \in A, \forall \alpha, y \in \mathcal{Y} \tag{4.4}$$

We can manipulate the left-hand side of the equation to obtain the following equality:

$$P[p = \alpha|Y = y, A = a] = \sum_{x \in \mathcal{X}} P[p = \alpha|Y = y, A = a, X = x] P[X = x|Y = y, A = a] \tag{4.5}$$

We know that $p = p(X, A)$ is a deterministic function that depends on A and X, and using this we can write the following equality:

$$P[p(X,A) = \alpha | Y = y, A = a, X = x] = P[p(X,A) = \alpha | A = a, X = x] \in \{0,1\} \tag{4.6}$$

The value $P[p(X, A) = \alpha | A = a, X = x]$ will be 0 or 1, depending on wether $p(x, a) = \alpha$ or not. From this last equality, we can see that the conditional probability $P[X = x | A = a, Y = y]$ can contribute to the summation only when $p(a, x) = \alpha$. Therefore, we can rewrite the left-hand side of equation (4.4) as follows:

$$P[p = \alpha | Y = y, A = a] = \sum_{s \in S_{X|a}^{(\alpha)}} P[X = x | Y = y, A = a] \tag{4.7}$$

We will denote $Q^{(\alpha)}(a, y) = \sum_{s \in S_{X|a}^{(\alpha)}} P[X = x | Y = y, A = a]$. Now, let us manipulate the right-hand side of equation (4.4) to obtain:

$$\begin{aligned} P[p = \alpha | Y = y] &= \sum_{a \in \mathcal{A}} \sum_{x \in \mathcal{X}} P[p = \alpha | Y = y, A = a, X = x] P[X = x, A = a | Y = y] \\ &= \sum_{a \in S_A^{(\alpha)}} \sum_{x \in S_{X|a}^{(\alpha)}} P[X = x | Y = y, A = a] P[A = a | Y = y] \\ &= \sum_{a \in S_A^{(\alpha)}} Q^{(\alpha)}(a, y) P[A = a | Y = y] \end{aligned} \tag{4.8}$$

The condition of equalized odds occurs if and only if equation (4.4) is satisfied. Therefore, the value $Q^{(\alpha)}(a, y)$ cannot change with a. Thus, by making the substitutions that were established, equation (4.4) can be rewritten as:

$$Q^{(\alpha)}(a,y) = \sum_{a \in S_A^{(\alpha)}} Q^{(\alpha)}(a,y) P[A = a | Y = y] = Q^{(\alpha)}(a,y) \sum_{a \in S_A^{(\alpha)}} P[A = a | Y = y] \tag{4.9}$$

This equality gives us condition I), since if it does not hold, it would imply that $\sum_{a \in S_A^{(\alpha)}} P[A = a|Y)y] < 1$. Condition II) arises naturally because, as we mentioned earlier, the value of $Q^{(\alpha)}(a, y)$ does not change with a. By definition, we directly have the condition $Q^{(\alpha)}(a, y) = \sum_{s \in S_{X|a}^{(\alpha)}} P[X = x | Y = y, A = a] = \sum_{s \in S_{X|a'}^{(\alpha)}} P[X = x | Y = y, A = a']$

Finally, the entire procedure is reversible, thus establishing the double implication. □

This property is useful for implementing post-processing techniques with the aim of ensuring that the equalized odds condition holds for the applied predictor. [Tang and Zhang, 2020].

4.1.5. Predictive rate parity

Definition 4.1.7 (Predictive rate parity)**.** A predictor p satisfies the condition of predictive rate parity if the probability of the attribute Y taking a value conditioned on the value of p is independent of A. In our context we can express this condition as $P[Y = z | p = r, A = 0] = P[Y = z | p = r, A = 1], \forall z, r \in \{0, 1\}$ [Zafar et al., 2017a]. We can simplify it, writing $P[Y | p, A = 0] = P[Y | p, A = 1]$.

This condition is also known as "sufficiency". It is equivalent to verifying two conditions: $P[Y = 1|p = 1, A = 0] = P[Y = 1|p = 1, A = 1]$ and $P[Y = 0|p = 0, A = 0] = P[Y = 0|p = 0, A = 1]$. These are known in the literature as "positive and negative predictive rate parity", respectively, or also "positive and negative predictive values rate parity". One possible way to relax this measure would be to consider only one of these two measures instead of both.

In this case, it also happens that with a perfect predictor, where $Y = p$, predictive rate parity is trivially satisfied, just like with equalized odds. Therefore, it is a desirable feature if one considers using this fairness definition.

The problems associated with its use are similar to those that occurred with equalized odds. As will be seen later, in fact, it is impossible to find a predictor p that is able to satisfy any pair of group fairness measures among the three previously proposed: demographic parity, equalized odds, and predictive rate parity.

To define both the next measures, it is necessary to consider that our predictor does not have a discrete output in the range of possible values that the attribute Y can take. We will consider that our predictor p assigns a score to each individual, denoted as $S(X, A) = s$. For simplicity and without loss of generality, we will assume that $s \in [0, 1]$. This value, for example, could be interpreted as the probability that a particular example belongs to the positive class. This value will later be subject to transformation into a label, with one of the most common approaches being the establishment of a threshold value above which instances will be classified as positive. Initially, we will focus only on the score assignment and not on the transformation into a label. Therefore, we can directly refer to the prediction process carried out by S.

4.1.6. Calibration

Definition 4.1.8 (Calibration)**.** A predictor p satisfies the condition of calibration if $P[Y = 1|S = s, A = 0] = P[Y = 1|S = s, A = 1]$, for every possible value of s.

The application of this measure is subject to being able to know all possible values that S can take. In case the number of possible values is discrete and manageable, one could consider using it. However, in the case of a continuous range, its application as defined will not be appropriate, as the probability of S taking a specific value will be equal to 0. In the case of continuous S, to continue applying this definition, threshold values are used as follows:

$$P[Y = 1|S \geq s, A = 0] = P[Y = 1|S \geq s, A = 1] \text{ , for every value of } s. \tag{4.10}$$

In practice, there are several heuristics to achieve calibration. For example, Platt scaling is a popular method that takes a possibly uncalibrated score and treats it as a feature to train a logistic regression. The goal is to find coefficients a and b to fit the value of a logistic function:

$$r = \frac{1}{1 + e^{(aS+b)}} \tag{4.11}$$

So that it fits the actual value of Y, attempting to minimize the loss function known as

"LogLoss" or "logarithmic loss function", which will be discussed in Section 8.3:

$$L(y, r) = -\left(y \log(r) + (1 - y) \log(1 - r)\right) \tag{4.12}$$

This measure is widely used, with risk assessment in the field of criminology being one of the clearest examples of its application [Danner et al., 2016]. It represents, as is evident, a good extension of the concept of predictive rate parity in case the predictor returns a score instead of a label. It can also be considered as a generalization of predictive rate parity to non-binary classification contexts. However, being a direct extension of the previous case, it suffers from the same problems [Corbett-Davies and Goel, 2018]. The score value s could be continuous, but in the case it can only take the values 0 or 1, achieving calibration for a predictor p would be equivalent to ensuring that it satisfies the predictive rate parity condition.

Let us consider the case where, having a continuous score s, classification is done setting a threshold value s_{th}, so that individuals with $s(x, a) > s_{th} \Rightarrow p(x, a) = 1$ and individuals with $s(x, a) \leq s_{th} \Rightarrow p(x, a) = 0$. By following this procedure, we will not always achieve that our predictor satisfies the predictive rate parity condition, as we will see [Garg et al., 2020]. Therefore the probability $P[Y = 1|p = 1, A]$ can be expressed as $P[Y = 1|S > s_{th}, A]$. Using this, we can write:

$$\begin{aligned} P[Y, S > s_{th}|A] &= \int_{s_{th}}^{1} \underbrace{P[Y|S, A]}_{\text{Calibration}} P[S|A] ds \Rightarrow \\ \Rightarrow \underbrace{P[Y|S > s_{th}, A]}_{\text{Predictive rate parity}} &= \frac{\int_{s_{th}}^{1} P[Y|S, A] P[S|A] ds}{\int_{s_{th}}^{1} P[S|A] ds} \end{aligned} \tag{4.13}$$

This equation relates predictive rate parity to calibration, showing that even when the calibration term ($P[Y|S, A]$) is the same for both groups, the probability distribution of the score, expressed in the lower equation by $P[S|A]$, can vary among different groups, causing parity in predictive ratios not to be satisfied. To make it more intuitive, consider a special case where there are only two values s_1, s_2 above the threshold value s_{th}, so that $P[S|A] \neq 0$. In other words, all individuals receiving risk scores above the threshold have the possibility of receiving only two scores: s_1 or s_2. Therefore, $P[S > s_{th}|A] = P[S = s_1|A] + P[S = s_2|A]$. In this special case, the last equation is reduced to:

$$P[Y = 1|P = 1, A] = \frac{P[Y = 1|S = s_1, A] P[S = s_1|A] + P[Y = 1|S = s_2, A] P[S = s_2|A]}{P[S = S_1|A] + P[S = s_2|A]} \tag{4.14}$$

Let us give an example where calibration is satisfied, but predictive rate parity is not. Consider two demographic groups: a and b. Suppose that S can return only three possible values: $0.25, 0.5, 0.75$. We set a threshold value $s_{th} = 0.49$. Suppose there are 100 individuals in each group, and the classification is done according to Table 4.1. For each group, the table shows the number of instances for which a certain score has been predicted, as well as the number of instances that were actually positive among those predicted with that score:

It can be seen that the calibration condition is satisfied because, for each score, the proportion of positive individuals among the predicted ones is equal for both groups. However, the predictive rate parity condition is not satisfied because in this case $P[Y = 1|p = 1|A = a] = \frac{2}{3}$, while $P[Y = 1|p = 1|A = b] = \frac{7}{12}$. Another measure with a more restictive definition based

Table 4.1.: Prediction procedure where calibration is fulfilled and predictive rate parity is not fulfilled.

Score	**Group** a		**Group** b		**Prediction after setting the threshold** $s\_th = 0.49$
	Predicted	Positives	Predicted	Positives	
0.25	40	16	40	16	Negative
0.50	20	10	40	20	Negative
0.75	40	30	20	15	Positive
Total	100	56	100	51	

on calibration, but also commonly used, is perfect calibration [Pleiss et al., 2017]:

Definition 4.1.9 (Perfect calibration)**.** A predictor p satisfies the condition of perfect calibration if $P[Y = 1|S = s, A = 0] = P[Y = 1|S = s, A = 1] = s$, for every possible value of s.

This condition is also known as "group calibration". If a predictor p satisfies this measure, it will not only meet calibration but also ensure that the score values assigned by the predictor truly correspond to the probabilities of belonging to the positive class. Let us see an interesting property of predictors that satisfy perfect calibration:

Lemma 4.1.1.1. *If a predictor p meets predictive rate parity, then it exists a function $l\colon [0,1] \to [0,1] : l(p)$ which satisfies the condition of perfect calibration.*

Proof. Let us consider a specific demographic group a and let $l(\alpha) = P[Y = 1|p = \alpha, A = a]$. Since p satisfies the predictive rate parity condition, this probability will be the same regardless of the value of a. Now, considering two different demographic groups a and b, we have:

$$\begin{aligned} \alpha &= P[Y = 1|l(p) = \alpha, A = a] \\ &= P[Y = 1|p \in l^{-1}(\alpha), A = a] \\ &= P[Y = 1|p \in l^{-1}(\alpha), A = b] \\ &= P[Y = 1|l(p) = \alpha, A = b] \end{aligned} \tag{4.15}$$

Where, for the third equality, we have once again used the predictive rate parity condition, thereby satisfying the perfect calibration condition. □

4.1.7. Balance for the positive/negative class

Definition 4.1.10 (Balance for the positive/negative class)**.** A predictor p satisfies the condition of balance for the positive class if $\mathbb{E}[s|Y = 1, A = 0] = \mathbb{E}[s|Y = 0, A = 0]$. Similarly, it satisfies the condition of balance for the negative class if $\mathbb{E}[s|Y = 0, A = 0] = \mathbb{E}[s|Y = 1, A = 0]$ [Verma and Rubin, 2018].

This condition can be seen as a generalization of the equalized odds metric to non-binary cases. Indeed, if s can only take the values 0 and 1, the value of s would directly become the prediction, and we would have exactly the equalized odds condition. Therefore, it is a natural extension to this family of metrics.

In [Kleinberg et al., 2016], it is proven that the only cases in which the conditions of balance for the positive class, balance for the negative class, and calibration can be simultaneously

met are when p is a perfect predictor, or there is an equal number of positive instances in each demographic group.

4.2. Fairness impossibility theorems

The three main definitions of group fairness —demographic parity, equalized odds, and predictive rate parity— have straightforward definitions and can be easily expressed in probabilistic terms. However, they have a drawback: no pair of them can be satisfied simultaneously for a given problem and classifier, except in extreme cases. Therefore, we will establish hypotheses that naturally arise to avoid such scenarios, known as "non-degeneracy hypothesis":

- **Dependence between A and Y**: If there were no such dependency, the attribute A could simply not be considered in the feature set, as it will not help to predict Y.
- **Dependence between p and Y**: If this were not the case, no learning process would be taking place, as our prediction would not depend on the attribute to be predicted.

Once these hypotheses are established, we will verify that if the predictor $p$satisfies either demographic parity, equalized odds, or predictive rate parity, any of the other two conditions can also be satisfied. The basis for the proof of these theorems can be found in [del Barrio et al., 2020].

Theorem 4.2.1 (Impossibility of demographic parity and equalized odds)**.** *Under the non-degeneracy hypothesis, demographic parity and equalized odds conditions cannot hold simultaneously for any classifier.*

Proof. Let us prove it by contradiction. To begin with, let us consider a relevant equality, which will provide another way to express $P[p = 1|A = a]$:

$$\begin{aligned} P[p=1|A=a] &= P[p=1,Y=1|A=a] + P[p=1,Y=0|A=a] \\ &= \frac{P[p=1,Y=1,A=a]}{P[A=a]} + \frac{P[p=1,Y=0,A=a]}{P[A=a]} \\ = \frac{P[p=1,Y=1,A=a]}{P[Y=1,A=a]}\frac{P[Y=1,A=a]}{P[A=a]} &+ \frac{P[p=1,Y=0,A=a]}{P[Y=0,A=a]}\frac{P[Y=0,A=a]}{P[A=a]} \\ = P[p=1|Y=1,A=a]P[Y=1|A=a] &+ P[p=1|Y=0,A=a]P[Y=0|A=a] \end{aligned} \tag{4.16}$$

And this happen $\forall a \in \{0, 1\}$.

Let us suppose now that the condition of equalized odds holds, and therefore $P[p = 1|Y = 1, A = 0] = P[p = 1|Y = 1, A = 1] = P[p = 1|Y = 1]$, as well as $P[p = 1|Y = 0, A = 0] = P[p = 1|Y = 0, A = 1] = P[p = 1|Y = 0]$.

Let us consider that the condition of demographic parity must also hold, so we have the following expression:

$$0 = P[p = 1|A = 1] - P[p = 1|A = 0] \tag{4.17}$$

Using the last equality from Equation (4.16) this can be expressed as:

$$\begin{aligned} 0 = P[p=1|Y=1,A=1]P[Y=1|A=1] + P[p=1|Y=0,A=1]P[Y=0|A=1] - \\ (P[p=1|Y=1,A=0]P[Y=1|A=0] + P[p=1|Y=0,A=0]P[Y=0|A=0]) \end{aligned} \tag{4.18}$$

As equalized odds holds, the previous expression is equivalent to:

$$\begin{aligned} 0 = P[p=1|Y=1]P[Y=1|A=1] + P[p=1|Y=0]P[Y=0|A=1] - \\ (P[p=1|Y=1]P[Y=1|A=0] + P[p=1|Y=0]P[Y=0|A=0]) \end{aligned} \tag{4.19}$$

By operating we obtain:

$$\begin{aligned} 0 = P[p=1|Y=0](P[Y=0|A=1] - P[Y=0|A=0]) + \\ P[p=1|Y=1](P[Y=1|A=1] - P[Y=1|A=0]) \end{aligned} \tag{4.20}$$

We observe that the probabilities in both subtractions are opposite. Therefore, we can rewrite everything as:

$$\begin{aligned} 0 = P[p=1|Y=0](P[Y=0|A=1] - P[Y=0|A=0]) + \\ P[p=1|Y=1](-P[Y=0|A=1] + P[Y=0|A=0]) \end{aligned} \tag{4.21}$$

And by operating we finally obtain:

$$0 = (P[p=1|Y=0] - P[p=1|Y=1])(P[Y=0|A=1] - P[Y=0|A=0]) \tag{4.22}$$

This expression is a product which result is 0, and therefore one of the two factors must be 0. However, if either factor were 0, it would violate the non-degeneracy hypothesis. □

Theorem 4.2.2 (Impossibility of demographic parity and predictive rate parity)**.** *Under the non-degeneracy hypothesis, demographic parity and predictive rate parity conditions cannot hold simultaneously for any classifier.*

Proof. Let us prove it by contradiction. Assuming both parities hold true simultaneously, $P[p|A] = P[p]$ and $P[Y|p,A] = P[Y|p]$. Due to the non-degeneracy hypothesis, we know that both Y and A are dependent, thus $P[Y|A] \neq P[Y]$. Nonetheless, by a similar process done at the previous proof and using the last equality from Equation (4.16) one can express $P[Y|A]$ as:

$$\begin{aligned} P[Y|A] &= P[Y,p=0|A] + P[Y,p=1|A] \\ &= \frac{P[Y,p=0,A]}{P[A]} + \frac{P[Y,p=1,A]}{P[A]} \\ &= \frac{P[Y,p=0,A]}{P[p=0,A]}\frac{P[p=0,A]}{P[A]} + \frac{P[Y,p=1,A]}{P[p=1,A]}\frac{P[p=1,A]}{P[A]} \\ &= P[Y|p=0,A]P[p=0|A] + P[Y|p=1,A]P[p=1|A] \end{aligned} \tag{4.23}$$

Since both parities occur, we can rewrite the last expression from the previous equation as:

$$\begin{aligned} P[Y|A] = P[Y|p=0,A]P[p=0|A] + P[Y|p=1,A]P[p=1|A] = \\ = P[Y|p=0]P[p=0] + P[Y|p=1]P[p=1] \end{aligned} \tag{4.24}$$

Using the law of total probability, this simplifies to $P[Y]$, and consequently, $P[Y|A] = P[Y]$. Therefore, by definition, Y is independent of A, which leads to a contradiction.

□

Theorem 4.2.3 (Impossibility of equalized odds and predictive rate parity)**.** *Under the non-degeneracy hypothesis, equalized odds and predictive rate parity conditions cannot hold simultaneously for any classifier except for a perfect classifier.*

Proof. Let us consider that both conditions hold true simultaneously. In particular, we will be focusing on the following equalities:

$$\begin{aligned} P[p=1|Y=1,A=0] &= P[p=1|Y=1,A=1] \\ P[p=1|Y=0,A=0] &= P[p=1|Y=0,A=1] \\ P[Y=1|p=1,A=0] &= P[Y=1|p=1,A=1] \end{aligned} \tag{4.25}$$

We will now use Equation 4.16 to write some expressions which are true $\forall a \in \{0,1\}$:

$$\begin{aligned} P[p=1|A=a] = P[p=1|Y=1,A=a]P[Y=1|A=a] + \\ P[p=1|Y=0,A=a]P[Y=0|A=a] \end{aligned} \tag{4.26}$$

Having stated this expression, let us consider the following one:

$$\begin{aligned} P[p=1|Y=1,A=a]P[Y=1|A=a] = \frac{P[p=1,Y=1,A=a]}{P[Y=1,A=a]}\frac{P[Y=1,A=0]}{P[A=a]} = \\ \frac{P[p=1,Y=1,A=a]}{P[p=1,A=a]}\frac{P[p=1,A=0]}{P[A=a]} = P[Y=1|p=1,A=a]P[p=1|A=a] \end{aligned} \tag{4.27}$$

We can substitute $P[p=1|A=a]$ in the last expression with what we have just obtained, resulting in the following expression:

$$\begin{aligned} P[p=1|Y=1,A=a]P[Y=1|A=a] = \\ P[Y=1|p=1,A=a](P[p=1|Y=1,A=a]P[Y=1|A=a] + P[p=1|Y=0,A=a]P[Y=0|A=a]) = \\ P[Y=1|p=1,A=a](P[p=1|Y=1,A=a]P[Y=1|A=a] + P[p=1|Y=0,A=a](1-P[Y=1|A=a])) \end{aligned} \tag{4.28}$$

We can now solve for $P[Y=1|A=a]$ to obtain:

$$P[Y=1|A=a] = \frac{P[Y=1|p=1,A=a]P[p=1|Y=0,A=a]}{P[Y=1|p=1,A=a]P[p=1|Y=0,A=a]+(1-P[Y=1|p=1,A=a])P[p=1|Y=1,A=a]} \tag{4.29}$$

Using Equation 4.25 we can see that:

$$P[Y=1|A=0] = \frac{P[Y=1|p=1]P[p=1|Y=0]}{P[Y=1|p=1]P[p=1|Y=0] + (1-P[Y=1|p=1])P[p=1|Y=1]} \tag{4.30}$$

This statement holds true if the denominator is not 0. In cases where it is 0, since both terms are positive, each must be 0. For the expression to represent a valid probability value, which ranges between 0 and 1, the numerator must also be 0 resulting in a $\frac{0}{0}$ indeterminate form. In this case both conditions could hold simultaneously, but the resulting classification will be poor, as $P[Y=1|p=1,A=a] = 0 \forall a$. Another interesting case is that of a perfect classifier,

where indeed both conditions can be satisfied. In all other cases, A and Y will be independent, contradicting the non-degeneracy hypothesis. □

4.3. Individual fairness

Individual fairness [Dwork et al., 2011] is based on the principle that similar individuals should be treated similarly. Therefore, the classes assigned to our individuals should be similar for similar individuals. This definition is inherently ambiguous, and, as can be inferred, the main problem that comes with this type of fairness measures lies in the need to define metrics to compare both individuals in the population and their labels, which inherently depends on the specific problem to be solved. There is a wide variety of such metrics with diverse characteristics, and it is not clear what type of metric to use for a given problem. Different interpretations of the problem could lead to using different measures, resulting in different outcomes.

Positive aspects regarding group fairness include the fact that it does not have subfamilies of measures that are inherently contradictory, and predictors that comply with them are less likely to "intentionally" make classification errors to balance conditional probabilities for each demographic subgroup.

To achieve a measure that complies with this definition, the first step is to define a method for measuring the similarity between individuals, as well as a method to measure the similarity in predictions made about them. For this purpose, let us consider the representation of an individual by its feature vector, obtained as a sample realization of the random vector $(X_1, \ldots, X_n, A) \in \mathbb{R}^{n+1}$. To measure similarity between individuals, we will consider the notion of distance.

Definition 4.3.1 (Distance)**.** Let M be a set. A function $d\colon M \times M \to \mathbb{R}$ is a distance $\Leftrightarrow$ it satisfies the following properties:

- **Non-negativity**: $d(x_1, x_2) \geq 0, \forall x_1, x_2 \in M$
- **Non-degegeneracy**: $d(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2, \forall x_1, x_2 \in M$
- **Symmetry**: $d(x_1, x_2) = d(x_2, x_1), \forall x_1, x_2 \in M$
- **Triangle inequality**: $d(x_1, x_2) \leq (x_1, x_3) + d(x_3, x_2) \forall x_1, x_2, x_3 \in M$

Definition 4.3.2 (Metric space)**.** A metric space is a pair (M, d) where M is a set and $d\colon M \times M \to \mathbb{R}$ is a metric.

Similarly, we aim to measure the distance between the predictions of our classifier. In this case, p is considered a random classifier that maps each individual in the population to a new probability function. This induces a new probability space over the set of potential classification outcomes. Let us make a series of definitions to formalize this concept:

Definition 4.3.3 (Δ set associated with another set)**.** Let $E \subset \mathbb{R}$. The set Δ_E is defined as $\Delta_E = \{(E, \mathcal{P}(E), P) : P \text{ is a probability measure}\}$. In other words, it is the set of all possible probability spaces over the measurable space $(E, \mathcal{P}(E))$. We can also interpret this set as the set of all probability metrics defined over the measurable space $(E, \mathcal{P}(E))$.

We can use this definitions to define a predictor function. Given the probability space $(\Omega, \mathcal{A}, P)$, and being the random variable to predict $Y\colon (\Omega, \mathcal{A}, P) \to (E \subseteq \mathbb{R}, \mathcal{B}, P_Y)$, our predictor function will be:

$$\begin{aligned} p\colon \Omega &\to \Delta_E \\ \omega &\to p(\omega) = (E, \mathcal{P}(E), P_\omega) \end{aligned} \tag{4.31}$$

So that each individual ω will be associated with a probability function P_ω over the measurable space $(E, \mathcal{P}(E))$. Therefore, we are interested in measuring the distance between the probability functions associated with each individual. This concept is known as "statistical distance" [Martos Venturini, 2015].

Definition 4.3.4. (Statistical distance) A statistical distance is a measure of similarity between two statistical objects.

These objects can vary significantly in nature; they can be probability measures, samples, random variables, and more. In our case, we are interested in distances used to compare probability measures. Statistical distances differ from traditional distances in that they do not necessarily satisfy the properties of distances; generally, they only satisfy a subset of them, though these conditions could also be relaxed. Examples of these include semi-distances, pseudodistances and quasi-distances. A subgroup with considerable relevance is that of divergences.

Definition 4.3.5 (Individual fairness)**.** Let $(\Omega, \mathcal{A}, P)$ be a probability space, $Y\colon (\Omega, \mathcal{A}, P) \to (E \subseteq \mathbb{R}, \mathcal{B}, P_Y)$ a random variable to predict, and $p\colon \Omega \to \Delta_E$ a probabilistic classifier. Consider a distance function $d_1\colon \Omega \times \Omega \to \mathbb{R}$ and a statistical distance $d_2\colon \Delta_E \times \Delta_E \to \mathbb{R}$. The predictor p satisfies the individual fairness condition if

$$d_2(p(\omega), p(\omega')) \leq d_1(\omega, \omega'), \forall \omega, \omega' \in \Omega \tag{4.32}$$

This condition is also known as "(d_2, d_1)-Lipschitz" [Dwork et al., 2011].

Whether the individual fairness condition is fulfilled or not depends on the chosen functions d_1 and d_2. Generally, to measure the distance between individuals, we will measure the distances between their random feature vectors, denoted as follows:

$$d_1(\omega, \omega') = d_1((X_1, \ldots, X_n, A)(\omega), (X_1, \ldots, X_n, A)(\omega')) \tag{4.33}$$

The metric d_2 must be a statistical distance. In [Dwork et al., 2011], two different divergences are proposed, independent of the problem definition, which allows the fairness-constrained optimization problem to be formulated as a polynomial-sized linear program with respect to the cardinalities of Ω and E:

Definition 4.3.6 (Total variation distance)**.** Let P_1, P_2 be two probability measures, both defined with respect to the same measurable space $(E, \mathcal{P}(E))$. The total variation distance between them is a statistical distance defined as::

$$d_{tv}(P_1, P_2) = \max_{F \subset E} |P_1(F) - P_2(F)| \tag{4.34}$$

Now, let us explore an equivalent way to express this measure, making the calculation much simpler for our context.

Lemma 4.3.0.1 (Characterization of total variation distance). $d_{tv}(P_1, P_2) = \frac{1}{2}\sum_{e\in E}|P_1(e) - P_2(e)|$

Proof. Let $G = \{e \in E : P_1(e) \geq P_2(e)\}$. Let $F \subset E$ be any event. Then, the following inequality chain is satisfied:

$$P_1(F) - P_2(F) \leq P_1(F \cap G) - P_2(F \cap G) \leq P_1(G) - P_2(G) \tag{4.35}$$

The first inequality is true, since $\overline{G}$ the complementary event of the event G, we know that every element $e \in F \cap \overline{G}$ are the only elements in F that follow $P_1(e) - P_2(e) < 0$, and this elements are the only ones removed in $F \cap G$. On the other hand, G has at least as many elements as in $F \cap G$, and additionally, all of them meet $P_1(e) \geq P_2(e)$, from which we can obtain the second inequality. Analogously we can deduce that:

$$P_2(F) - P_1(F) \leq P_2(\overline{G}) - P_1(\overline{G}) \tag{4.36}$$

Furthermore, both bounds that have been set are the same, as $1 = P_1(G) + P_1(\overline{G}) = P_2(G) + P_2(\overline{G}) \Rightarrow P_1(G) - P_2(G) = P_2(\overline{G}) - P_1(\overline{G})$. We are able to reach the bound when we set $F = G$. Using all of the above we can rewrite:

$$d_{tv}(P_1, P_2) = \frac{1}{2}|P_1(G) - P_2(G) + P_2(\overline{G}) - P_1(\overline{G})| = \frac{1}{2}\sum_{e\in E}|P_1(e) - P_2(e)| \tag{4.37}$$

□

Corollary 4.3.0.1. *Total variation distance is a mathematical distance.*

Proof. All properties are trivial, except for triangle inequality. Let us consider three probability metrics P_1, P_2 and P_3. Then, $d_{tv}(P_1, P_2) = \frac{1}{2}\sum_{e\in E}|P_1(e) - P_2(e)| = \frac{1}{2}\sum_{e\in E}|P_1(e) - P_3(e) + P_3(e) - P_2(e)| \leq \frac{1}{2}\sum_{e\in E}|P_1(e) - P_3(e)| + \frac{1}{2}\sum_{e\in E}|P_3(e) - P_2(e)| = d_{tv}(P_1, P_3) + d_{tv}(P_3, P_2)$ □

Definition 4.3.7 (d_∞ relative measure)**.** Let P_1, P_2 be two probability measures defined over the same measurable space $(E, \mathcal{P}(E))$, where we will additionally require that $P_1(e), P_2(e) > 0, \forall e \in E$. Then, d_∞ relative measure between both of them, in the case where E is continuous, is defined as:

$$d_\infty(P_1, P_2) = \sup_{e\in E} \log\left(\max\left\{\frac{P_1(e)}{P_2(e)}, \frac{P_2(e)}{P_1(e)}\right\}\right) \tag{4.38}$$

Corollary 4.3.0.2. *d_∞ relative measure is a distance.*

Proof. All properties are trivial but the triangle inequality. Let P_1, P_2, P_3 be probability measures. Using the additional hypothesis given at the definition of the measure, the set defined as $\left\{\log\left(\max\left\{\frac{P_1(e)}{P_2(e)}, \frac{P_2(e)}{P_1(e)}\right\}\right) : e \in E\right\}$ is upper bounded, and for that reason, $d_\infty(P_1, P_2)$ exists. The same condition will be needed for each relative set using (P_1, P_3) and (P_2, P_3) to proof the existence of $d_\infty(P_1, P_3)$ and $d_\infty(P_2, P_3)$. As all this sets are upper bounded, we can take a sequence $\{e_n\}_{n\in\mathbb{N}} \subseteq E$ such that $\lim_{n\to\infty}\log\left(\max\left\{\frac{P_1(e)}{P_2(e)}, \frac{P_2(e)}{P_1(e)}\right\}\right) = d_\infty(P_1, P_2)$. We can write this convergent sequence the following way:

$$\{e_n\}_{n\in\mathbb{N}} : e_n = \begin{cases} \log\left(\frac{P_1(e_n)}{P_2(e_n)}\right) = \log\left(\frac{P_1(e_n)}{P_3(e_n)}\right) + \log\left(\frac{P_3(e_n)}{P_2(e_n)}\right), \text{ if } \frac{P_1(e_n)}{P_2(e_n)} > \frac{P_2(e_n)}{P_1(e_n)} \\ \log\left(\frac{P_2(e_n)}{P_1(e_n)}\right) = \log\left(\frac{P_2(e_n)}{P_3(e_n)}\right) + \log\left(\frac{P_3(e_n)}{P_1(e_n)}\right), \text{ if } \frac{P_1(e_n)}{P_2(e_n)} \leq \frac{P_2(e_n)}{P_1(e_n)} \end{cases} \tag{4.39}$$

Using this sequence we can directly verify that:

$$\begin{aligned} \max\left\{\log\left(\frac{P_1(e_n)}{P_2(e_n)}\right), \log\left(\frac{P_2(e_n)}{P_1(e_n)}\right)\right\} \leq \\ \max\left\{\log\left(\frac{P_1(e_n)}{P_3(e_n)}\right), \log\left(\frac{P_3(e_n)}{P_1(e_n)}\right)\right\} + \max\left\{\log\left(\frac{P_3(e_n)}{P_2(e_n)}\right), \log\left(\frac{P_2(e_n)}{P_3(e_n)}\right)\right\} \end{aligned} \tag{4.40}$$

This is true $\forall n \in \mathbb{N}$. For this reason, the sequence $\{e_n\}_{n\in\mathbb{N}}$ will help us establish the following inequality:

$$\begin{aligned} d_\infty(P_1, P_2) &= \lim_{n\to\infty} \log\left(\max\left\{\frac{P_1(e_n)}{P_2(e_n)}, \frac{P_2(e_n)}{P_1(e_n)}\right\}\right) \\ &\leq \lim_{n\to\infty} \log\left(\max\left\{\frac{P_1(e_n)}{P_3(e_n)}, \frac{P_3(e_n)}{P_1(e_n)}\right\}\right) + \lim_{n\to\infty} \log\left(\max\left\{\frac{P_3(e_n)}{P_2(e_n)}, \frac{P_2(e_n)}{P_3(e_n)}\right\}\right) \\ &\leq d_\infty(P_1, P_3) + d_\infty(P_3, P_2) \end{aligned} \tag{4.41}$$

□

Total variation distance ranges from 0 to 1, while d_∞ is not upper bounded. In the case of total variation distance, two probability measures can be considered similar if d_{tv} is close to 0 and different when d_{tv} approaches 1. In the case of d_∞, two distributions may be deemed similar when d_∞ is close to 0 while they can be considered different if $d_\infty \gg 0$.

Besides these two measures, which are presented here for their good properties in implementing constraint optimization problems and their independence from context, many other context-independent measures can be found in [Deza and Deza, 2009]. The distance measure between individuals within the population could be proposed by an expert, as mentioned earlier and as stated in the original publication. However, recently, methods have been proposed to define such measures based on the datasets themselves [Ilvento, 2020], rather than being solely reliant on human expert proposals, although their definition may still involve information that only an expert can provide.

In [Ilvento, 2020], methods are developed based on the definition of submetrics to ensure the fulfillment of individual fairness. Through the construction of these submetrics based on representative elements, it is ensured that the distance cost will not be overestimated, and the information required from our expert is also regulated.

Let us briefly delve into some of the theory in [Mukherjee et al., 2020]. Here, two relatively simple ways of learning metrics from data are proposed, but some level of expert information will always be necessary. These metrics are based on the Mahalanobis distance:

$$d_1(\omega, \omega') = \langle \phi(x_1) - \phi(x_2), \Sigma(\phi(x_1) - \phi(x_2)) \rangle \tag{4.42}$$

Where $\phi\colon \Omega \to \mathbb{R}^d$ is an injective function, and $\Sigma \in S_+^d$, being S_+^d the set of positive-semidefinite matrices of rank d.

4.3.1. FACE Method: Factor Analysis of Comparable Embeddings

The first proposed method is known as "Factor Analysis of Comparable Embeddings", and it involves learning the value of Σ based on comparable samples. These samples can be extracted by experts or manually generated that differ based on certain meaningful forms. In this case, the value of the function ϕ will consist of:

$$\phi_i = A_* x_i + B_* a_i + \epsilon_i \tag{4.43}$$

Where $\phi_i \in \mathbb{R}^d$ is the representation of our individual, identified by the pair (x_i, a_i), and ϵ_i is an error term. A pair of samples will be comparable if their relevant attributes are similar. For example, in [Bolukbasi et al., 2016], where the goal is to eliminate bias in the context of word embeddings, words that only differ in gender context, such as he-she, or queen-king, are considered as similar.

This factorial model decomposes the variance of the representation ϕ_i into the variance due to the sensitive attribute and the variance due to the relevant non-sensitive attributes. The goal is to establish a technique that disregards the variance associated with the sensitive attribute and only considers the variance due to the rest of the attributes. This way, the metric will treat any pair of instances that only differ in the sensitive attribute equally. To demonstrate a possible choice of such a matrix, we will introduce a series of concepts:

Definition 4.3.8 (Orthogonal complement)**.** Let $(V, +, \cdot, K)$ be a vector space, and suppose that V is associated with the bilinear form $B : V \times V \to K$. The orthogonal complement of a subset $W \subseteq V$ is the set of all vectors which are orthogonal to each vector in W. It is represented as follows:

$$W^{\perp} = \{x \in V : B(x, y) = 0, \forall y \in W\} \tag{4.44}$$

Lemma 4.3.0.2 (Orthogonal complement as subspace)**.** *The orthogonal complement $W^{\perp}$ of $W \subseteq V$, with the bilinear product B is a vector subspace of V.*

Proof. We will test if every vector subspace definition conditions are met:

- **Closed under sum**: $\forall x, y \in W^{\perp}, x + y \in W^{\perp}$. Let us consider $x, y \in W^{\perp} \Leftrightarrow B(x, z) = 0 \wedge B(y, z) = 0, \forall z \in W \Rightarrow B(x + y, z) = B(x, z) + B(y, z) = 0, \forall z \in W$, using the basic properties of bilinear forms.
- **Closed under scalar product**: $\forall x \in W^{\perp}, k \in K, kx \in W^{\perp}$. As for every $x \in W^{\perp} \Leftrightarrow B(x, z) = 0, \forall z \in W$, using again the bilinear product basic properties, we can obtain that $\forall k \in K, B(kx, z) = kB(x, z) = 0, \forall z \in W$.

□

Lemma 4.3.0.3 (Vector space division using orthogonal complement)**.** *If W is a finite-dimensional vector subspace of V, then $V = W \oplus W^{\perp}$.*

Proof. Any vector $w \in W$ can be orthogonal to itself unless it is vector 0, by the definition of the bilinear form B. Therefore, $W \cap W^{\perp} = \{0\}$. Let us now check that $W + W^{\perp} = V$. Let $U = W + W^{\perp}$. We can build an orthonormal basis of U, and extend it to a basis for V. If $U \neq V$, then there exists at least one additional element e in the extended basis of V. Since e belongs to the basis of V, it is orthogonal to every other basis component. For this reason it is orthogonal to U. Since $W \subset U$, and e is orthogonal to $W \Rightarrow e \in W^{\perp}$. By the same reasoning,

$e \in W \Rightarrow e = 0$, which contradicts the assumption that e is a basis element of V, thereby leading to a contradiction. □

Definition 4.3.9 (Orthogonal projection)**.** Let W be a finite-dimensional subspace of V. Orthogonal projection of V onto W is the lineal operator $P_W \colon V \to V$, defined as follows: for each $v \in V$, there exist unique vectors $w \in W$ and $w' \in W^{\perp}$ such that $v = w + w'$. Then, $P_W(v) = w$.

A possible choice for matrix Σ is the projection matrix onto the orthogonal complement of the subspace generated by the columns of matrix A_*. This subspace is denoted as $V^c_{A_*}$. By doing so, if we consider two comparable elements, we obtain:

$$\begin{aligned} d_1(w, w') &= \langle \phi_1 - \phi_2, (I - P_{V^c_{A_*}})(\phi_1 - \phi_2) \rangle \\ &\approx \langle B_*(v_1 - v_2), (I - P_{V^c_{A_*}}) B_*(v_1 - v_2) \rangle \end{aligned} \tag{4.45}$$

This choice ignores differences between ϕ_1 and ϕ_2 due to variations in sensitive attributes. Although the matrix $V^c_{A_*}$ is inherently unknown, it can be estimated based on the learned representation and comparable sample groups. It is important to emphasize that the objective is $V^c_{A_*}$, not A_*, which simplifies the typical requirements of factor analysis. This enables the application of an estimation process through an appropiate factor analysis procedure.

4.3.2. EXPLORE Method: Embedded Xenial Pairs Logistic Regression

The EXPLORE method learns a fair pairwise comparison metric. More specifically, the data comes in the form of triplets that encapsulate expert knowledge: $(x_{i_1}, x_{i_2}, y_i)_{i=1}^n$, where the value $y_i \in \{0, 1\}$ indicates whether an expert considers the data comparable $(y_i = 1)$ or not $(y_i = 0)$. It is hypothesized that (x_{i_1}, x_{i_2}, y_i) follows a binary response model, where:

$$y_i | x_{i_1}, x_{i_2} \sim \text{Ber}(2\sigma(-d_i)) \tag{4.46}$$

Where $\sigma(x) = \frac{1}{1+e^{-x}}$ denotes the logistic function, discussed in Section 8.3, and d_i is defined as:

$$\langle \underbrace{(\phi_{i_1} - \phi_{i_2})(\phi_{i_1} - \phi_{i_2})^T}_{D_i}, \Sigma_0 \rangle \tag{4.47}$$

Where ϕ_{i_1} and ϕ_{i_2} are the learned representations of x_{i_1} and x_{i_2} respectively, and the matrix $\Sigma_0 \in S^d_+$ needs to be estimated. To achieve this, the following function is proposed:

$$l_n(\Sigma) = \frac{1}{n} \sum_{i=1}^{n} y_i \log \left(\frac{2\sigma(-\langle D_i, \Sigma \rangle)}{1 - \sigma(-\langle D_i, \Sigma \rangle)} \right) + \log(1 - \sigma(-\langle D_i, \Sigma \rangle)) \tag{4.48}$$

Since the function $l_n(\Sigma)$ is concave with respect to Σ, it is suggested to use an iterative optimization process, such as stochastic gradient descent, to find the Σ that maximizes it.

4.4. Counterfactual fairness

To specify counterfactual fairness, as outlined in [Kusner et al., 2018], we need to define our problem using a causal model. Based on these causal models, we can construct probabilities

and analyze how each element influences the system.

We will begin by defining what a causal model is [Galles and Pearl, 1998]:

Definition 4.4.1 (Causal model)**.** A causal model is a triplet, $M = \langle U, V, F \rangle$, where its components are:

- **Exogenous variables**: U is a set of variables known as "exogenous variables" or "unobservable variables", which are determined by factors external to the model.
- **Endogenous variables**: V is a set of variables known as "endogenous variables" or "observable variables", which are determined by variables within the model. We denote them as $V = \{V_1, \ldots, V_i\}$.
- **Structural equations**: F is a set of functions $\{f_1, \ldots, f_n\}$ known as "structural equations", that completely determine the value of variables V in terms of variables $U \cup (V \backslash V_i), \forall i \in \{1, \ldots, n\}$. They are often represented in the form $v_i = f_i(pa_i, u), \forall i \in \{1, \ldots, n\}$, where pa_i refers to the set of variables in $V \backslash V_i$ on which the value of V_i depends, known as the "parent variables of V_i".

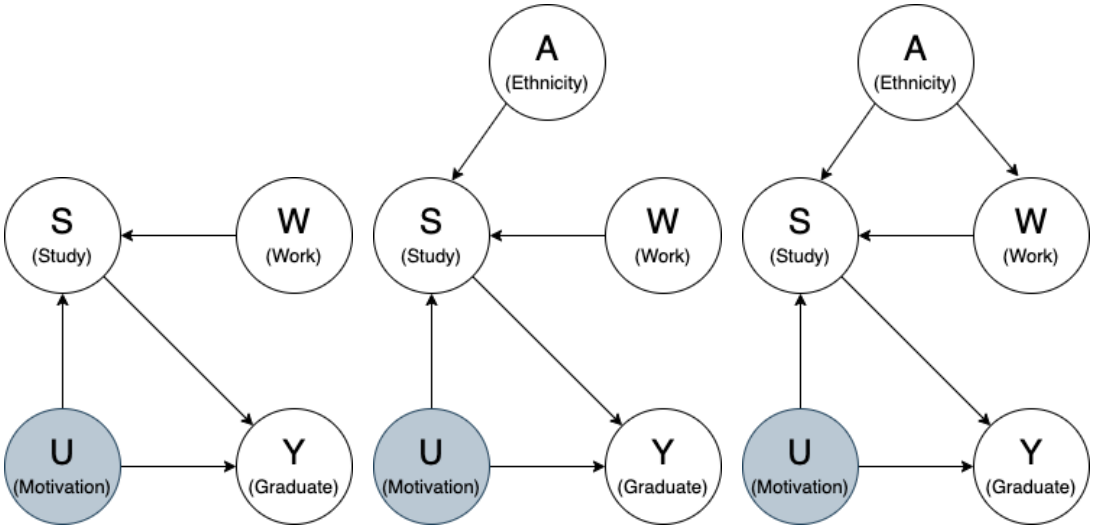
Definition 4.4.2 (Causal graph)**.** Every causal model M has an associated causal graph $G(M)$, which is a directed graph where the variables U, V are the nodes, and the functions F establish directed edges between them. For each f_i, directed edges are formed from all variables on which that function effectively depends to V_i. It can also be verified that this graph is always acyclic, since if it were not, it would contradict the first property of set F. Examples of this can be seen in Figure 4.1.

Definition 4.4.3 (Causal submodel)**.** Let M be a causal model, let $X \subset V$, and let x be a set of different values that X can take. We will define the causal submodel M_x as the causal model given by the following tern: $M_{X \leftarrow x} = \langle U, V, F_{X \leftarrow x} \rangle : F_{X \leftarrow x} = \{f_i : V_i \notin X\} \cup \{X = x\}$.

In other words, the functions f_i associated with variables $V_i \in X$ are removed, and they are replaced with constant functions that assign the corresponding value to each variable fixed in x. Additionally, it is necessary to establish the condition that these new structural equations must completely determine the value of the remaining free variables in V in terms of the variables U, as this cannot be guaranteed for any assignment x of any subset of variables X. A simple causal model illustrating this can be seen in Figure 4.2.

This assignment x to X is also known as an "intervention", and it is usually denoted by $X \leftarrow x$. Complex interventions can be constructed applying logical expressions to simple interventions, and their notation extends from the one previously discussed. The interventions mentioned so far can be interpreted as a conjunction of interventions with respect to each of the individual variables forming X, assigning them their specific value from x. Only in these cases, and if the set of variables X is understood, the notation can be simplified, allowing us to write M_x and F_x directly. We will only consider this last type of interventions.

Definition 4.4.4 (Effect of an intervention)**.** Let $M = \langle U, V, F \rangle$ be a causal model, X a set of variables in V, and x a specific realization of X. The effect of an intervention $X \leftarrow x$ on M is represented by the causal model M_x.



(a) Causal graph associated with a causal model where no protected attribute is considered. The unobservable variable is shown in a darker tone.

(b) Causal graph from Figure 4.1a including the a sensitive variable (Ethnicity) and a structural equation that relates it to the rest of the model.

(c) Causal graph from Figure 4.1b considering an additional structural equation.

Figure 4.1.: Three examples of causal graphs.

Definition 4.4.5 (Potential response to an intervention)**.** Let $M = \langle U, V, F \rangle$ be a causal model, where $Y \in V$ is an observable variable, and $X \subset V$. The potential response of variable X to a specific intervention $X \leftarrow x$ is denoted as $Y_{X \leftarrow x}(u)$, or simply $Y_x(u)$ when context allows. It represents the value that variable Y would take under the mentioned intervention and given a realization u of the unobservable variables U. Since M_x is a causal model, $Y_x(u)$ will take a unique value.

Definition 4.4.6 (Counterfactual)**.** Let $M = \langle U, V, F \rangle$ be a causal model, where $Y \in V$ be an observable variable, and $X \subset V$. The value that the variable Y would have taken under the intervention $X \leftarrow x$, also known as the "counterfactual value" in that context, is represented by the potential response value $Y_x(u)$.

Definition 4.4.7 (Probabilistic causal model)**.** A probabilistic causal model is a pair $\langle M, P[U = u] \rangle$, where M is a causal model and $P[U = u]$ is a probability measure over the domain of U.

Now, while the values of the observable variables are still determined by the values of the structural equations in F, we consider the unobservable variables as random variables for which we assign probabilities. This allows us to define probabilities over the observable variables as follows:

$$P[Y = y] = \sum_{\{u : Y(u) = y\}} P(u) \tag{4.49}$$

Here, $Y(u)$ indicates the value of variable Y based on the realization u of the unobservable

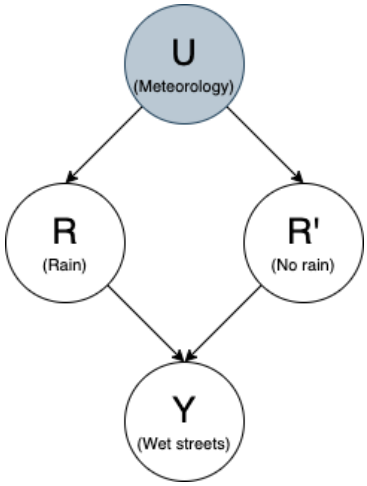


Figure 4.2.: For this causal model, if we consider $X = (L, L')$, the assignment $(0, 0)$, would not be valid, meaning "False" and "False". Consequently, the value of the variable Y cannot be uniquely determined.

variables. Similarly, we can define counterfactual probabilities in an analogous manner:

$$P[Y_x = y] = \sum_{\{u:Y_x(u)=y\}} P(u) \tag{4.50}$$

We can also define joint probabilities. For example, we can define probabilities $P[Y_x = y, X = x']$, and if $x \neq x'$, these probabilities are naturally defined as follows:

$$P[Y_x = y, X = x'] = \sum_{\{u:Y_x(u)=y \wedge X(u)=x'\}} P(u) \tag{4.51}$$

$$P[Y_x = y, Y_{x'} = y'] = \sum_{\{u:Y_x(u)=y \wedge Y_{x'}(u)=y'\}} P(u) \tag{4.52}$$

For calculating conditional probabilities, issues might arise if variables are assigned in an incompatible way with the conditioning variable. However, in [Pearl, 1999], a method for calculating these conditional probabilities is demonstrated as follows:

$$\begin{aligned} P[Y_{x'} = y' | X = x, Y = y] &= \frac{P[Y_{x'} = y', X = x, Y = y]}{P[X = x, Y = y]} \\ &= \sum_u P[Y_{x'}(u) = y'] P[u = U | x = X, y = Y] \end{aligned} \tag{4.53}$$

In this manner, for each possible realization that the unobservable variables u can take, we calculate their conditional probability given that $x = X, y = Y$, and then multiply it by the probability that variable Y takes the value y' according to the model $M_{x'}$.

In order to define the concept of counterfactual fairness, we must consider our variables X, A as observable variables in the model. Additionally, our predictor will be a function of these variables, so we can include it as an observable variable, thereby $X \cup A \cup p = V$.

Definition 4.4.8 (Counterfactual fairness)**.** Let $M = \langle U, V, F \rangle$ be a causal model under the conditions previously described. The predictor p is said to satisfy the counterfactual fairness condition if:

$$P[p_{A \leftarrow a}(U) | X = x, A = a] = P[p_{A \leftarrow a'}(U) | X = x, A = a] \tag{4.54}$$

For each possible value y of Y and each possible value a' of A.

This condition can be interpreted such that the value of A should not be the cause of the value of p for any given instance. In other words, changing the value of A while keeping the rest of the values of variables that do not directly depend on the value of A should not alter the probability distribution of p.

The main advantage of using this fairness measure is that it proposes a way to understand the causes of possible bias through the causal graph. Thus, by changing the value of the attribute A, we can observe how the rest of the variables that depend on its value are affected. For this reason, it eliminates all possible problems of fairness due to unawareness. Furthermore, unlike individual fairness, it does not need the introduction of additional metrics.

However, the main problem of using this model lies in the definition of the causal model. Indeed, the model gives us an idea of how the rest of the variables are influenced by fixing a certain value of A, but this influence, based on the model, is determined by the model we consider, which may not be equal to the underlying real model. Constructing a correct causal model requires an in-depth knowledge of the context of the problem and a clear understanding of the dependencies between variables to define the structural relations, which is generally challenging. Therefore, although the idea is theoretically robust, in practice, it will have a more limited application than other fairness criteria. However, if the causal model is known with certainty, it could be the best fairness measure among all the proposed ones.

5. Multiobjective optimization

In this chapter, the mathematical theory supporting multiobjective optimization and Many-objectives optimization will be studied. Concepts such as Pareto dominance and quality indicators for approximation sets will be discussed.

5.1. Multiobjective optimization problems

We will begin by defining what a multiobjective optimization (MOO) problem is:

Definition 5.1.1 (Multiobjective optimization problem (MOP))**.** . Let $\Omega \subset \mathbb{R}^n, \Theta \subset \mathbb{R}^m, m > 1$. Let $f\colon \Omega \to \Theta$, with $f(\omega) = (f_1(\omega), \ldots, f_m(\omega)), \forall \omega \in \Omega$, and each $f_i\colon \Omega \to \mathbb{R}, \forall i \in \{1, \ldots, m\}$. A multiobjective optimization problem consists of:

$$\min_{\omega \in \Omega} \text{ or } \max_{\omega \in \Omega} (f_1(\omega), \ldots, f_m(\omega)) \tag{5.1}$$

Depending on whether we want to minimize or maximize, we refer to a multiobjective minimization or maximization problem, respectively. In our context, each ω will be a binary classifier p, and each objective function f_i will be a measure to evaluate the performance of classifiers (some examples of these measures are discussed in Section 9.4).

This definition can lead to ambiguity, as there is not a specified order relation in $\Theta \subset \mathbb{R}^m$ when $m > 1$ which would allow us to compare the image of the function f. We cannot directly induce the usual order relation in $\mathbb{R}$ that enables us to determine maximum and minimum. The comparison criterion from which we can define maximum and minimum will be later discussed.

Definition 5.1.2 (Decision space or search space)**.** The decision space or search space is the set $\Omega \subset \mathbb{R}^n$ used in Definition 5.1.1.

Definition 5.1.3 (Objective space)**.** The objective space is the set $\Theta \subset \mathbb{R}^m, m > 1$ used in Definition 5.1.1.

Definition 5.1.4 (Dimension of a multiobjective optimization problem)**.** The dimension of a multiobjective optimization problem is the value $m > 1$ used in Definition 5.1.1.

Definition 5.1.5 (Objective functions)**.** An objective function is each $f_i : \Omega \to \mathbb{R}, i \in \{1, \ldots, m\}$ used in Definition 5.1.1. Following this notation, the objective functions of a multiobjective optimization problem are $\{f_1, \ldots, f_m\}$, also referred to as "objectives".

5.2. Dominance of solutions in multiobjective optimization problems

We need a comparison method for images of the objective function f, as stated before, in order to determine the optimal value we are searching for, which can be a maximum or minimum.

We will now present this criterion, and discuss concepts related to optimal solutions in a multiobjective optimization problem.

Definition 5.2.1 (Feasible solution of a multiobjective optimization problem)**.** A feasible solution of a multiobjective optimization problem, or simply a solution, is an element $\omega \in \Omega$.

The criterion of solution dominance takes into account that the image by the objective function of each solution lies in $\mathbb{R}^m$ where $m > 1$. In multiobjective optimization problems, there may be some solutions that are better in some objectives but worse in others. This occurs in multiobjective problems of interest, where the objectives are conflicting. That is, although we may improve all objective functions uniformly for a while, there will come a point where if we want to continue improving a particular objective or set of objectives, there must necessarily be a tradeoff in the rest of the objectives, which will worsen in our optimization. Examples could include maximizing user comfort and energy savings in a heating system simultaneously, or simultaneous minimization in a production chain of unit time in the chain, unit cost, and product variability with respect to the quality standard.

In order to continue defining concepts, we will need to focus on either minimization or maximization problems. In this case we will consider minimization problems, as the practical study undertaken works with minimization problems. Definitions for maximization problems are analogous.

Definition 5.2.2 (Pareto dominance)**.** Let $\psi, \omega \in \Omega$ be solutions to a multiobjective minimization problem. ψ Pareto dominates ω, or simply ψ dominates ω, denoted as $\psi \prec \omega$, if and only if the following properties are met:

1. $f_i(\psi) \leq f_i(\omega), \forall i \in \{1, \ldots, m\}$.
2. $\exists j \in \{1, \ldots, m\} : f_j(\psi) < f_j(\omega)$.

Definition 5.2.3 (Order relation)**.** A binary relation R over a set X is an order relation if and only if the following properties are met:

- **Reflexivity**: $aRa, \forall a \in X$.
- **Antisymmetry**: $aRb, bRa \Rightarrow a = b, \forall a, b \in X$.
- **Transitivity**: $aRb, bRc \Rightarrow aRc, \forall a, b, c \in X$.

Lemma 5.2.0.1 (No ordering in Pareto dominance)**.** *Pareto dominance relation is not an order relation.*

Proof. It is an irreflexive relation. By contradiction, if $\exists \omega \in \Omega : \omega \prec \omega \Rightarrow \exists j \in 1, \ldots m : f_j(\omega) < f_j(\omega)$, which is a contradiction. □

As Pareto dominance is not an order relationship, we cannot define a partially ordered set from which to obtain lower bounds or a minimum. In fact, even if we were to eliminate the second condition in the definition of Pareto dominance, we could not guarantee that this relationship would be an order relationship. This is because, even if we achieved reflexivity, injectivity in each function f_i would also be necessary to ensure antisymmetry. What we will do is directly provide a definition with a notion equivalent to the notion of lower bound, which will allow us to precisely define our optimization goals.

Definition 5.2.4 (Pareto-optimal solution)**.** Let $\omega \in \Omega$ be a solution of a multiobjective minimization problem. ω is a Pareto-optimal solution $\Leftrightarrow \forall \omega' \in \Omega, \omega' \nprec \omega$

If a solution ω is Pareto-optimal, it means that there is no other solution that is strictly better in at least one objective without worsening in at least another objective with respect to ω. Therefore, these solutions are not improvable in all objectives and represent a limit in our optimization. These solutions are not comparable, as if ω, ω' are Pareto-optimal solutions, then $\omega' \nprec \omega$, but also $\omega \nprec \omega'$, so there can be multiple such solutions in our search space. This is the reason why our goal is to find the greatest set of distinct Pareto-optimal solutions. We will consider these as the best solutions, and they will be our ideal optimization goal, allowing the end users to choose a preferred solution based on their criteria.

Definition 5.2.5 (Pareto-optimal set)**.** The Pareto-optimal set ($\mathcal{P}$) is the set of Pareto-optimal solutions, $\mathcal{P} = \{\omega \in \Omega : \forall \omega' \in \Omega, \omega' \nprec \omega\}$

Definition 5.2.6 (Pareto front)**.** The Pareto front ($\mathcal{PF}$) is the set of images of the elements from the Pareto-optimal set under f: $\mathcal{PF} = \{f(\omega) : \omega \in \mathcal{P}\}$

By studying the Pareto front, we can observe and understand how the objectives relate to each other, as well as how the objectives of these solutions behave, providing us with a criterion for choosing among them. Furthermore, $|\mathcal{PF}| \leq |\mathcal{P}|$ since there may be several solutions in the Pareto-optimal set that have the same objective values (f may not be an injective function). In practice, we will be more interested in finding the entire Pareto front than in finding the entire Pareto-optimal set. This is because we want to know how far can we go in joint optimization, rather than all Pareto-optimal solutions even if they give us repeated objective function values; one solution for each objective function value could be enough.

Finding the entire Pareto-optimal set is computationally expensive, as to identify each element of the Pareto-optimal set, we would need to compare it with the rest of the elements in Ω, which is impractical. Therefore, we will introduce the following concepts:

Definition 5.2.7 (Approximation set)**.** An approximation set is a set $A \subseteq \Omega : \forall a_1, a_2 \in A$, with $a_1 \neq a_2 \Rightarrow a_1 \nprec a_2 \wedge a_2 \nprec a_1$.

Definition 5.2.8 (Pareto dominance in approximation sets)**.** Let $A, B \subseteq \Omega$ be approximation sets. A dominates B, denoted as $A \prec B \Leftrightarrow \forall a \in A \exists b \in B : a \prec b$.

Lemma 5.2.0.2 (Pareto dominant subset)**.** *For any non-empty finite subset $B \subseteq \Omega$, there exists a non-empty approximation set $A \subseteq B$ such that $A \prec B \backslash A$, in case $A \neq B$.*

Proof. $A = \{a \in B : \nexists b \in B : b \prec a\}$. A is an approximation set, as for any two distinct elements $a_1, a_2 \in A, a_1 \neq a_2 \Rightarrow a_1 \nprec a_2 \wedge a_2 \nprec a_1$, by the definition of A, and for that reason A is an approximation set. A is non-empty, as in any other case, $\forall b \in B, \exists b' \in B : b' \prec b$, which is impossible. By contradiction, let us suppose A is empty. Let $b_0 \in B$ be a fixed element of B. Then, $\exists b_1 \in B : b_1 \prec b_0$. That $b_1 \neq b_0$, as $\prec$ is irreflexive. For that b_1 there is another $b_2 \in B : b_2 \prec b_1 \prec b_0$. $b_2 \neq b_1$, using the same reasoning as before, and $b_2 \neq b_0$ because $b_2 \prec b_0$ using transitivity, and we can apply again that $\prec$ is irreflexive. Iterating this process we can create a chain $b_{|B|} \prec \cdots \prec b_0$, being all distinct elements, and therefore every element in B is present. But in that case, by definition of A, $b_{|B|} \in A$, as if $b_{|B|} \notin A \Rightarrow \exists b \in B : b \prec b_{|B|}$. But because of how was $b_{|B|}$ chosen, $b_{|B|} \prec b$, and that would contradict that $\prec$ is irreflexive.

To proof that $A \prec B \backslash A$, in case $A \neq B$, let us see that $\forall b \in B \backslash A, \exists a \in A : a \prec b$. To do that, the following iterative procedure is considered: Let us suppose that there is an element $b \in B \backslash A$. By definition of A, we know that $\exists b_0 \in B : b_0 \prec b$. If $b_0 \in A$, we have finished. In any other case, $b_0 \in B \backslash A$, and for that reason $\exists b_1 \in B : b_1 \prec b_0$. We can repeat this procedure as much as needed. As B is a finite set, this procedure has to end using a finite amount of steps. □

We cannot generalize Lemma 5.2.0.2 for infinite B subsets. We can find a simple counterexample, using the multiobjective minimization problem $f : \mathbb{R}^2 \to \mathbb{R}^2$, where f is the identity function. In this case, we could choose an infinite set B only containing every element of the sequence $\{x_n\}_{n \in \mathbb{N}} : x_n = (\frac{1}{2^n}, \frac{1}{2^n})$. Given a fixed $m \in \mathbb{N}, x_r \prec x_m, \forall r > m$. But the set $\{r \in \mathbb{N} : r > m\}$ is an infinite set, as $\mathbb{N}$ is a non-majorized infinite set. So B is a set in which each element has an infinite set of elements that dominates it.

Definition 5.2.9 (Front of level 0)**.** Given a non-empty finite set $B \subseteq \Omega$, the front of level 0 of B, denoted as B_0, is the set $\{f(a) : a \in A\}$, where A is the set found in Lemma 5.2.0.2 .

Using this definition, if B is an approximation set, then $B_0 = B$. Once B_0 is found from set B, and if $B \backslash B_0$ is not empty, we can repeat the same process used in the proof of Lemma 5.2.0.2 with the $B \backslash B_0$ subset. We can repeat all this steps iteratively, and as B is finite, we will end reaching an empty set in a finite amount of steps. Having observed this, the following definition is natural:

Definition 5.2.10 (Front of level n)**.** Given a non-empty finite subset $B \subseteq \Omega$, the front of level n of B, which is written as B_n, is the set B_0 with respect to the set $B \cup_{i=0}^{n-1} B_i$, in case it is a non-empty set.

Definition 5.2.11 (Optimization goal in a MOP)**.** The optimization goal in a MOP will be to find an approximation set $A \subset \Omega$ meeting the following conditions:

- **Convergence**: Every $a \in A$ has to be as close as possible to $\mathcal{P}$.
- **Spread**: Every $a \in A$ has to be as diverse as possible within the objective space.

This definition is very ambiguous. To begin with, the reason why we want to solve a multiobjective optimization problem is that we do not know the set $\mathcal{P}$, and there is no actual way to measure how close we are to $\mathcal{P}$ without knowing it. We could understand closeness in terms of dominance, so "as close as possible to $\mathcal{P}$" could mean having the least amount of other solutions that dominates them. Or it could involve calculating real distances between elements of $\mathcal{P}$ and elements of A.

But our interest not only relies on convergence, but also on how well distributed are with respect to the objective space. The distribution has to be as spread and uniform as possible, and describe the best as possible the actual shape of $\mathcal{P}$. There are lots of different options to measure uniformity and spread, but again we do not know the set $\mathcal{P}$.

So it is important to define a set of other quality indicators in order to better evaluate the approximation sets found, which will be the aim of the following section.

5.3. Quality indicators for solution sets

Definition 5.3.1 (Quality indicator)**.** A k-ary quality indicator is a function $Q\colon \mathcal{P}(\Omega)^k \to \mathbb{R}$.

Currently, there are a large number of quality indicators, some of which are outlined in [Li et al., 2014, Audet et al., 2020, Van Veldhuizen, 1999]. Quality indicators can be classified based on the aspect they attempt to measure on the given sets in their arguments. We can make the following classification, which is not mutually exclusive:

- **Cardinality indicators**: Measure the cardinality of the set with respect to a certain feature.
- **Convergence indicators**: Measure convergence to the Pareto-optimal set.
- **Uniformity indicators**: Measure the uniformity in the distribution of the set's elements.
- **Diversity indicators**: Measure diversity of the solutions within the objective space.

For the calculation of some of these indicators, knowledge of the Pareto-optimal set $\mathcal{P}$ is assumed. Since, in most cases, and particularly in our study, we do not know it, we will treat such k-ary quality measures as (k+1)-ary measures. That last parameter will be a known set that approximates $\mathcal{P}$. When this happens, we will denote the last parameter as $\mathcal{P}_{ref}$ to indicate that this approximation is used.

Let us now look at the definitions of some of the most common quality indicators, which we will be used in our analysis:

Definition 5.3.2 (Hypervolume or $\mathcal{S}$-metric)**.** Without loss of generality, let us consider a multiobjective minimization problem with $f\colon \Omega \to (0,1)^m$. Hypervolume is a unary quality indicator, $\mathcal{H}\colon \mathcal{P}(\Omega) \to (0,1)$, such that for every solution set $A \subseteq \Omega$:

$$\mathcal{H}(A) = \int_{(0,1)^m} \mathbb{1}_{D_A}(z)dz \tag{5.2}$$

Being $D_A = \{z = (z_1, \ldots, z_m) \in (0,1)^m : \exists a \in A : a \prec z\}$ and $\mathbb{1}_{D_A}$ the characteristic function of the set D_A.

Hypervolume measures the quality of a given set with respect to the convergence, diversity, and uniformity of its solutions. The main problem with this measure is that its value heavily depends on the shape of the Pareto front, which can favor certain solutions over others. In terms of all these aspects, we can show examples where, despite having a more diverse, uniformly distributed or approximate set to the Pareto-optimal set, we have a lower hypervolume, although this generally does not occur. Regarding convergence, however, we have a monotonicity property which almost any other indicator with such good properties has, among those widely used nowadays:

Lemma 5.3.0.1 (Monotonicity of hypervolume)**.** *Let $A, B \subseteq \Omega$ be two finite approximation sets. If $A \prec B$, and $\mathcal{H}(A) = \mathcal{H}(B) = 0$ does not happen, then $\mathcal{H}(A) > \mathcal{H}(B)$.*

Proof. As $A \prec B \Rightarrow \forall b \in B, \exists a \in A : a \prec b$. Therefore, $\forall z \in (0,1)^m : \mathbb{1}_{D_B}(z) = 1 \Rightarrow \exists b \in B : b \prec z$, and as $\exists a \in A : a \prec b \Rightarrow \mathbb{1}_{D_A}(z) = 1$, then $\mathcal{H}(A) \geq \mathcal{H}(B)$. We will now proof that $\mathcal{H}(\mathcal{A}) \gneq \mathcal{H}(B)$. To achieve this, we will find a measurable non-null set $\mathcal{S}$ which will be

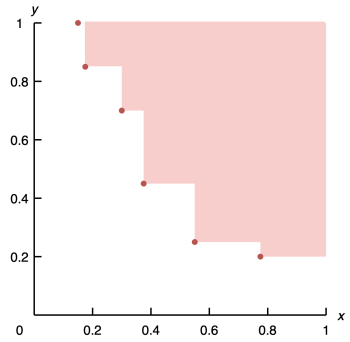


Figure 5.1.: 2D example of the hypervolume for the images of a given solution set, represented using dots.

contained in D_A but not in D_B. Let us choose any $b \in B$, and its respective $a \in A : a \prec b$, which exists as $A \prec B$. Now, $\forall i \in \{1, \ldots, m\}$, if $f_i(b) \neq \max\{f_i(b') \in \mathbb{R} : b' \in B\}$ (that maximum exists $\forall i \in \{1, \ldots, m\}$ since B is a finite set), then there exists at least one element, which will be denoted as $b^i \in B$, which meets: $f_i(b^i) = \min\{f_i(b') \in \mathbb{R} : b' \in B \wedge f_i(b') > f_i(b)\}$. Since $a \prec b \Rightarrow f_i(a) \leq f_i(b), \forall i \in \{1, \ldots, m\}$. We will now introduce a set of elements $\{d_1, \ldots, d_m\}$ where:

$$d_i = \begin{cases} f_i(b) - f_i(a) & \text{if } f_i(b) - f_i(a) > 0 \\ f_i(b^i) - f_i(a) & \text{if } f_i(b) - f_i(a) = 0 \wedge \exists b^i \\ 1 - f_i(a) & \text{if } f_i(b) - f_i(a) = 0 \wedge \nexists b^i \end{cases} \tag{5.3}$$

Each of these d_i is always strictly greater than 0 unless $f_i(b) - f_i(a) = 0 \wedge f_i(a) = f_i(b) = 1$, and consequently the hypervolume contribution of both a and b will be 0. In that case, we have to select another point. If all points meet that condition, then $\mathcal{H}(A) = \mathcal{H}(B) = 0$, this being the only exception.

If that is not the case, we select a solution b for which this does not happen. Then, $\forall z = (z_1 \ldots z_m) \in \mathcal{S} = (f_1(a), f_1(a) + d_1) \times \cdots \times (f_m(a), f_m(a) + d_m), \mathbb{1}_{D_A}(z) = 1 \wedge \mathbb{1}_{D_B}(z) = 0$. Indeed, $\mathbb{1}_{D_A}(z) = 1$ since $z_i > f_i(a), \forall i \in \{1, \ldots, m\}$ and $a \in A$. We will prove that $\mathbb{1}_{D_B}(z) = 0$ by contradiction. Let us suppose that $\mathbb{1}_{D_B}(z) = 1 \Rightarrow \exists \tilde{b} \in B : f_i(\tilde{b}) \leq z_i, \forall i \in \{1, \ldots m\} \wedge \exists j \in \{1, \ldots, m\} : f_j(\tilde{b}) < z_j$.

If d_i was assigned using the third case ($\nexists i \in \{1, \ldots, m\} : f_i(b) - f_i(a) = 0 \wedge \nexists b^i$), we have to observe that $a \prec b \Rightarrow \exists j \in \{1, \ldots, m\} : f_j(a) < f_j(b)$. Therefore, $z_j \in (f_j(a), f_j(b)) = (f_j(a), f_j(a) + d_j) \Rightarrow z_j < f_j(b)$, and in the case that $\tilde{b}$ exists, for it to be $f_j(\tilde{b}) \leq z_j$, at least $f_j(\tilde{b}) < f_j(b)$ needs to happen. For the rest of objectives, depending on how was their respective d_i defined:

- If $d_i = f_i(b) - f_i(a)$, then we can follow a similar reasoning and $f_i(\tilde{b}) < f_i(b)$.
- If $d_i = f_i(b^i) - f_i(a)$, following an analogous reasoning, then we obtain that $f_i(\tilde{b}) < f_i(b^i)$, but in this case, using the definition of b^i, and for $\tilde{b} \in B$, it needs to happen that $f_i(\tilde{b}) \leq f_i(b)$.

- If $d_i = 1 - f_i(a)$ then $f_i(b) - f_i(a) = 0 \wedge \nexists b^i \Rightarrow f_i(b) = f_i(a) \wedge f_i(b) = \max\{f_i(b') : b' \in B\}$. Because of this reasoning and for $\tilde{b} \in B$, it needs to happen that $f_i(\tilde{b}) = f_i(b)$.

Consequently, $\tilde{b}$ needs to meet that $\forall i \in \{1, \ldots, m\}, f_i(\tilde{b}) \leq f_i(b) \wedge \exists j \in \{1, \ldots, m\} : f_j(\tilde{b}) < f_j(b) \Rightarrow \tilde{b} \prec b$, which contradicts that B is an approximation set.

To summarize, it has been proven that $\mathbb{1}_{D_B}(z) = 0$. Therefore it is immediate to deduce that $\mathcal{S} \subseteq D_A \backslash D_B$. Additionally, as $\mathcal{S} = (f_1(a), f_1(a) + d_1) \times \cdots \times (f_m(a), f_m(a) + d_m)$, and each $d_i > 0$, the following is met:

$$\begin{aligned} \mathcal{H}(A) - \mathcal{H}(B) &= \int_{(0,1)^m} \mathbb{1}_{D_A}(z)dz - \int_{(0,1)^m} \mathbb{1}_{D_B}(z)dz = \int_{(0,1)^m} \mathbb{1}_{D_A}(z) - \mathbb{1}_{D_B}(z)dz \\ &= \int_{(0,1)^m} \mathbb{1}_{D_A \backslash D_B}(z)dz \geq \int_{(0,1)^m} \mathbb{1}_{\mathcal{S}}(z)dz = \prod_{i=1}^{n} d_i > 0 \Rightarrow \mathcal{H}(A) > \mathcal{H}(B) \end{aligned} \tag{5.4}$$

□

We can see some images that can help us understand the procedure carried out in the proof in Figure 5.2. Additionally, we can observe the case where $\mathcal{H}(A) = \mathcal{H}(B) = 0$ in Figure 5.3.

The reciprocal of this result is not true, and can be proven taking $A = (B \cup \{a\})_0, a \in \Omega$, such that A is another approximation set.

Definition 5.3.3 (Error ratio)**.** Error ratio is a binary quality indicator. $ER\colon \mathcal{P}(\Omega)^2 \to [0, 1]$ such that:

$$ER(A, \mathcal{P}_{ref}) = 1 - \frac{|\{a \in A : a \in \mathcal{P}_{ref}\}|}{|A|} \tag{5.5}$$

This is the proportion of elements from A which do not belong to $\mathcal{P}_{ref}$. The lower this quality indicator is, the more solutions in A will also be in $\mathcal{P}_{ref}$, which is preferable.

It could happen that there are elements in our set which are solutions belonging $\mathcal{P}_{ref}$, but due to rounding errors, they are not considered equal. For that reason we will introduce a margin of error ϵ when comparing both sets. Therefore, the actual metric is calculated as: $ER(A, \mathcal{P}_{ref}) = 1 - \frac{|\{a \in A : \exists b \in \mathcal{P}_{ref} : ||a-b|| < \epsilon\}|}{|A|}$.

Definition 5.3.4 (Proportion)**.** Proportion is a binary quality indicator, $PROP\colon \mathcal{P}(\Omega)^2 \to [0, 1]$ such that:

$$PROP(A, \mathcal{P}_{ref}) = \frac{|\{a \in \mathcal{P}_{ref} : a \in A\}|}{|\mathcal{P}_{ref}|} \tag{5.6}$$

This is the proportion of elements from $\mathcal{P}_{ref}$ that also belong to A. The greater this quality indicator is, the more elements from $\mathcal{P}_{ref}$ are also in A, which is preferable.

We can relax this metric in exactly the same way as we did with the error ratio, by adding a constant error margin ϵ.

Definition 5.3.5 (Generational distance)**.** Generational distance is a binary quality indicator, $GD\colon \mathcal{P}(\Omega)^2 \to \mathbb{R}_0^+$, such that:

$$GD(A, \mathcal{P}_{ref}) = \frac{\left(\sum_{i=0}^{|A|} d_i^t\right)^{1/t}}{|A|} \tag{5.7}$$

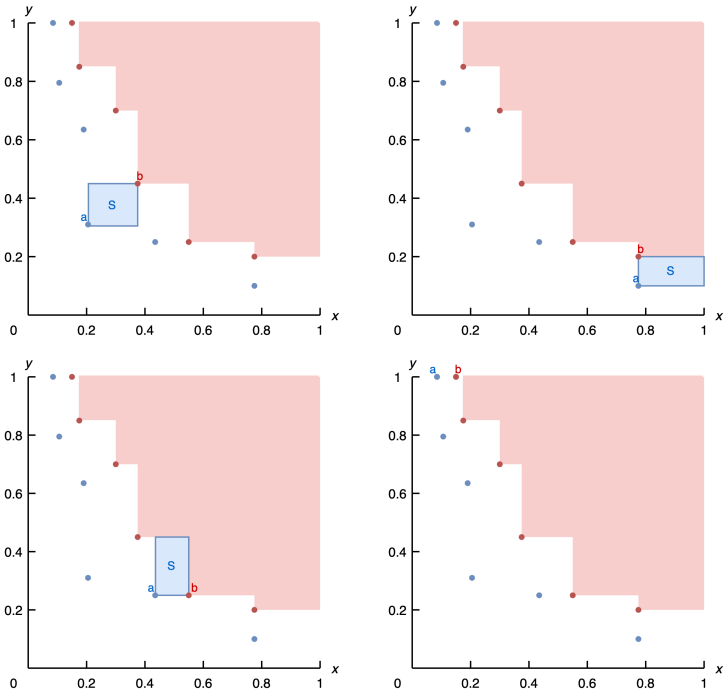


Figure 5.2.: Building of set S used in the proof of Lemma 5.3.0.1, depending on the chosen solution b, and its associated solution a (identified by their objectives). Red solutions belong to B, while blue solutions belong to A. There are four possible scenarios, where only the last one forces us to choose another solution b.

Where t is a fixed integer and $d_i = \min\{d(f(x_i), f(y)) : y \in \mathcal{P}_{ref}\}$ for a given $x_i \in A$. d can be any euclidean distance, but for our purpose, we will consider the distance associated with the euclidean norm in $\mathbb{R}^m$.

The larger t is, the lower generational distance will be. The absolute difference between two solutions will be lower, whereas relative difference will increase. Usual values of t are 1 and 2. We will consider the value $t = 2$, resulting in the following metric:

$$GD(A, \mathcal{P}_{ref}) = \frac{\left(\sum_{i=0}^{|A|} \min_{y \in A} ||f(x_i) - f(y)||_2^2\right)^{1/2}}{|A|} \tag{5.8}$$

Generational distance is a convergence indicator. The lower generational distance is, the higher the convergence. It is a simple, direct and widely used indicator in the field of multiobjective optimization.

Definition 5.3.6 (Inverted generational distance)**.** Inverted generational distance is a binary quality indicator, $IGD\colon \mathcal{P}(\Omega)^2 \to \mathbb{R}_0^+$, such that:

$$IGD(A, \mathcal{P}_{ref}) = \frac{\left(\sum_{i=0}^{|\mathcal{P}_{ref}|} d_i^t\right)^{1/t}}{|\mathcal{P}_{ref}|} \tag{5.9}$$

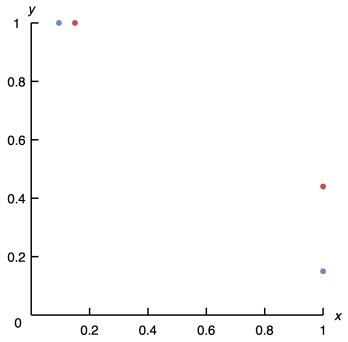


Figure 5.3.: Example of B (red solutions) and A (blue solutions) sets where $A \prec B$ and $\mathcal{H}(A) = \mathcal{H}(B) = 0$.

Where t is a fixed integer and $d_i = \min\{d(f(x_i), f(y)) : y \in A\}$ for a given $x_i \in \mathcal{P}_{ref}$. d can be any euclidean distance, but for our purpose, we will consider the distance associated with the euclidean norm in $\mathbb{R}^m$. Just the same as before, we will choose $t = 2$, and for that reason the final metric will be:

$$IGD(A, \mathcal{P}_{ref}) = \frac{\left(\sum_{i=0}^{|\mathcal{P}_{ref}|} ||f(x_i) - f(y)||_2^2\right)^{1/2}}{|\mathcal{P}_{ref}|} \tag{5.10}$$

Inverted generational distance is also a convergence indicator, where smaller values are better, but it takes a different approach compared to generational distance. While the latter evaluates only convergence, this measure is sensitive to both convergence and the representativeness of solutions from A with respect to $\mathcal{P}_{ref}$. If there is a region in $\mathcal{P}_{ref}$ that has not been explored by the solutions in A, inverted generational distance would increase, whereas generational distance might not necessarily do so.

Definition 5.3.7 (Maximum spread)**.** Maximum spread is an unary quality indicator, $MS : \mathcal{P}(\Omega) \to \mathbb{R}_0^+$, such that:

$$MS(A) = \sqrt{\sum_{i=0}^{m} \max_{a,a' \in A} (a_i - a'_i)^2} \tag{5.11}$$

Maximum spread is a diversity indicator. The higher it is, the more distant values have been found, so a wider portion of the objective space has been explored.

In order to define the next quality indicator, named overall Pareto spread, we will introduce some preliminary definitions:

Definition 5.3.8 (Ideal point)**.** Given a multiobjective optimization problem and let $f = (f_1, \ldots f_m)$ its objective function, the ideal point, denoted as F^I, is the point having the following coordinates: $F_i^I = \min_{x \in \Omega} f_i(x)$. This is the solution to the single objective minimization problem considering only f_i as the objective function and ignoring the others.

If we want to find F^I, then m single objective minimization problems have to be solved. However, this is not always feasible, as these problems can be challenging due to the complexity of either Ω, f_i, or both. In most cases, a approach to obtaining approximate solutions

is through the use of metaheuristics. Nevertheless, such approximations may not meet the required precision. The execution time to solve these problems is typically unacceptable.

Because of all these reasons, the practical approach to using F^I involves utilizing approximations instead of the real value of F^I. The next property will help us to define a good approximation criterion. Let $C^I = \{x \in \Omega : \exists i \in \{1, \dots m\} : f_i(x) \leq f_i(x'), \forall x' \in \Omega\} = \{x \in \Omega : \exists i \in \{1, \dots m\} : f_i(x) = F_i^I\}$. With this set, we can make the following observation:

Lemma 5.3.0.2 (Single minimization solutions and Pareto-optimal set)**.** *The set* $C_0^I \subseteq \mathcal{P}$

Proof. Let $p \in C_0^I$. As $p \in C^I \supseteq C_0^I \Rightarrow \exists i \in \{1, \dots m\} : p_i \leq \omega_i, \forall \omega \in \Omega$. Additionally, since $p \in C_0^I$ it is not dominated by any other element from C^I. Furthermore, is it not dominated by any other element from $\Omega \backslash C^I$, as all elements from this set have strictly greater values in coordinate i compared to p_i. This implies that p is not dominated by any element in $(\Omega \backslash C^I) \cup C^I = \Omega \Rightarrow p \in \mathcal{P}$. □

Thanks to the previous lemma, if we had the real $\mathcal{P}$, then the calculation of F^I would be trivial. Using this information, we can approximate F^I by obtaining an approximation set A as the solution of the problem. The approximation, denoted as F^{I*}, would have the following coordinates: $F_i^{I*} = \min_{x \in A} f_i(x)$. If the approximation set is good enough, then F^{I*} will be as well.

Definition 5.3.9 (Nadir point)**.** Given a multiobjective optimization problem, and let $f = (f_1, \dots f_m)$ be its objective function, the nadir point, denoted as F^N, is the point having the following coordinates: $F_i^N = \max_{x \in \Omega} f_i(x)$. This is the solution to the single objective maximization problem considering only f_i as the objective function and ignoring the others.

The nadir point tells us the worst possible value that a specific objective can reach, without considering the rest of the objectives. Similar to the ideal point, finding it involves solving m single objective optimization problems, which is generally not feasible. This time, we do not have a clear way to approximate this point using the final approximation set, as the solutions it provides are optimized for minimization. Various methods for approximating the nadir point were tested, but in the end, a method was chosen that introduces a relatively low tradeoff with respect to total program execution time. This method calculates the nadir point as the point where each coordinate is equal to the maximum value for that coordinate among all individuals used to calculate it, which in practice will be the same individuals used to calculate the ideal point. We will denote the approximation of the nadir point as F^{N*}.

Definition 5.3.10 (Overall Pareto spread)**.** Overall Pareto spread is an unary quality indicator, $OS \colon \mathcal{P}(\Omega)^2 \to [0, 1]$ such that:

$$OS(A) = \prod_{i=0}^{m} \frac{|\max_{x \in A} f_i(x) - \min_{x \in A} f_i(x)|}{|F_i^N - F_i^I|} \tag{5.12}$$

In order to use this quality indicator, we need to know both ideal and nadir points, which are not easy to calculate as discussed before, so the quality indicator which will be used in practice will use the prior mentioned approximations, resulting in the following final expression:

$$OS(A) = \prod_{i=0}^{m} \frac{|\max_{x \in A} f_i(x) - \min_{x \in A} f_i(x)|}{|F_i^{N*} - F_i^{I*}|} \tag{5.13}$$

Definition 5.3.11 (Spacing)**.** Spacing is an unary quality indicator, $SP\colon \mathcal{P}(\Omega) \to [0,1]$ such that:

$$SP(A) = \sqrt{\frac{1}{|A|-1}\sum_{i=1}^{|A|}(\overline{d} - d_i)^2} \tag{5.14}$$

Where $d_i = \min_{x_j \in A, x_j \neq x_i}\{||f(x_i) - f(x_j)||_1\}$, and being $\overline{d} = \frac{\sum_{i=0}^{|A|} d_i}{|A|}$.

This quality indicator is a uniformity indicator. The larger it is, the greater the average separation distance between each individual and its nearest neighbor, which is preferable up to a point. However, if it becomes too large, the solutions will be more scattered, exploring less zones of the objective space. If $|A| = 1$ this quality indicator is not defined. In practice we will assign it a value of 0 in this case.

Definition 5.3.12 (Coverage)**.** Coverage is a binary quality indicator, $COV\colon \mathcal{P}(\Omega)^2 \to [0,1]$ such that:

$$COV(A,B) = \frac{|\{b \in B : \exists a \in A : a \prec b\}|}{|B|} \tag{5.15}$$

The greater the coverage of a set A over another set B, the more solutions in A dominate those in B, making set A preferable over B. However, it is also necessary to calculate $COV(B,A)$ to make that statement, as coverage is not a symmetrical quality indicator.

This quality indicator is a convergence indicator. The larger it is, the more individuals from set B are dominated by at least one individual from set A, indicating less convergence of set B relative to set A.

An example of 2-dimensional coverage of two sets A and B can be seen in Figure 5.4.

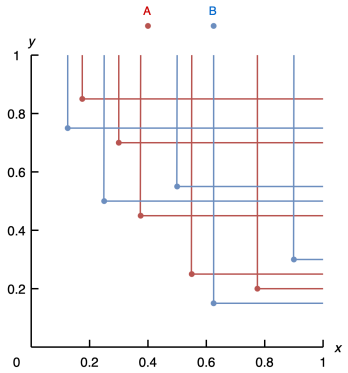


Figure 5.4.: In this image we can see solutions sets, identified by their objective values: A and B. It can be seen that $C(A,B) = \frac{2}{5}$ while $C(B,A) = \frac{3}{5}$. In set B there are solutions that dominate other solutions from the same set, and even so it covers more A solutions than what set A covers to set B.

5.4. NSGA-II Algorithm

The NSGA-II algorithm, or Non-Dominated Sorting Genetic Algorithm II, will be our proposal for a practical, efficient multiobjective optimization algorithm with a genetic approach. This algorithm emerges as an improvement over the original NSGA and is considered a basic and highly versatile model for multiobjective optimization problems. Its main differences from NSGA are that it uses a crowding operator instead of niches, it compares the parent population with the offspring population, and it is computationally more efficient.

The main features of this algorithm are the following:

- **Random seed**: NSGA-II uses a seed for random calculations to ensure reproducibility of results.
- **Initial population**: NSGA-II creates an initial random population of solutions. For our study, this initial population will include individuals with unrestricted size, as some decision space dimensions could need to be bounded in terms of the final size of these individuals, as shown in Section 9.3.
- **Offspring generation**: NSGA-II generates an offspring population using a classic genetic approach (selection, crossover, and mutation).
- **Non-dominance**: Once all offspring are combined with the original population, they are sorted according to the non-dominance criterion into as many level dominance fronts as necessary.
- **Selection**: After sorting, individuals from the best fronts are selected until the population size is reached. For the last front from which individuals are selected, they are sorted using a crowding distance function, which indicates the elements that are more dispersed relative to others in that dominance front. The best-performing individuals with respect to that function are selected until the population size is reached, promoting exploration by picking individuals from less explored areas.

The implemented algorithm is based on the one available in a public Github repository, and it can be found at the following link: `https://github.com/baopng/NSGA-II`. The pseudocode of the implementation is shown in Algorithm 2.

Crossover is performed according to the probability p_c. If it is applied, two parents (p, p') will be selected to generate offspring of two children (o, o'). For each hyperparameter h_i, the corresponding one for each of the two children is calculated as follows:

$$h_{i,o} = \frac{h_{i,p} + h_{i,p'}}{2} + \beta \frac{|h_{i,p} - h_{i,p'}|}{2}, h_{i,o'} = \frac{h_{i,p} + h_{i,p'}}{2} - \beta \frac{|h_{i,p} - h_{i,p'}|}{2} \tag{5.16}$$

Where $\beta \sim \mathcal{U}(0, 1)$. Therefore, each hyperparameter of each child is calculated as an average of the values of the parents, plus a random factor, which models how different the values of the parents were. In this way, the smaller the difference, the less variability in the new hyperparameters. For our study, $p_c = 1$ because, for each generation, the population from the previous one is considered to ensure elitist selection.

Algorithm 2: Meta-learning algorithm based on NSGA-II.

Input: Objective function $(f_1, \ldots, f_n)$, hyperparameter range $([\min(h_i), \max(h_i)], \forall i \in \{1, \ldots, m\}$, protected attribute name (A).
Data: Complete dataset (D)
Parameters: Number of generations (n_g), population size (n_i), mutation probability (p_M), crossover probability (p_c), crossover parameter (γ), random seed (s)
Output: Set containing predictors whose validation functions is the best approximation set found.

Set the random seed s.
Divide dataset D into training, validation and test sets (D_l, D_v, D_t).
Create initial population (P_0), containing n_i individuals.
Train each individual using the training dataset D_l.
Calculate objectives using validation dataset D_v.
Rank individuals with respect to Pareto dominance, dividing them into ranks.
Calculate crowding distance within each set.
for *j in range* $[1, n_g)$ **do**
    Create next population P_j containing n_i individuals using tournament selection, crossover (with probability p_c), and mutations (with probability p_M) over P_{j-1}.
    Train each individual using the training dataset D_l.
    Calculate objectives using validation dataset D_v.
    $P_j \leftarrow P_j \cup P_{j-1}$.
    Select one individual from P_j for each different objective value among them.
    If $|P_j| < n_i$, create as many copies as necessary of all individuals from P_j until $|P_j| \geq n_i$.
    Rank individuals with respect to Pareto dominance, dividing them into ranks.
    Calculate crowding distance within each set.
    $P_j \leftarrow$ best n_i individuals from P_j.
return *Non dominated individuals from* $P_{n_g} - 1$

With respect to mutation, the probability of it occurring for each generated individual is p_M. When it occurs, only one hyperparameter of the individual will be mutated, which may result in a detriment to diversity as the dimension of the search space increases, but it is an acceptable trade-off to improve the efficiency of the algorithm. Once the gene to be mutated is specified, two random values are generated, with $u \sim \mathcal{U}(0,1)$, and δ being the value given by:

$$\delta = \begin{cases} -1 + 2u'^{\frac{1}{1+\gamma}} & \text{if } u' \leq \frac{1}{2}. \\ 1 - 2(1-u')^{\frac{1}{1+\gamma}} & \text{if } u' > \frac{1}{2}. \end{cases} \text{ where } u' \sim \mathcal{U}(0,1) \tag{5.17}$$

Once δ is set, mutation is done as follows:

$$h_{i,o} = \begin{cases} h_{i,o} + \delta(h_{i,o} - \min(h_i)) & \text{if } u' \leq \frac{1}{2}. \\ h_{i,o} + \delta(\max(h_i) - h_{i,o}) & \text{if } u' > \frac{1}{2}. \end{cases} \text{ where } u' \sim \mathcal{U}(0,1) \tag{5.18}$$

There are two reasons why, in each generation, only one individual with each given objective function value is selected. First, it is more important to obtain a better overall representation of the Pareto front rather than multiple solutions with the same optimal objective function value. Second, due to the characteristics of the newly implemented models, there is a high probability that, using this algorithm to perform a hyperparameter optimization process based on these models, it finds a great amount of solutions having very extreme objective function values, as will be seen in Section 11.1. As this is not a desired result, this change improved getting a better populated Pareto front. In the case that selecting only one individual with each unique objective function value does not reach the population size n_i, as many copies of all individuals will be added to the population until it reaches or excedes n_i individuals. While this could impact the calculation of crowding distance, it is important to note that all non-selected individuals share the same objective values as the selected ones, which also affects the calculation of crowding distance. Diversity is indeed reduced, but this is a reasonable tradeoff to generally obtain better-populated Pareto fronts.

The time complexity of the basic NSGA-II algorithm is $\mathcal{O}(nn_i^2 n_g)$ (where n is the number of objectives, n_i is the population size, and n_g is the number of generations). This is because, to select individuals in each front, we need to compare each individual with all others, objective by objective. Therefore, the time complexity to compare an individual with all others in one generation would be of the order $\mathcal{O}(nn_i)$ and since this needs to be repeated for all individuals, the total time complexity would be of the order $\mathcal{O}(nn_i^2)$. Taking into account all generations, we arrive at the order $\mathcal{O}(nn_i^2 n_g)$ [Deb et al., 2002]. However, for the basic case of NSGA-II, the evaluation of the objective function for an individual is done in $\mathcal{O}(1)$, while in our case, it depends on the time complexity of our training algorithm.

Therefore, now our algorithm will have a time complexity of the order $\mathcal{O}(n_g(nn_i^2 + n_i a(n, n_i)))$, where $a(n, n_i)$ is the time complexity of the learning algorithm. If the learning algorithm has a complexity greater than nn_i, then the time complexity will be dominated by the second term, which is highly likely to happen. In our case, for both CART and LightGBM algorithms, this dominance happens.

NSGA-II algorithm is a basic algorithm in classical multiobjective optimization problems, due to its effectiveness and speed. But when dealing with Many-objective optimization prob-

lems, it does not yield such good solutions [Knowles and Corne, 2007], and it is preferable to use other algorithms designed to address the challenges that this kind of problems present.

5.5. Many-objectives optimization problems

We will also study a specific class of multiobjective optimization problems, known as "Many-objectives optimization problems" (MaOPs).

Definition 5.5.1 (Many-objectives optimization problem)**.** A Many-objective optimization problem is a multiobjective optimization problem with $m > 3$.

Due to their high dimensionality, MaOPs are particularly difficult to solve, and specialized techniques are required for this task, different from those we might use for problems with 2 or 3 objective functions. This is because various phenomena arise, such as the convergence resistance phenomenon. Therefore, we must employ algorithms specifically dedicated to addressing these challenges.

The main challenges that one has to face when trying to solve a Many-objectives optimization problem are the following [Li et al., 2015]:

- **Dominance Resistance (DR)**: When the number of dimensions increases, due to the dominance criterion we employ, many solutions that do not dominate each other start to appear. It is important to remember that if a solution is worse in all objectives compared to another except in one, there will be no dominance between them. This is why many conventional algorithms, such as NSGA-II, struggle to discern between better and worse solutions as the number of conflicting objectives increases [Purshouse and Fleming, 2007, Fonseca and Fleming, 1998, Corne and Knowles, 2007].
- **Population size**: Related to DR, as the number of solutions that no longer dominate each other increases, there will likely be a growing range of values in the Pareto-optimal set. If we want to fully describe it, the population size will have to be increased exponentially with respect to the number of objectives. This leads to a considerable decrease in the computational efficiency of the methods used to solve them.
- **Visualization**: Direct data visualization becomes impractical as the dimensionality increases. Therefore, specialized techniques are required, such as projection to different dimensions or dimensionality reduction, as well as examining other types of objective function representations. However, when applying these visualization techniques, we are inherently losing information, and to reconstruct the original information we should use a large number of specific representations, which is not usually the best option. Therefore, a compromise needs to be carefully established.

The algorithms currently representing the state-of-the-art for solving MaOPS are Many-Objectives Evolutionary Algorithms (MaOEAs). These genetic optimization algorithms, originally based on the NSGA-II algorithm, employ various techniques to better handle the abundance of objectives to optimize. There are many MaOEAs based on a wide range of concepts and techniques. In fact, they can be classified into different families [Li et al., 2015].

Part III.

Methodology

In this part, the theory and complete descriptions behind the developed algorithms will be explained to deeply understand them.

6. First algorithm: Fair Decision Tree

In this chapter, the underlying theory behind the first algorithm developed, Fair Decision Tree (FDT), will be discussed. As previously explained, this algorithm involves modifying the impurity criterion in the learning process of a decision tree to incorporate a fairness criterion.

6.1. Description of the algorithm

A traditional decision tree learning algorithm relies on an impurity calculation criterion, with the most commonly used ones being the Gini and entropy criteria. Both are closely related, as observed in Figure 3.2. However, these criteria do not consider fairness in the learning process. It would be beneficial to incorporate a fairness criterion to influence the learning process, aiming to infer fairer trees. Therefore, the main goal of this first algorithm is to incorporate and consider a fairness criterion during the learning process of a decision tree.

The incorporation of a fairness criterion during the decision tree learning process will be achieved by modifying the node impurity calculation criterion. This modification will involve a convex combination of a traditional impurity criterion with a fairness measure of our choice. Therefore, our new impurity criterion will take the form:

$$(1-\lambda)\cdot k\cdot \text{traditional impurity criterion} + \lambda\cdot(-\text{fairness criterion}) \tag{6.1}$$

Where λ is a parameter that controls the relative importance assigned to our fairness criterion. This parameter specification is crucial, and even a small change in it could lead to learning very different trees structurally. The optimal value for this parameter can heavily be influenced by the structure of each given problem, so a parameter optimization procedure can be crucial to find it. It is clear that if $\lambda = 0$, then no fairness metric is taken into account during the learning process, while if $\lambda = 1$, no Gini or entropy criterion will be considered during training (which usually is far from optimal).

The fairness criterion can take values in the range $[0, 1]$, while the values that the traditional impurity criterion can take vary between entropy ($[0, 1]$) and Gini ($[0, 0.5]$), as seen in Figure 3.2. To ensure that both fairness and impurity criteria are on the same scale, we will introduce a normalization factor k: it will be 1 when using entropy and 2 when using Gini.

In order to perform this calculation, we need to determine the value of the chosen fairness criterion at each node of the tree. During the training phase, new nodes of the decision tree will be generated, and a given set of instances from the training set will fall into them. Traditionally, for the Gini and entropy impurity criteria, only the number of instances from each class falling into that node is taken into account. However, when considering a fairness criterion, we cannot solely divide those instances by class; instead, we must also consider the

value of the protected attribute for each instance, as well as other relevant metrics.

Calculating group fairness metrics involves using the confusion matrix, dividing the instances by each demographic group. For this reason, these confusion matrices have to be considered to calculate any group fairness metric. However, to calculate a confusion matrix, predictions need to be made. The question arises: how do we calculate those predictions at any intermediate node, thereby determining the number of correctly and incorrectly classified instances from each group?

Using the traditional decision tree classification criterion could be considered, but this hinders the calculation of fairness metrics. For this reason, other methods for calculating these confusion matrices will be proposed. A classification of these methods will now be presented. They are divided using two dimensions. First, we will consider whether the prediction is constant (always predicting the same class) or probabilistic (the prediction is not constant, and there is a probability of predicting one class or another). Second, we will determine if the prediction is made dependent on the protected attribute or independent. This classification can be seen in Table 6.1.

Table 6.1.: Classification of criteria for calculating predictions and confusion matrices at any given node of a decision tree.

Are predictions probabilistic?	Considers fairness in node split criterion?	
	No	Yes
No	Constant prediction	Fair constant prediction
Yes	Probabilistic prediction	Fair probabilistic prediction

All the classes appearing in Table 6.1 will be now discussed:

- **Constant prediction**: This criterion is employed by traditional decision trees after being trained to assign a class to an instance, predicting the same class for all instances falling into a certain leaf. This poses a significant problem for our procedure because, in such cases, many fairness metrics will either be 0 or 1, and some may not even be calculable, thus assigning degenerate values as well. This limitation prevents us from considering gradual changes in fairness and implies an additional computational cost. Theoretically, the prediction in each leaf in this case would be $P[p = 0|A = 0] = P[p = 0|A = 1] = c \in \{0,1\}$, $P[p = 1|A = 0] = P[p = 1|A = 1] = 1 - c$. The estimation of the value c is typically made using maximum likelihood estimation. Let P be the number of training instances falling into that node belonging to the positive class, and N be the number from the negative class. Then, $c = 0$ if $P > N$ and $c = 1$ if $P < N$. If $P = N$, any criterion could be used.

- **Fair constant prediction**: In this case, it is more appropriate for the decision-making process to account for the value of the protected attribute since fairness measures are theoretically calculated using probabilities conditioned on this attribute's value. However, integrating this consideration does not resolve the issues of degeneration or the prior calculation impossibility. In each leaf, we would have $P[p = 0|A = 0] = c_0$, $P[p = 0|A = 1] = c_1$, where $c_0, c_1 \in \{0,1\}$, $P[p = 1|A = 0] = 1 - c_0$, and $P[p =$

$1|A = 1] = 1 - c_1$. Extending the notation from the previous class, considering P_0 as the number of training instances belonging to positive class and unprivileged group (P_1, N_0 and N_1 analogously), then $c_i = 0$ if $P_i > N_i$, and $c_i = 1$ if $P_i < N_i$, $i \in \{0,1\}$.

- **Probabilistic prediction**: In this scenario, we encounter a situation analogous to constant prediction, with the distinction being that probabilities may have values different from 0 or 1. In each leaf, we would have $P[p = 0|A = 0] = P[p = 0|A = 1] = c \in [0,1]$, and $P[p = 1|A = 0] = P[p = 1|A = 1] = 1 - c$. In this case, for the estimation of parameter c using maximum likelihood estimation we can be more precise. The value of c will then be $c = \frac{N}{P+N}$.

- **Fair probabilistic prediction**: This prediction approach is arguably the most theoretically complex among those discussed, but it is very interesting since the group fairness criteria always involve conditioning probabilities on the protected attribute. Therefore, it can be useful to calculate prediction probabilities by conditioning them on the values of the protected attribute, even though the final probability in the leaves is computed without considering these values. In this approach, $P[p = 0|A = 0] = c_0$, $P[p = 0|A = 1] = c_1$, where $c_0, c_1 \in [0,1]$, $P[p = 1|A = 0] = 1 - c_0$, $P[p = 1|A = 1] = 1 - c_1$. In this case, $c_0 = \frac{N_0}{P_0+N_0}$, and similarly, $c_1 = \frac{N_1}{P_1+N_1}$. However, this method introduces a drawback when calculating false positives and false negatives, as both will have the same value $\frac{P_i N_i}{P_i+N_i}$ for each $i \in \{0,1\}$. Consequently, certain fairness metrics may be equal. Notably, the disparities between True Positive Rate (TPR) and Positive Predictive Value (PPV) will be equal, given that false positives and false negatives are equivalent.

An example of the different confusion matrices calculated using the discussed criteria for the same node of a decision tree is shown in Figure 6.1.

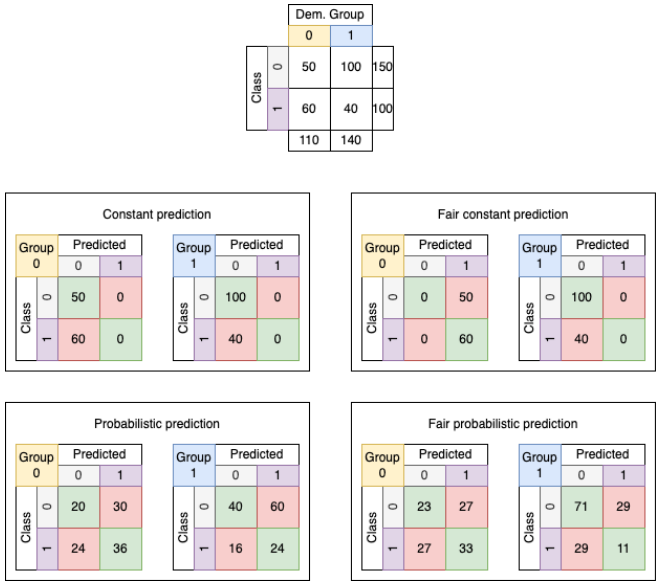


Figure 6.1.: Different criteria for calculating confusion matrices are applied to a node, showing its training instances along with their respective classes and groups in the upper table.

The criterion we will use for calculating confusion matrices at each intermediate node in our implementation will be the latter one: fair probabilistic prediction. This is because it is the most complex and leads to fewer degenerate values in both confusion matrices and fairness metrics. In particular, FPR_{diff}, which is the fairness metric that will be used in the experimentation, is a calculable fairness metric for this criterion. Nevertheless, fair probabilistic prediction is not a perfect criterion, and any other could be used instead.

The pseudocode of the algorithm is essentially the same as Algorithm 1, with the only difference being the consideration of this new modified impurity criterion, where fair probabilistic prediction will be used for calculating confusion matrices at each node.

6.2. SWOT Analysis

In this subsection, the different strengths, weaknesses, opportunities, and threats of this algorithm will be analyzed. These analyses, commonly known as SWOT analyses for their acronym, are two-dimensional assessments frequently conducted in the business world to evaluate the viability of a particular proposal. Before conducting the analysis, we must first specify the meaning of each of these concepts within our context.

- **Strengths**: These are the internal strong points where the algorithm excels. They are characteristics that make it a viable option for use.
- **Weaknesses**: These are the internal weak points where the algorithm falters. They are characteristics that make one consider whether it should be used or not.
- **Opportunities**: These are external factors that can potentially make the algorithm stronger. In this case, we will consider all improvement possibilities and potential future development lines for this algorithm.
- **Threats**: These are external factors that can potentially make the algorithm weaker. In this case, we will consider all possible competitors that rival the algorithm and might lead to reconsidering which algorithm to use.

SWOT analyses are considered to be two-dimensional analyses since these characteristics can be grouped into the following two dimensions:

- **Goodness**: Are these characteristics helpful (strengths and opportunities) or harmful (weaknesses and threats) in terms of achieving our goal?
- **Origin**: Are these characteristics internal to the algorithm (strengths and weaknesses) or external to it (opportunities and threats)?

Once this analysis has been defined, the characteristics identified for the FDT algorithm can be examined in Figure 6.2.

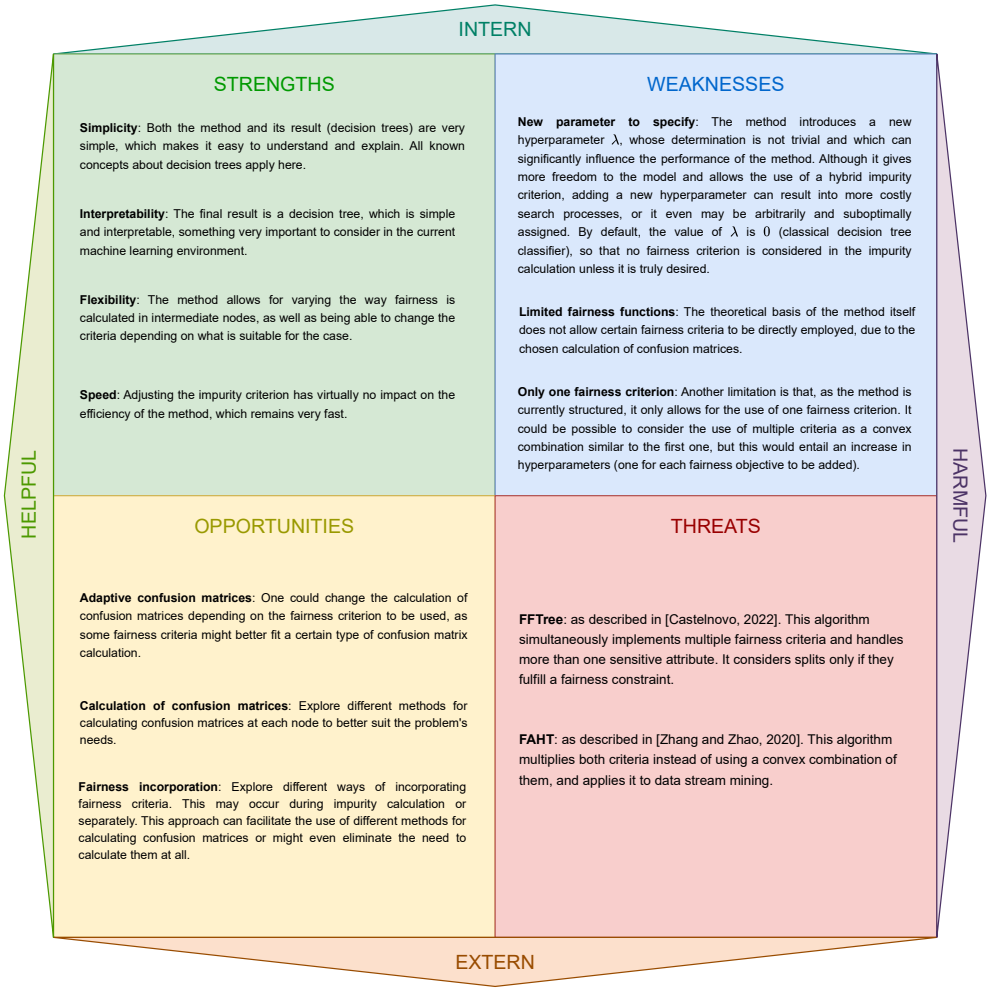


Figure 6.2.: SWOT analysis for the FDT algorithm. The references for the threat algorithms are as follows: [Castelnovo, 2022] and [Zhang and Zhao, 2020].

7. Second algorithm: Fair Genetic Pruning

In this chapter, the second developed algorithm, named Fair Genetic Pruning (FGP), will now be discussed. This algorithm is based on genetic metaheuristics, decision trees, and pruning techniques applied to them. The basic idea is to construct the largest decision tree possible and then create a space for pruning it, from which individuals are derived as pruned versions of the base tree.

7.1. Description of the algorithm

In this section, each characteristic of the algorithm will be described in detail. These include decision space, representation of individuals, population initialization, crossover criterion, mutation criterion, distance between decision trees, and generational replacement.

7.1.1. Decision space

To define the decision space of our problem, we first need to understand what constitutes a solution for the problem we are dealing with and explore possible characterizations of it.

We start with a binary matrix tree, which serves as the foundation for our algorithm. This matrix tree is created without restrictions in terms of size. In practice, multiple such trees are built using different balances for both positive and negative classes, as problems can be unbalanced, which affects the size of the matrix tree and its initial accuracy. We then select the one with the least error (measured by $1-$ G-mean, as used in the experimentation). If some of them share the same value for this metric, then we select the one with the greatest number of leaves (providing the largest pruning space).

This approach leads to the construction of a decision tree that perfectly classifies our training sample, but it will likely overfit the training set, resulting in a drop in precision in any real deployment environment (as well as for validation and test sets). Fairness metrics measured on this matrix tree will not be as good either, considering the tradeoff between precision and fairness. This tree will be rather large; in fact, it is the optimal tree with respect to the impurity criterion used, perfectly classifying the training sample. This implies that any other tree using the same impurity criterion will classify our training sample worse in terms of accuracy and will be smaller in size. The key advantage of this tree is that it is the largest we can build with the available training samples and is optimized with respect to the impurity criterion.

The main goal of this algorithm is to find the subtrees of this matrix tree that have the same root node and the optimal balance in terms of the considered objective functions. Consequently, our decision space will include all possible subtrees with the same root node as our matrix tree. The root has to always be the same because starting the classification process from a different root node would mean that only a subset of the training sample is considered,

which does not make sense. The classification process should always start the same way, so the root node must be the same too.

It is important to note that a subtree of the matrix tree, concerning the already cited requirements, can also be seen as a set of prunings applied to the matrix tree. This is because applying pruning to a tree results in a new tree that has the same structure as the base one, but with all the subtree structure after a given node removed. All subtrees sharing the root node of the matrix tree can be constructed by applying a set of prunings to it. For this reason, the decision space can also be seen as the possible trees that can be obtained after applying a set of prunings to the matrix tree. We will now explore this fact.

7.1.2. Representation of individuals

As we have mentioned, the algorithm will consider a matrix tree, which will be the base tree from which to generate all population individuals, being subtrees of it sharing the same root. Additionally, all these subtrees can also be defined as sets of prunings performed on the matrix tree. Using these characteristics, an equivalence can be established between a subtree and the set of prunings that characterize it. In this way, we will establish two ways of representing population individuals, which will be used during the algorithm depending on which one is more suitable for each task to be performed:

- **Pruning representation**: Each individual in our population can be represented as a set of prunings applied to the matrix tree. Since we are only working with binary trees, then we can create a simple mathematical representation of its nodes and use it to define what constitutes a pruning.

  Definition 7.1.1 (Representation of nodes in a binary tree)**.** A binary tree, which is a directed graph with only one root node, can have its nodes represented by an ordered list of values $[n_1, n_2, \ldots, n_r]$, where $n_i \in \{0, 1\}, \forall i \in \{1, \ldots, r\}$, representing a path to follow. Starting from the root node, a value of $n_1 = 0$ means that we will travel through its left child, and a value of $n_1 = 1$ means that we will move to its right child. We continue this procedure iteratively from the node where we arrive after following all the previous path indications ending at the node where we arrive after applying the last value n_r.

  Definition 7.1.2 (Pruning of a matrix tree)**.** A pruning of the matrix tree involves removing the child nodes and links from a particular node. The node at which the pruning occurs is specified by an ordered list of values $[n_1, n_2, \ldots, n_r]$, where $n_i \in \{0, 1\}$, $\forall i \in \{1, \ldots, r\}$. This list indicates the path from the root node to the node where the pruning will be performed. Thus, we identify this operation as pruning $[n_1, n_2, \ldots, n_r]$.

  Using this definition, we can represent any individual as a set of prunings. The matrix tree would be represented by [ ] (absence of prunings), while other subtrees would be represented by the prunings that have been performed to them. There could be redundant prunings in the representation of an individual (for example, for an individual having the representation $[[0, 1], [0]]$ the pruning $[0, 1]$ would be meaningless

since the pruning [0] has already been applied). This redundancy is not desired since we want that all individuals have unique representations. Fortunately, prunings can be efficiently sorted using lexicographic order. This will not only allow us to avoid redundant prunings with efficient checks but also to perform other optimizations at the code level. We will consider ordered representations without any redundant prunings as the actual representations, which are unique and minimal for each individual. Examples of this representation of individuals can be seen in Figure 7.1.

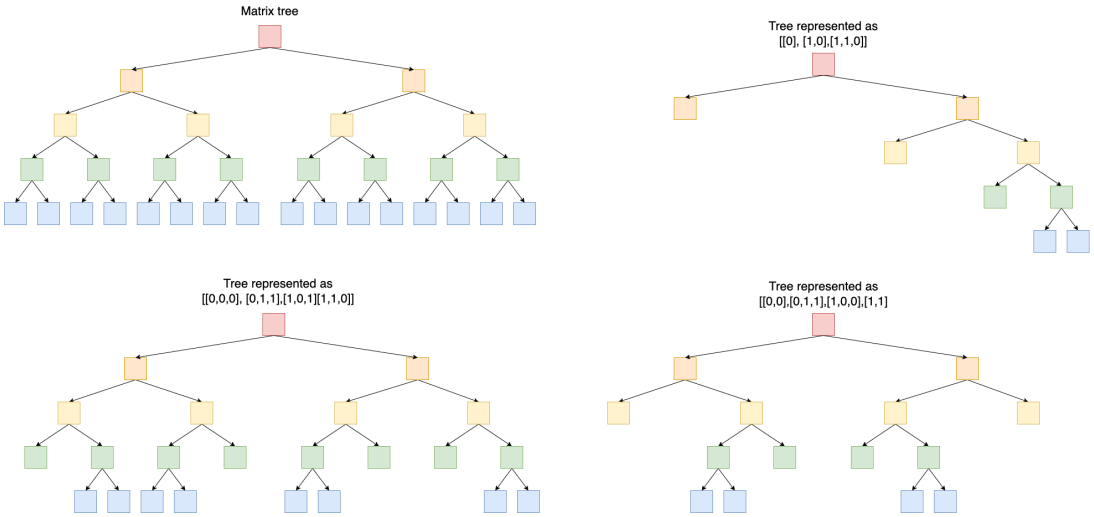


Figure 7.1.: Pruning representations of individuals given a matrix tree. These representations have their prunings sorted lexicographically and do not include any redundant prunings.

- **Leaves representation**: We can use the representation of nodes and prunings to define the structure of our matrix tree. A key observation, as shown in Figure 7.1, is that the pruning representation can actually be interpreted as a representation in which only the new leaves that were not originally part of the matrix tree appear. This characteristic allows us to extend this notation and represent the matrix tree using a format that includes only the representation of its leaves. From a pruning perspective, we can consider the matrix tree as an 'infinitely large' tree, where a pruning was performed at each of its actual leaves.

  With this consideration, we can define another representation of individuals as shown in Figure 7.2, where each leaf that the tree ultimately has is encoded. It will be known as "leaves representation". This representation is often very redundant compared to pruning representation, which allows us to save many elements. However, it can be useful for certain tasks, such as the mutation criterion, which will be discussed later.

7.1.3. Population initialization

To create the initial population, we need to define the matrix tree as mentioned earlier. For this task, all of our training instances need to be used, and the largest (unrestricted in terms

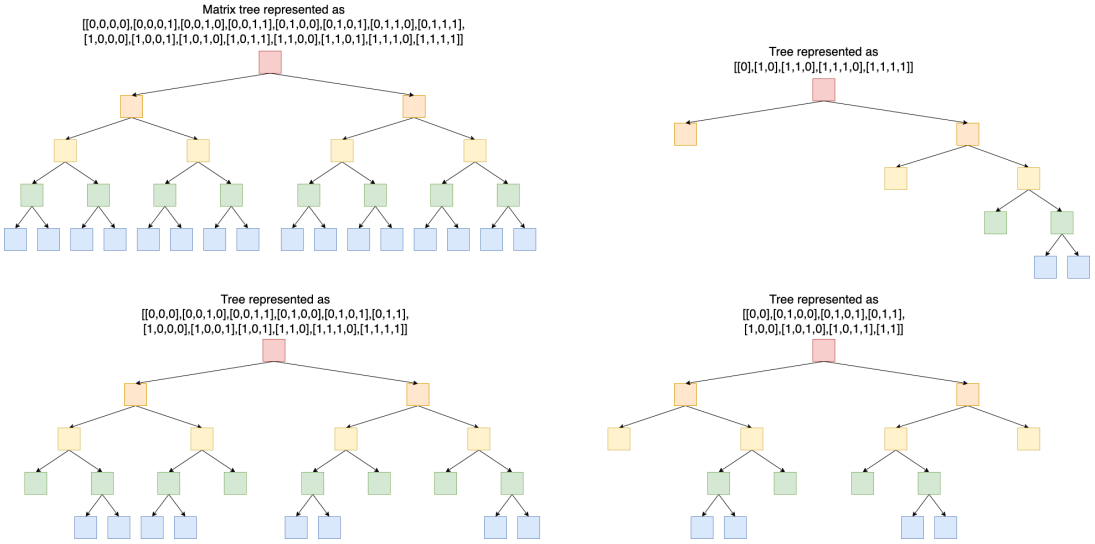


Figure 7.2.: Example of leaves representation of the same individuals and matrix tree as shown in Figure 7.1.

of size, considering different balances for both classes) tree has to be trained. Once our matrix tree is established, we can begin defining our initial population based on it.

For our initial population, we will have a set of fixed size n_i containing subtrees as defined in Section 7.1.2. This population size n_i will not vary over time, so it is a parameter that has to be previously specified.

The goal of defining this initial population is to create a diverse set of individuals that will aid us in searching for good solutions as generations pass. To achieve this, we will follow the following criteria:

- **Matrix tree**: The matrix tree (represented as [ ], no prunings applied) will be included in the initial population.
- **Pruning probabilities**: For the remaining individuals up to size n_i, we consider the matrix tree and traverse it from the root. The probability of pruning each node will be assigned independently of its depth, making all nodes equally likely to be pruned. Let us denote this probability of pruning a certain node as α. A crucial observation here is that if an ancestor of a given node has already been selected for pruning, that node cannot be selected for pruning. This implies that the probability of any deeper node must be greater than that of any shallower one.

  The probability of pruning nodes at depth 1 will be α. For nodes at depth 2, the probability cannot be α, because if its parent has been pruned, it cannot be selected for pruning. Therefore, the actual probability we have to assign is x, satisfying $\alpha = x(1-\alpha) \Rightarrow x = \frac{\alpha}{1-\alpha}$. We can iteratively apply this process to determine that the probability we have to assign to a node at depth n is $\frac{\alpha}{(1-\alpha)^{n-1}}$, in order to be selected for

pruning with probability α.

However, we must note that $f(n) = \frac{\alpha}{(1-\alpha)^{n-1}}$ is monotonically increasing with respect to n, and it can exceed 1. In fact, $f(n) = 1 \Leftrightarrow \frac{\alpha}{(1-\alpha)^{n-1}} = 1 \Leftrightarrow \alpha = (1-\alpha)^{n-1} \Leftrightarrow \log(\alpha) = (n-1)\log(1-\alpha) \Leftrightarrow n = 1 + \frac{\log(\alpha)}{\log(1-\alpha)} = 1 + \log_{1-\alpha}(\alpha)$. This implications are bidirectional as $0 < \alpha < 1$.

This means that for any tree of depth n, we can assign a probability α which satisfies $1 > \frac{\alpha}{(1-\alpha)^{n-1}} \Leftrightarrow 1 > \alpha + \alpha^{\frac{1}{n-1}}$. To numerically solve it, it is better to express it in terms of the inequality $\alpha - (1-\alpha)^{n-1} < 0$, which can be solved very efficiently using numerical methods.

- **Final pruning probability**: As a general criterion, given a matrix tree of depth n, our value α will be half of the solution to the equation $x - (1-x)^{n-1} = 0$.

We can see an example of this in Figure 7.3.

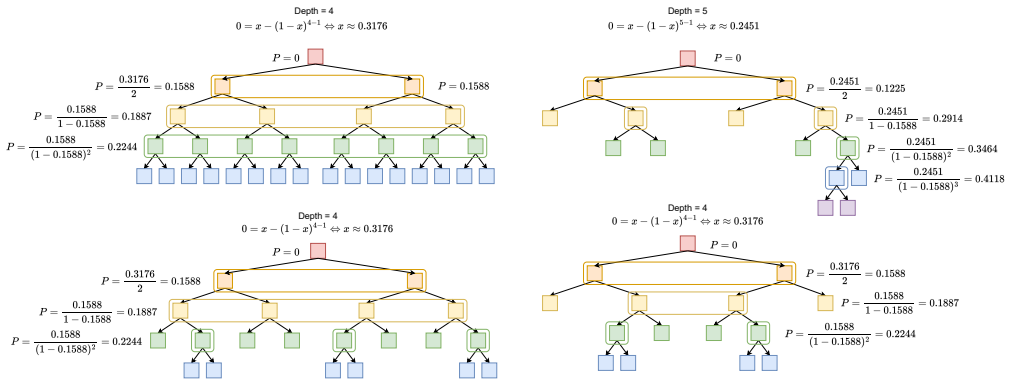


Figure 7.3.: Initial probabilities associated with the pruning of each of the nodes of the matrix trees displayed.

7.1.4. Crossover between individuals

Crossover between individuals is a crucial aspect to consider when defining a genetic evolutionary algorithm. Once we have a population of solutions, we will need to create a population of descendant solutions. To do this, we need to specify how parent solutions are selected (selection criterion), and how the crossover between them is performed, in cases where it is applied.

In our case, we will use two parent solutions to generate a pair of child solutions. To select each parent individually, we will randomly choose 2 individuals from the current population. We will evaluate how fit these individuals are for reproduction (in terms of the objective function) and select the better one. If neither is better than the other, one will be randomly selected. Once both parents are chosen through this procedure, they may or reproduce with a

probability p_c initially defined. If no crossover occurs, the two parent solutions are returned to belong to the new population of child solutions. In our case, we will set $p_c = 1$, as all individuals from the previous and the current generation will be considered in the generational replacement stage, as discussed in Section 7.1.7.

The crossover procedure to generate two offspring from two parent individuals will be as follows:

- **Select probability**: To begin with, a random value $\gamma \in [0.2, 0.8]$ using an uniform distribution will be considered. This value will control how prunings will be assigned from the parents to the children. γ will represent the probability of assigning a pruning from the parents to the first child, and therefore the probability of assigning it to the second child will be $1 - \gamma$. This value is used instead of always using a probability of 0.5 to generate more diversity and better explore the search space.
- **Decide to which child the first pruning will belong**: We will consider the pruning representations of both parents. We will begin by taking, using lexicographic order, the first pruning item from among the two parents' representations. It will be assigned to the first child with probability γ, and if not assigned, then it will be assigned to the other child. There is one exception to this rule: if an attempt is made to assign a pruning to a child at a node that has already been pruned, it will be assigned to the other individual. For example, if the first child already contains the pruning $[0]$, it cannot be assigned the pruning $[0, 1]$; therefore, it will be assigned to the other child. If any can have that pruning assigned, the pruning will be discarded.
- **Iterate**: This process is repeated iteratively until both parent representations are exhausted.

The pseudocode associated with the process of obtaining a population of child solutions from a given population and the crossover of trees can be shown in Algorithm 3.

7.1.5. Mutation criterion

A mutation of an already created solution consists of altering that solution and creating another similar one. The purpose of mutation is to introduce more variability in the search process, enabling the algorithm to explore other regions of the search space that would not typically be explored. Therefore, this aspect enhances exploration rather than exploitation of our search space.

When discussing similarity, we need to introduce a distance metric that allows us to calculate similarities across the search space. To maintain consistency in terms of similarity, it is common to establish a boundary that enables the generation of similar solutions around a given one. Using the distance metric, we can consider a topological ball centered at our current solution, with the boundary defined by a predefined radius. A mutation involves randomly generating a solution within that ball.

In our case study, the search space consists of subtrees of a given matrix tree sharing the root node, which can also be viewed as different pruning sets applied to that matrix tree. To define how mutations will be done, we need to introduce a distance metric over this space.

Algorithm 3: Children population generation and crossover criterion for FGP algorithm.

Input: Objective function $f = (f_1, \ldots, f_n)$, previous population P.
Parameters: Crossover probability (p_c), crossover parameter (γ).
Output: Children population P'.

```
Create an empty children population P' = [].
for i ∈ {1, ..., n_i/2} do
    for j ∈ {1, 2} do
        Select 2 random individuals P_{j,1} and P_{j,2}.
        Select the best between them (P_j) with respect to the objective function f.
    if random uniform number in [0, 1) < p_c then
        Create two empty representations for children P'_1 and P'_2.
        Join both P_1 and P_2 pruning representations, P_t, preserving order.
        Assign a random probability γ ∈ [0.2, 0.8] to P'_1.
        for Each pruning p in P_t do
            if The node represented in p has already been pruned in P'_1 then
                if p has not been pruned in P'_2 then
                    Assign p to P'_2.
                else
                    Discard p.
            else if The node represented in p has already been pruned in P'_2 then
                if p has not been pruned in P'_1 then
                    Assign p to P'_1.
                else
                    Discard p.
            else
                if Random uniform number in [0, 1) < γ then
                    Assign p to P_1.
                else
                    Assign p to P_2.
        Add P'_1 and P'_2 to children population P'.
    else
        Add P_1 and P_2 to children population P'.
return Children population P'.
```

Taking advantage of the fact that individuals in this space are decision trees, we can define a distance measure between these objects.

7.1.6. Distance between trees

There are various distances between trees that can be used. Some of the most classical ones, such as Tree Edit Distance (TED), involve calculating the optimal number of insertion/deletion operations needed to transform one tree into the other. Other metrics have been developed, for instance, in the context of phylogenetic trees, which may include operations like contraction and rearrangement of nodes. Additionally, there are methods known as "kernel methods", which translate trees into another scalar feature space, enabling the calculation of a dot product between the representations of both trees. The resulting value can then be used to compare them.

For our algorithm, a modification of the Tree Edit Distance criterion will be used to define a specific neighborhood for decision trees. Given a decision tree, its neighborhood will consist of all trees that can be generated by applying contraction/expansion operations on a specific leaf (pruning), up to a maximum of a levels. To achieve this, we first choose a node. Subsequently, it will be decided whether to contract/expand, and how many levels (distance x) randomly between 1 and $\left\lceil \frac{\text{Max depth}}{10} \right\rceil$, where "Max depth" is the maximum depth reached in the matrix tree. A contraction involves pruning its x-th predecessor, and an expansion (specifically for prunings) involves removing the pruning and advancing downwards a total of x positions. It is important to note that, for example, if pruning $[0]$ becomes pruning $[0, 1, 0]$, the entire branch hanging from nodes $[0, 1]$ and $[0, 1, 1]$ are not pruned, extending all the way to the leaves of the matrix tree. This is done to avoid a drastical increase in the number of prunings performed, since in the case of extending a node by x levels could potentially introduce up to 2^x new prunings to the representation.

If a mutation is performed to an individual, this whole process of selecting a leaf and mutating it will be repeated a random number of times r, selected from 1 to $\left\lceil \frac{\text{Num leaves}}{10} \right\rceil$ with uniform probability, where "Num leaves" is the number of leaves that individual has. We can see an example of this with a radius distance of 1 in Figure 7.4. All possible mutations in that figure can be seen in Figure 7.5.

Having considered this distance criterion between trees, it is clear that the leaves representation is more beneficial for its implementation, as it considers all the actual leaves of the tree to be mutated. It is much more efficient to implement a method to translate between pruning and leaves representations than to work with the pruning representation using this mutation criterion.

The pseudocode associated with the mutation process of an individual can be found in Algorithm 4.

7.1.7. Generational replacement

The generational replacement will be similar to that used in Algorithm 2. We unite all solutions from the current population with all solutions from the offspring population (after

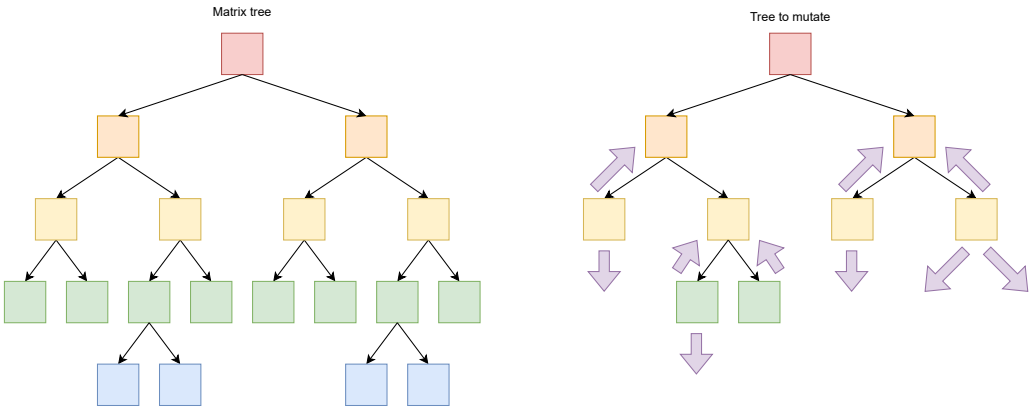


Figure 7.4.: Representation of a matrix tree and a given tree to mutate using a distance of 1. Purple arrows indicate the possible paths that can be traversed for mutation.

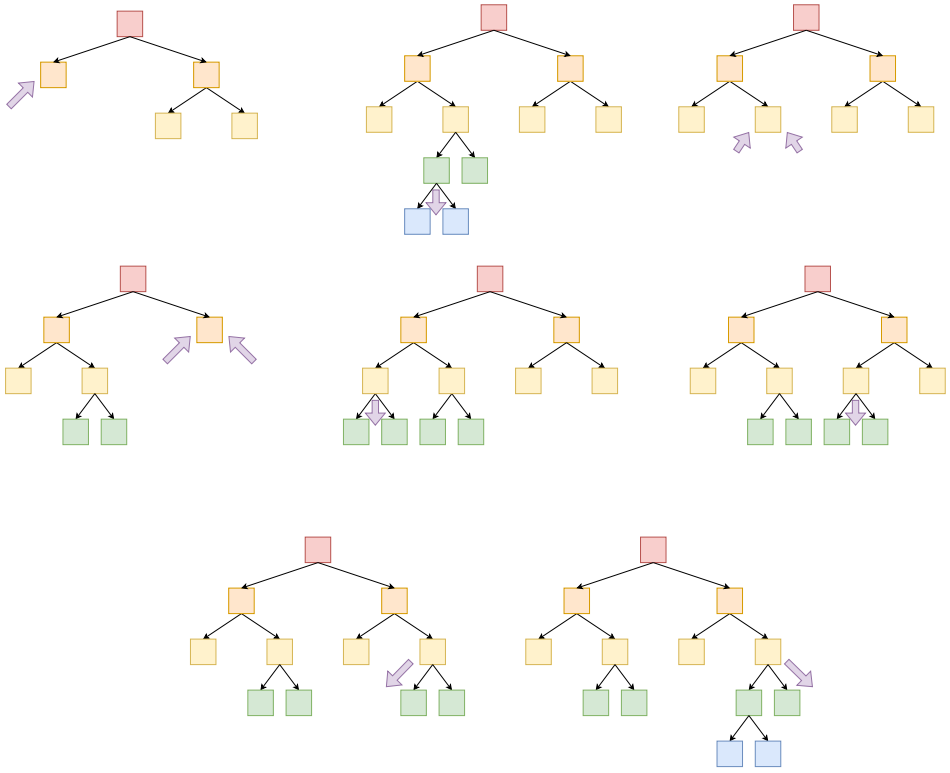


Figure 7.5.: All possible mutations of discance 1 of the tree shown in Figure 7.4.

Algorithm 4: Mutation of an individual.

Input: Individual to mutate I.
Parameters: Mutation probability (p_M).
Output: Mutated solution I'.

if *Random uniform number in the range* $[0, 1] < p_M$ **then**
  $L =$ leaves representation of I.
  Select a random number r, with uniform probability from $\{1, \ldots, \frac{|L|}{10}\}$.
  for *i in* $\{1, \ldots, r\}$ **do**
    Select $l \in L$ with uniform probability and create a copy of it $l' = l$.
    Decide with uniform probability to go down (if l is not a leaf node of the matrix tree) or up (if l is not the root node).
    Randomly select the distance $x \in \left\{1, \ldots, \left\lceil \frac{\text{Max depth}}{10} \right\rceil \right\}$.
    for *k in* $\{1, \ldots, x\}$ **do**
      if *It was decided to go up* **then**
        if *Length of* $l' > 0$ **then**
          Remove last element from l'.
      else
        Randomly insert 0 or 1 to l' with uniform probability.
        if l' *is not a valid pruning* **then**
          Undo the insertion.
    Remove l from I' and insert l' using lexicographical order.
  Convert I' to pruning representation.
else
  $I' \leftarrow I$.
return *Mutated solution* I'.

crossovers and mutations) to ensure elitism in the generational replacement. Then, all individuals are sorted as in NSGA-II. Dominance fronts are established, and solutions are selected from these fronts as in NSGA-II.

7.2. SWOT Analysis

The SWOT analysis for FGP algorithm is presented in Figure 7.6.

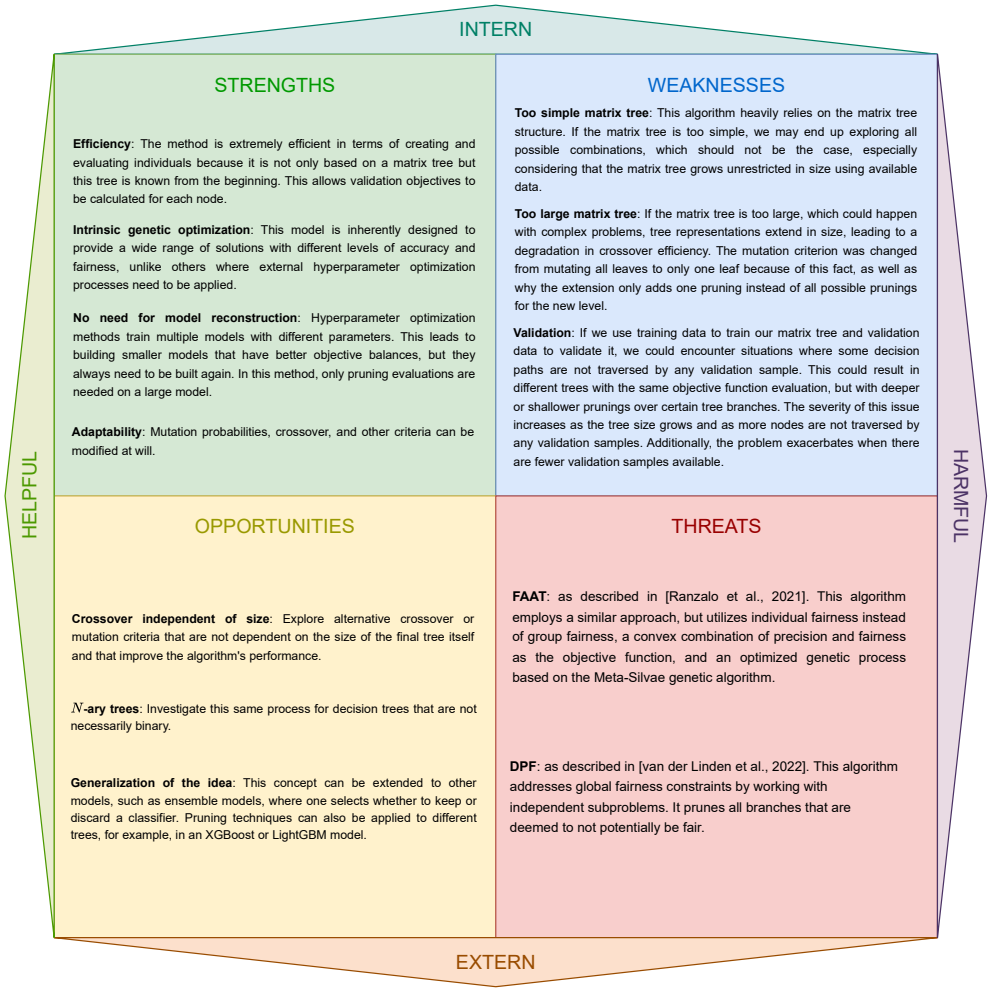


Figure 7.6.: SWOT Analysis for the FGP algorithm. The references for the threat algorithms are as follows: [Ranzato et al., 2021] y [van der Linden et al., 2022].

8. Third algorithm: Fair LightGBM

In this chapter, the final algorithm developed will be discussed. This algorithm, named Fair LightGBM (FLGBM) is based on the LightGBM algorithm. LightGBM is conceptually an algorithm which employs Gradient Boosted Decision Tree (GBDT) algorithm with additional features that enhance its scalability for high-dimensional datasets, considering both features and instances. The final result is a robust ensemble model containing weak decision trees classifiers.

We will introduce Gradient Boosting in Machine Learning, focusing on Gradient Boosted Decision Tree. Related algorithms like XGBoost and LightGBM will also be discussed. Particularly, LightGBM will be the main focus, and we will explore how the gradient function can be modified to incorporate fairness into it.

8.1. Introduction to Gradient Boosting in Machine Learning

In this section, we will introduce the basic concepts of Gradient Boosting (GB). We will begin by establishing the mathematical setup needed to understand how boosting works using parameterized functions. Then, we will extend it to using a function space approach and explore the problems of using a finite sample and how to solve them. Finally, we will discuss regularization.

8.1.1. Mathematical setup

In order to conceptually understand what gradient boosting is, it is important to clarify the context in which we are working [Friedman, 2001]. In every machine learning problem, the primary goal is usually to predict the value of a response variable Y using the values of some explanatory variables $X = \{X_1, \ldots, X_n\}$. In theory, there exists a function $f : \mathbb{R}^n \to \mathbb{R}$ which maps X into Y, and our aim is to estimate that function f using a finite training sample of size $N \in \mathbb{N}$: $\{(x_i, y_i) : i \in \{1, \ldots, N\}\}$ being $x_i = (x_{i_1}, \ldots, x_{i_n})$. The estimation is performed in such a way that it minimizes the expectation of a loss function, typically defined by the user as $L(Y, f(X))$, over all possible distributions of (X, Y).

$$f^* = \arg\min_f E_{Y,X} L(Y, f(X)) = \arg\min_f E_X[E_Y(L(Y, f(X))|X] \tag{8.1}$$

The value of this loss function also needs to be estimated using the training sample. Some frequent regression loss functions are $(Y - f(X))^2$ and $|Y - f(X)|$, which are named squared error and absolute error, respectively. For classification, one of the most commonly employed is the log loss function, whose expression is $-(Y \log(f(X)) + (1 - Y) \log(1 - f(X)))$, where Y can only take values 0 and 1, and f is a function that returns probabilities for belonging to the positive class (typically the logistic function). This function will be studied in Section 8.3.

The main problem is that the space in which to find the function f is infinitely dimensional, so some restrictions have to be imposed on it to make the problem tractable. For that reason, one of the most common approaches is to use a parametrized class of functions, denoted as $f(X;P)$, where $P = \{P_1, \ldots, P_k\}$ is a set of parameters for that function, and its values uniquely identify one function.

The function to be studied is a parameterized function, which follows an additive expansion expression:

$$f(X;\{(\beta_m, A_m)\}_1^M) = \sum_{m=1}^{M} \beta_m h(X;A_m) \tag{8.2}$$

Where $\{(\beta_m, A_m)\}_1^M = \{(\beta_m, A_m) : m \in \{1, \ldots, M\}\}$, being $M \in \mathbb{N}$. The function $h(X;A_m)$ is itself another parametrized function that uses parameters $A = \{A_1, \ldots, A_{m_2}\}$, and $\beta_m \in \mathbb{R}, \forall m \in \{1, \ldots, M\}$. For it to be useful in practical applications, h must be simple enough to allow for multiple approximations within a given timeframe. This additive expansion is widely employed in many machine learning models, including neural networks, radial basis functions, and support vector machines, and constitutes an ensemble model. In our scenario, h will be a simple decision tree, and the parameters will represent the splitting variables.

When using a parametrized class of functions, the problem of function estimation can be rewritten using the following notation:

$$\begin{aligned} P^* &= \arg\min_P \phi(P), \\ \phi(P) &= E_{Y,X} L(Y, f(X;P)) \\ f^*(X) &= f(X;P^*) \end{aligned} \tag{8.3}$$

We need to apply numerical optimization methods to solve this equation in most cases, as it is still a difficult problem to solve analytically. An expression that is commonly used to find the best set of parameters is:

$$P^* = \sum_{m=0}^{M} p_m \tag{8.4}$$

We start with an initial guess p_0, and then refine these parameters in subsequent steps, incorporating all previous adjustments. The computation of each subsequent p_k is determined by the optimization method. One of the simplest and most frequently used methods for refining this value is the steepest descent, which involves calculating the derivative of the function ϕ with respect to the parameters:

$$g_m = \{g_{j,m}\} = \left\{ \left[\frac{\partial \phi(P)}{\partial P_j} \right]_{P=P^*_{m-1}} \right\}, P^*_{m-1} = \sum_{i=0}^{m-1} p_i \tag{8.5}$$

The step taken will be $p_m = -\rho_m g_m$, where:

$$\rho_m = \arg\min_\rho \phi(P^*_{m-1} - \rho g_m) \tag{8.6}$$

The negative gradient $-g_m$ defines the steepest descent direction, and ρ_m is the optimal decreasing value along that direction.

8.1.2. Numerical optimization using function spaces

Let us suppose that we are not dealing with parametrized functions, but instead we are considering a function space. Returning to the initial expression:

$$\phi(f) = E_{Y,X} L(Y, f(X)) = E_X[E_Y L(Y, f(X))|X] \tag{8.7}$$

We can express for a certain X:

$$f^*(X) = E_Y[L(Y, f(X))|X] \tag{8.8}$$

Despite being in an infinite-dimensional space of functions, datasets only contain a finite amount of data. Therefore, we will have only a finite set of $f(x_i)$, where $i \in \{1, \ldots, N\}$. We will optimize using the numerical optimization procedure discussed in Section 8.1.1, so the solution will be:

$$f^*(X) = \sum_{m=0}^{M} f_m(X) \tag{8.9}$$

Where $f_0(X)$ is the initial guess, and the subsequent ones are the boosts over that initial guess. For steepest-descent:

$$f_m(X) = -\rho_m g_m(X) \tag{8.10}$$

where:

$$\begin{gathered} g_m(x) = \left[\frac{\partial \phi(f(X))}{\partial f(X)}\right]_{f(X)=f^*_{m-1}(X)} = \left[\frac{\partial E_Y[L(Y, f(X))|X]}{\partial f(X)}\right]_{f(X)=f^*_{m-1}(X)} \\ f^*_{m-1}(X) = \sum_{i=0}^{m-1} f_i(X) \end{gathered} \tag{8.11}$$

And finally, assuming that we can interchange differentiation and integration, which is not true in general, we can obtain the final expression:

$$g_m(X) = E_Y\left[\frac{\partial L(Y, f(X))}{\partial f(X)}\middle| X\right]_{f(X)=f^*_{m-1}(X)} \tag{8.12}$$

and ρ_m is determined by the line search:

$$\rho_m = \arg\min_{\rho} E_{Y,X} L(Y, f^*_{m-1}(X) - \rho g_m(X)) \tag{8.13}$$

8.1.3. The problem with finite data samples

The issue with using a finite data sample in the previous approach is that we have to estimate the distribution of (X, Y) with a finite data sample, $\{(x_i, y_i) : i \in \{1, \ldots, N\}\}$. $E_Y[\cdot|X]$ is generally not accurately estimable given only the finite number of values (x_i, y_i), and those known data points have limited relevance since we are interested in estimating $f^*(X)$ at values of X different from the known ones. Therefore, we need to impose a certain level of smoothness on the actual function f in order to infer f^* using the available information. Although there will likely be some error, our ability to work with this data will be significantly improved. The most common way to impose smoothness is, again, by considering a

parametrized class of functions to search from. In this sense, we search for parameters that match:

$$\{(\beta_m, A_m)\}_1^M = \arg \min_{\{\beta'_m, A'_m\}_1^M} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta'_m h(x_i; A'_m)\right) \tag{8.14}$$

Typically, there are situations where this expression becomes intractable, even with the steepest-descent optimization procedure. In such cases, we can resort to a greedy stagewise approach. We can calculate the parameters using the following expression:

$$(\beta_m, A_m) = \arg \min_{\beta, A} \sum_{i=1}^{N} L(y_i, f^*_{m-1}(x_i) + \beta h(x_i; A)) \tag{8.15}$$

Where the function f^*_m is calculated using only the previous one (which internally contains all previous additions):

$$f^*_m(X) = f^*_{m-1}(X) + \beta_m h(X; A_m) \tag{8.16}$$

And this is done $\forall m \in \{1, \ldots, M\}$. This procedure is known as "boosting". The function h is known as "base learner" or "weak classifier" and is typically a decision tree with restricted growth, aiming to avoid over-generalization and focus on specific regions of the decision space. In this case, given any approximate solution $f^*_{m-1}(X)$, the function $\beta_m h(X; A_m)$ can be seen as the best greedy correction to the approximator function, adapting to the training sample provided. The function $h(X; A)$ can be considered as a direction, the steepest one, and the parameter β is the best value in that direction.

The value of the gradient function is calculated as follows:

$$-g_m(x_i) = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(X)=f^*_{m-1}(X)} \tag{8.17}$$

This gradient is defined only for the training data sample $\{(x_i, y_i) : i \in \{1, \ldots N\}\}$ and cannot be generalized to other (X, Y) values. One way to enable generalization is to select from the parameterized class the function $h(X; A_m)$ that is most parallel to $-g_m \in \mathbb{R}^N$. This is the function from the parameterized function space that correlates the most with $-g_m(X)$ over the data distribution. It can be obtained from the following expression:

$$A_m = \arg \min_{A, \beta} \sum_{i=1}^{N} (-g_m(x_i) - \beta h(x_i; A))^2 \tag{8.18}$$

This parameter A_m can be used to calculate $h(x; A_m)$ instead of using $-g_m(x)$, which is the one used in the steepest-descent strategy. Specifically, the line search is performed as follows:

$$\rho_m = \arg \min_{\rho} \sum_{i=1}^{N} L(y_i, f^*_{m-1}(x_i) + \rho h(x_i; A_m)) \tag{8.19}$$

With this calculation, the next approximation to the real function can be obtained by updating the previous one as follows:

$$f^*_m(X) = f^*_{m-1}(X) + \rho_m h(X; A_m) \tag{8.20}$$

This enables us to transform the problem into a least-squares minimization problem (Equation 8.19), followed by a single parameter optimization (Equation 8.20). This implies that for any $h(X; A)$ for which a feasible least-squares algorithm exists to solve Equation 8.19, we can use this approach to minimize any differentiable loss $L(Y, f)$ using our additive model. This leads to the following algorithm, using the steepest-descent algorithm:

Algorithm 5: Gradient Boost.

Input: L, $\{(x_i, y_i) : i \in \{1, \ldots, N\}\}$, M, h

1 $f_0^*(X) = \arg\min_\rho \sum_{i=1}^N L(y_i, \rho)$
2 **for** $m \in \{1, \ldots, M\}$ **do**
3 $\quad \tilde{y}_i = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(X)=f_{m-1}^*(X)}, \forall i \in \{1, \ldots, N\}$
4 $\quad A_m = \arg\min_{A,\beta} \sum_{i=1}^N (\tilde{y}_i - \beta h(x_i; A))^2$
5 $\quad \rho_m = \arg\min_\rho \sum_{i=1}^N L(y_i, f_{m-1}^*(x_i) + \rho h(x_i; A_m))$
6 $\quad f_m^*(X) = f_{m-1}^*(X) + \rho_m h(X; A_m)$
7 **end**
8 **return** f_M^*

If the weak model h is a decision tree, this algorithm is known as "Gradient Boosted Decision Tree" (GBDT). The resulting model f_M^* is an ensemble model containing weak decision tree classifiers ($\{h(X, A_m) : m \in \{0, \ldots, M\}\}$), which are weighted ($\{\rho_m : m \in \{0, \ldots, M\}, \rho_0 = 1\}$) and then summed together to obtain the final model.

In line 4 of this algorithm, the estimation of the parameters is done by a least-squares optimization procedure, but it is not the only procedure that could be used. It is used by default as it has good computational properties and can generally be done efficiently, but any other method could also be employed to accomplish the same task of finding the best parameters for the weak classifier.

8.1.4. Regularization

The process of fitting training data helps us find a function that approximates the data well, assuming that the function to predict is smooth enough. We can achieve good results up to a certain point. Overfitting is a common problem characterized by fitting too closely to the training data, which can hinder the generalization ability of the learned function to data outside the training set. To address this issue, various techniques have been proposed, known as "regularization techniques".

In the context of additive models, the number of components M may seem like the most natural regularization parameter. A higher value of M typically results in more accurate models for the training data, as the expectation of the loss function decreases. However, using fewer terms for regularization may lead to better results. Nonetheless, there are studies indicating that other regularization techniques often perform better than this approach [Copas, 1983]. Particularly, one technique involves adding a regularization parameter ν, typically referred to as the learning rate, during each boost of the function, as follows:

$$f_m^*(X) = f_{m-1}^*(X) + \nu \rho_m h(X; A_m), 0 < \nu \leq 1 \tag{8.21}$$

This technique is known as "shrinkage", and with it, there are two direct regularization parameters in the model, M and ν. Both parameters are directly related, as lower values for ν cause the updates to affect the model less, potentially increasing the optimal value for M. The tradeoff between these two parameters has been studied and documented [Friedman, 2001].

8.1.5. Example of Gradient Boosting algorithm: XGBoost algorithm

As an example of a powerful algorithm based on the gradient boosting ideas described before using GBDT, we have the outstanding eXtreme Gradient Boosting, or XGBoost algorithm, created in 2016 [Chen and Guestrin, 2016]. It is a popular and powerful algorithm based on GBDT, known for its speed, scalability, and performance on large datasets. Some of its main features will now be listed:

- **Regularization and pruning**: XGBoost includes regularization parameters to control overfitting. It introduces $L1/L2$ penalties, calculated using the residuals of the leaves [Johnson and Zhang, 2014].
- **Handles sparse data**: XGBoost can efficiently handle sparse datasets using the weighted quantile sketch algorithm.
- **Parallel learning and out-of-core computing**: The algorithm features a block structure for parallel learning, which allows it to scale using multicore machines or clusters. It also supports out-of-core computing, using data structures based on disk storage rather than memory storage, enabling it to handle very large datasets.
- **Optimization parameters**: XGBoost includes a set of parameters that can help optimize the learning procedure, controlling all the aspects mentioned above, apart from others.

8.2. LightGBM algorithm

LightGBM, developed by Microsoft, is an open-source gradient boosting algorithm based on GBDT, designed for efficiency with large datasets [Ke et al., 2017]. Introduced in 2017, its primary goal is to enhance training speed and predictive performance in gradient boosting algorithms. Widely adopted, LightGBM has achieved state-of-the-art results across various domains. It incorporates scalability enhancements that enable its use in diverse scenarios, including big data, sparse data, efficient handling of instance weights, and leveraging parallel and distributed computation.

Its main features are as follows:

- **Histogram-based split points**: The main cost when training a GBDT is learning the decision trees, with the most time-consuming aspect being the identification of the optimal split points. To tackle this issue, a method based on histograms is employed. This method bins continuous features into discrete bins and constructs feature histograms from these bins to determine the best split points. This approach is significantly more memory-efficient and accelerates the training process.
- **Gradient-based One-Side Sampling (GOSS)**: A new method for instance sampling is proposed. This method assigns a weight to each instance based on its gradient.

Instances with small gradients typically have low error, indicating that the model is already well-trained on them. However, removing all such instances could alter the data structure. To address this, the algorithm retains the top $\alpha\%$ instances with the highest gradients. Instead of discarding the remaining instances, it randomly subsamples a proportion β from the remaining $(100 - \alpha)\%$. This approach enhances model performance by reducing the number of instances while intelligently selecting which instances to keep, without significantly compromising model generalization capabilities.

- **Exclusive Feature Bundling (FEB)**: This feature is crucial for large-scale and sparse datasets. It involves reducing the number of features by bundling them together with minimal loss. These feature bundles are created for mutually exclusive features, which ultimately produce the same histograms for splits. However, there are two main issues:
  - **Greedy bundling**: Determining which features should be bundled together is an NP-hard problem, as it can be reduced to the graph coloring problem. Therefore, a greedy strategy is employed. Additionally, in the final implementation, a low conflict rate is assumed to increase further bundling, thereby enhancing algorithm performance without significantly sacrificing accuracy.
  - **Bundling formation**: In order to merge features together to create the same histogram, a straightforward yet effective approach is to apply an offset to each or some features. This ensures that values from different features fall into different bins of the histogram. For example, if one feature takes values in the range $[0, 1)$ and another in the range $[0, 2)$, an offset can be added to the second feature so that it takes values in the range $[1, 3)$ to achieve this property.

  Using this approach, sparse features can be bundled together, eliminating lots of zero-valued computations and thus increasing the overall performance of the model when this kind of features exist. While this can be highly advantageous for high-dimensional sparse datasets, it may degrade performance for small or non-sparse datasets. Despite this issue, the calculation remains relatively efficient ($\mathcal{O}(z)$, where z represents non-zero data).

- **Leaf-wise tree growth**: LightGBM employs this strategy instead of depth-wise tree growth, used by C4.5, CART, or XGBoost. This approach expands the leaf with the maximum loss reduction, rather than attempting to expand all nodes at the same level in each step. While it can reduce loss more effectively than depth-wise growth, it may lead to overfitting by producing more complex and deeper trees. However, the complexity can be controlled using regularization parameters, which help limit overfitting.

In addition to its optimized speed, efficient handling of large datasets, and memory efficiency, LightGBM's implementation also integrates high parallelism and distributed learning, along with a wide array of parameters for tuning all these features. Its performance in terms of speed and accuracy ranks among the top in current data science competitions and it is widely employed in numerous real-world applications [Xing et al., 2024, Zhuang et al., 2024, Truong et al., 2024].

In summary, LightGBM is a powerful algorithm based on GBDT, with features that help it adapt to nearly any situation in terms of speed, accuracy, and scalability. For these reasons, it has been chosen as the base method for our latest algorithm.

8.3. The log loss function

We will now discuss the log loss function. This is a widely used loss function in the context of binary classification.

The log loss function has the following structure:

$$L(y,x) = \log\text{loss}(y,x) = -\left(y\ln\left(\frac{1}{1+e^{-x}}\right) + (1-y)\ln\left(1-\frac{1}{1+e^{-x}}\right)\right) \tag{8.22}$$

The value y is expected to take values in $\{0,1\}$, while the value of x is expected to take values in $\mathbb{R}$. Inside the log loss function, another crucial function is the logistic function, which is a sigmoid function. It is defined as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{8.23}$$

It is important to highlight some good properties of this function, such as being strictly monotonic as well as:

$$\lim_{x\to-\infty}\sigma(x) = 0,\ \sigma(0) = \frac{1}{2},\ \lim_{x\to\infty}\sigma(x) = 1 \tag{8.24}$$

So this function maps values from $\mathbb{R}$ to the range $[0,1]$, which can be interpreted as a probability or degree of certainty of belonging to the positive class. With this definition, we can redefine the log loss function in terms of y and σ as follows:

$$L(y,\sigma) = -\left(y\ln\sigma + (1-y)\ln(1-\sigma)\right) \tag{8.25}$$

Additionally, the derivative of the logistic function can be calculated as follows:

$$\frac{\partial\sigma(x)}{\partial x} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}}\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) = \sigma(x)(1-\sigma(x)) \tag{8.26}$$

Using the logistic function as an intermediate function, we can calculate both the derivative and Hessian functions of our loss function with respect to x using the chain rule:

$$\begin{gathered}\frac{\partial L(y,x)}{\partial x} = \frac{\partial L(y,\sigma)}{\partial\sigma}\frac{\partial\sigma(x)}{\partial x} = \frac{\partial L(y,\sigma)}{\partial\sigma}\sigma(x)(1-\sigma(x)) = \\ \left(-\frac{y}{\sigma(x)} + \frac{1-y}{1-\sigma(x)}\right)\sigma(x)(1-\sigma(x)) = -y(1-\sigma(x)) + (1-y)\sigma(x) = \sigma(x) - y\end{gathered} \tag{8.27}$$

And for the second-order derivative:

$$\frac{\partial^2 L(y,x)}{\partial^2 x} = \frac{\partial}{\partial x}\frac{\partial L(y,x)}{\partial x} = \frac{\partial}{\partial x}(\sigma(x)-y) = \frac{\partial}{\partial x}\sigma(x) = \sigma(x)(1-\sigma(x)) \tag{8.28}$$

By employing log loss as our loss function, defining a LightGBM model and finding optimized models for this metric is straightforward. Log loss is a natural and interpretable loss function with very simple first- and second-order derivatives, which enhance performance.

8.4. Description of the algorithm

In order to define a fairness-aware loss function, we will create a continuous extension of our classical fairness functions. Let us begin with the difference in the False Positive Rate (FPR) metric. We need to consider that the classical metric is defined by the following expression:

$$\mathrm{FPR}_{\mathrm{diff}} = \left| \frac{\mathrm{FP}_1}{\mathrm{FP}_1 + \mathrm{TN}_1} - \frac{\mathrm{FP}_0}{\mathrm{FP}_0 + \mathrm{TN}_0} \right| \tag{8.29}$$

This metric is used for classification, where our predicted values are either 0 or 1. However, in this case, we have a continuous output. We can consider the values of our prediction after applying the logistic function $\sigma(x)$ as a score that resembles the probability of that sample belonging to the positive class. However, it does not necessarily have to be treated as a strict probability, as we do not impose any probability function constraint. With these scores, we can generalize these functions and others by first extending the concepts of TP, FP, TN and FN.

We will use FP as an example to illustrate this. To define FP in classical classification terms, we will use an auxiliary indicator function, $\mathbb{1}_{i,j,k}$:

$$\mathbb{1}_{i,j,k}(x,y,p) = \begin{cases} 1 & \text{if } x = i \wedge y = j \wedge p = k \\ 0 & \text{in any other case} \end{cases} \tag{8.30}$$

We can subsequently define the functions $\mathbb{1}_{i,j,-}$, $\mathbb{1}_{i,-,k}$ and $\mathbb{1}_{-,j,k}$ analogously, without considering the values of p, y and x respectively. For instance, $\mathbb{1}_{i,j,-}$ will be defined as:

$$\mathbb{1}_{i,j,-}(x,y,p) = \mathbb{1}_{i,j,-}(x,y) = \begin{cases} 1 & \text{if } x = i \wedge y = j \\ 0 & \text{in any other case} \end{cases} \tag{8.31}$$

Using this definition, we can express FP as a function as follows: Let x_i be the prediction for sample i, y_i its actual class, and p_i the value of its protected attribute. Let $X = (x_1, \ldots, x_N)$ be the column vector of predicted values, $Y = (y_1, \ldots, y_N)$ the column vector of true outputs, and $P = (p_1, \ldots, p_N)$ the column vector of protected attributes. FP is then defined as:

$$\mathrm{FP}(X,Y) = \sum_{i=0}^{N} \mathbb{1}_{1,0,-}(x_i, y_i) \tag{8.32}$$

The sum just defined gives us the number of false positives within our sample in a classification context. Now, we aim to extend this definition to problems involving classification scores. If X represents the column vector of scores instead of the predicted values, we define FP as follows:

$$\mathrm{FP}(X,Y) = \sum_{i=0}^{N} \mathbb{1}_{-,0,-}(y_i) x_i \tag{8.33}$$

To illustrate the previous expression, let us consider an example. Suppose that a score of 0.2 is assigned to an instance whose real class is 0. In this scenario, there is an error of 0.2 with respect to the actual class. This is because we are predicting it belongs to class 1 with a score of 0.2, whereas the prediction score for class 0 is 0.8. Using Equation 8.33, the FP value in this case would be 0.2. In our context, when using the log loss function, prediction scores

are obtained after applying the logistic function to the values of x, as explained in Section 8.3.

Let us now consider a vectorized definition of FP, TP, FN, and TN using the σ function vector-wise, which will help us calculate the derivative and Hessian of the loss function about to be defined. Considering a vector of tuples $((x_1, y_1, p_1), \ldots, (x_N, y_N, p_N))$, then we can define the extension of the σ function vector-wise as follows: $\mathbb{1}_{i,j,k}(((x_1, y_1, p_1), \ldots, (x_N, y_N, p_N))) = (\mathbb{1}_{i,j,k}(x_1, y_1, p_1), \ldots, \mathbb{1}_{i,j,k}(x_N, y_N, p_N))$. The rest of $\mathbb{1}$ function extensions are defined analogously. Using Equation 8.33, we can consider a vector version of that definition:

$$\text{FP}(X, Y) = \mathbb{1}_{-,0,-}(Y)^T X \tag{8.34}$$

In this definition, there is a matrix product (which is equivalent to a dot product) of values of $\mathbb{1}_{-,0,-}(Y)$ and X. What is important to note here is that $\mathbb{1}_{-,0,-}(Y)$ is constant with respect to the vector of scores X, which is the reason why we will use an abuse of notation writing $\mathbb{1}_{-,0,-}(Y)^T = \mathbb{1}^T_{-,0,-}$. Using this notation, the final expression of our metrics will be:

$$\text{FP}(X, Y) = \mathbb{1}^T_{-,0,-} X, \text{TP}(X, Y) = \mathbb{1}^T_{-,1,-} X, \text{FN}(X, Y) = \mathbb{1}^T_{-,1,-}(1 - X), \text{TN}(X, Y) = \mathbb{1}^T_{-,0,-}(1 - X) \tag{8.35}$$

Using this notation, FPR difference can be written in vectorial form using scores as follows:

$$\text{FPR}_{\text{diff}}(X, Y, P) = \left| \frac{\mathbb{1}^T_{-,0,1} X}{\mathbb{1}^T_{-,0,1} X + \mathbb{1}^T_{-,0,1}(1 - X)} - \frac{\mathbb{1}^T_{-,0,0} X}{\mathbb{1}^T_{-,0,0} X + \mathbb{1}^T_{-,0,0}(1 - X)} \right| \tag{8.36}$$

There is a key observation here: $\mathbb{1}^T_{-,0,1} X + \mathbb{1}^T_{-,0,1}(1 - X) = \mathbb{1}^T_{-,0,1} 1$, where 1 represents the N-dimensional vector where all values are 1. This expression is indeed equivalent to the amount of instances where $y_i = 0$ and $p_i = 1$. Using this property, the final expression for our function is:

$$\text{FPR}_{\text{diff}}(X, Y, P) = \left| \frac{\mathbb{1}^T_{-,0,1} X}{\mathbb{1}^T_{-,0,1} 1} - \frac{\mathbb{1}^T_{-,0,0} X}{\mathbb{1}^T_{-,0,0} 1} \right| \tag{8.37}$$

Another important property of this definition is the value of its derivative. In order to calculate it, we must consider that X is a vector whose components are always non-negative, as are outputs of the σ function. Using this fact, the derivative is calculated as follows:

$$\begin{aligned}
\frac{\partial \text{FPR}_{\text{diff}}(X, Y, P)}{\partial X} &= \frac{\partial}{\partial X}\left(\left| \frac{\mathbb{1}^T_{-,0,1} X}{\mathbb{1}^T_{-,0,1} 1} - \frac{\mathbb{1}^T_{-,0,0} X}{\mathbb{1}^T_{-,0,0} 1} \right| \right) = \frac{\partial}{\partial X}\left(\left| \frac{(\mathbb{1}^T_{-,0,0} 1 \mathbb{1}^T_{-,0,1} - \mathbb{1}^T_{-,0,1} 1 \mathbb{1}^T_{-,0,0}) X}{\mathbb{1}^T_{-,0,1} 1 \mathbb{1}^T_{-,0,0} 1} \right| \right) \\
&= \frac{1}{\mathbb{1}^T_{-,0,1} 1 \mathbb{1}^T_{-,0,0} 1} \frac{\partial}{\partial X}(|(\mathbb{1}^T_{-,0,0} 1 \mathbb{1}^T_{-,0,1} - \mathbb{1}^T_{-,0,1} 1 \mathbb{1}^T_{-,0,0}) X|) \\
&= \frac{|\mathbb{1}^T_{-,0,0} 1 \mathbb{1}_{-,0,1} - \mathbb{1}^T_{-,0,1} 1 \mathbb{1}_{-,0,0}|}{\mathbb{1}^T_{-,0,1} 1 \mathbb{1}^T_{-,0,0} 1} = \left| \frac{\mathbb{1}_{-,0,1}}{\mathbb{1}^T_{-,0,1} 1} - \frac{\mathbb{1}_{-,0,0}}{\mathbb{1}^T_{-,0,0} 1} \right| = C
\end{aligned} \tag{8.38}$$

Which is a constant vector with respect to X (but not with respect to Y or P), so the second-order derivative with respect to X will be equal to the vector 0. Furthermore, despite the function being inside an absolute value, it is differentiable for any value of X since all its terms are non-negative. These properties are very advantageous for its incorporation into the loss function, considering that it needs to be differentiated.

Taking all of this into account, we are now able to define our fairness-aware loss function using the extension of the FPR_{diff} metric. The function we will use is the following:

$$L_f(Y, X, P) = -(1-\lambda)k\left(Y \ln\left(\frac{1}{1+e^{-X}}\right) + (1-Y)\ln\left(1-\frac{1}{1+e^{-X}}\right)\right) + \lambda\left|\frac{\mathbb{1}^T_{-,0,1}\frac{1}{1+e^{-X}}}{\mathbb{1}^T_{-,0,1}1} + \frac{\mathbb{1}^T_{-,0,0}\frac{1}{1+e^{-X}}}{\mathbb{1}^T_{-,0,0}1}\right| \tag{8.39}$$

Where $\lambda \in [0,1]$. $k = \frac{-1}{\ln 0.5}$ is a normalization factor. Although the first summand is not upper bounded, the value obtained by a "dummy" classifier is $\ln 0.5$ ($X = 0 \Rightarrow \frac{1}{1+e^{-X}} = 1 - \frac{1}{1+e^{-X}} = 0.5$), so $\frac{-1}{\ln 0.5}$ will be the value considered to "normalize" the first summand. L_f can be defined in terms of the logistic function (using Σ instead of σ following vector notation), as follows:

$$\begin{aligned} L_f(Y, \Sigma, P) &= -(1-\lambda)k\left(Y \ln \Sigma + (1-Y)\ln(1-\Sigma)\right) + \lambda\left|\frac{\mathbb{1}^T_{-,0,1}\Sigma}{\mathbb{1}^T_{-,0,1}1} - \frac{\mathbb{1}^T_{-,0,0}\Sigma}{\mathbb{1}^T_{-,0,0}1}\right| \\ &= k(1-\lambda)L(Y, \Sigma) + \lambda \text{FPR}_{\text{diff}}(\Sigma, Y, P) \end{aligned} \tag{8.40}$$

Considering $L(Y, \Sigma)$ as the vectorized version of the logistic function discussed in Section 8.3. This function exhibits a similar structure to that used in the FDT algorithm, constituting a convex combination between the traditional loss function $L(Y, \Sigma)$, and the desired fairness criterion, here represented by $\text{FPR}_{\text{diff}}(\Sigma, Y, P)$. While this definition may appear straightforward, it is essential to remember that it requires a continuous extension definition of the fairness criterion. Moreover, this definition proves highly advantageous for computing the first- and second-order derivatives. The first-order derivative is calculated as follows:

$$\begin{aligned} \frac{\partial L_f(Y, \Sigma, P)}{\partial X} &= \frac{\partial L_f(Y, \Sigma, P)}{\partial \Sigma}\frac{\partial \Sigma(X)}{\partial X} \\ &= \left((1-\lambda)k\frac{\partial L(Y, \Sigma, P)}{\partial \Sigma} + \lambda\frac{\partial \text{FPR}_{\text{diff}}(\Sigma, Y, P))}{\partial \Sigma}\right)\frac{\partial \Sigma(X)}{\partial X} \\ &= \left((1-\lambda)k\left(-\frac{Y}{\Sigma(X)} + \frac{1-Y}{1-\Sigma(X)}\right) + \lambda C\right)(\Sigma(X)(1-\Sigma(X)) \\ &= (1-\lambda)k(\Sigma(X) - Y) + \lambda C\Sigma(X)(1-\Sigma(X)) \end{aligned} \tag{8.41}$$

This represents yet another convex combination, this time between the derivative of the log loss function and the derivative of the FPR_{diff} function. The second-order derivative calculation can be done as follows:

$$\begin{aligned} \frac{\partial^2 L_f(Y, \Sigma, P)}{\partial^2 X} &= \frac{\partial}{\partial X}((1-\lambda)k(\Sigma(X) - Y) + \lambda C\Sigma(X)(1-\Sigma(X))) \\ &= (1-\lambda)k(\Sigma(X)(1-\Sigma(X)) + \lambda C\left((\Sigma(X)(1-\Sigma(X))^2 - \Sigma(X)^2(1-\Sigma(X))\right) \end{aligned} \tag{8.42}$$

We can directly integrate these expressions into a LightGBM algorithm to be able to use this new fairness-aware loss function. This is what constitutes the Fair LightGBM algorithm.

8.5. SWOT Analysis

The SWOT analysis for FLGBM algorithm is presented in Figure 8.1.

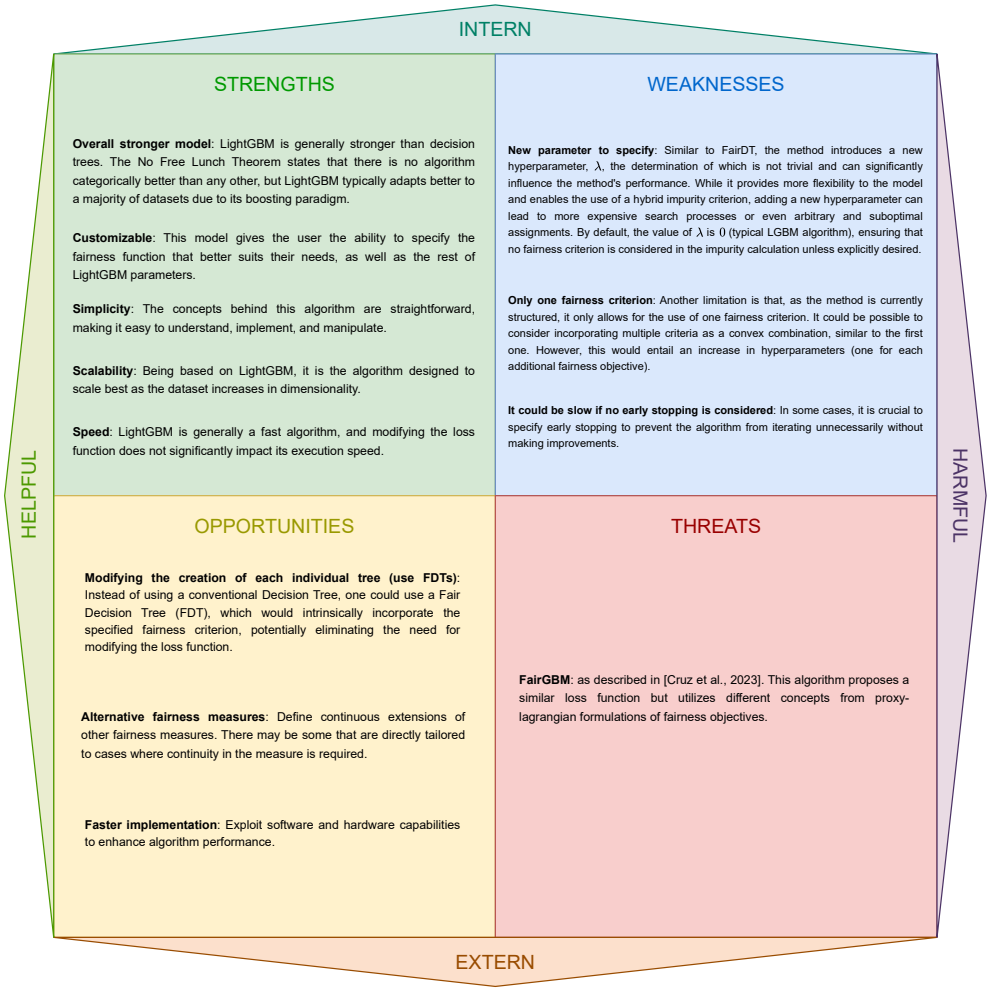


Figure 8.1.: SWOT Analysis for the FLGBM algorithm. The references for the threat algorithms are as follows: [Cruz et al., 2023].

Part IV.

Experimentation

In this part, the experimentation conducted to test the presented algorithms will be discussed, including dataset descriptions and implemented code.

9. Experimental framework

In this chapter, we will discuss the experimental framework used to test the developed algorithms, the methodology employed, and how the results will be evaluated.

In order to test all of these algorithms, we will need to define an experimental setup that enables us to explore and further understand their individual capabilities. To this end, we will provide a detailed explanation of all datasets to which these methods have been applied, along with our experimental framework and the tests conducted on them.

9.1. Overview of the experimentation

The experimentation will involve testing all three developed algorithms in a controlled environment and conducting individual and comparative studies of each of them. Additionally, we will include a baseline control algorithm for comparison, which is the decision tree (DT). Therefore, in total, we will be using four algorithms.

To conduct these tests, we will select 10 datasets on which each of the algorithms will be tested. These datasets will vary in terms of dimensions, context, and content, allowing us to address a diverse range of cases.

Let us remember that we are in a multiobjective optimization context where we have one dataset to split into training, validation, and test sets. Therefore, we cannot run each algorithm just once for each dataset; we must run them several times with different data partitions to find an average Pareto-optimal set of solutions.

To find these Pareto-optimal solutions using the given training, validation, and test sets, it is proposed to include each algorithm within a process of searching for these solutions. To do this, we will divide our algorithms into two groups:

- **DT, FairDT, and FairLGBM algorithms**: These models will be included in a hyperparameter optimization process using the genetic multiobjective algorithm NSGA-II described in Section 5.4. Each individual in the solution population will be a binary classifier, represented by a set of hyperparameters for the respective learning algorithm. The decision space will then consist of different values for these learning algorithms' hyperparameters. Given this, the goal is to find the set of hyperparameters which, after being applied to the learning algorithm using the training dataset, return models that achieve the best balance among the considered objectives on the validation dataset.

- **Fair Genetic Pruning algorithm**: A multi-objective genetic search process is inherently applied, as proposed in Chapter 7, with criteria for representing individuals, creating the initial population, crossover, mutation, and replacement outlined in their respective sections. After generating the matrix tree with the training set, the aim will be to

generate pruned trees that achieve the best balance among the considered objectives on the validation set.

For all of these methods, the size of the population n_i will be 150, while the number of generations will be $n_g = 300$. The probability of crossover $p_c = 1$, and the probability of mutation $p_M = 0.3$.

9.2. Datasets

Datasets are a key part of our experimentation, as we are focusing on a specific problem that can be found in a wide range of different contexts and situations. Therefore, it is crucial for us to select and identify datasets where discrimination can occur, along with the protected attributes they contain that may be subject to discrimination.

To find a dataset with these characteristics, one cannot arbitrarily choose a dataset and select an attribute that might apparently lead to unfairness. A study must be conducted on the influence of such attributes on the specific problem, typically requiring professionals in humanitarian or sociological fields. Therefore, it is beyond our scope to choose any protected attribute, and thus, datasets that have been previously studied and known to have some form of injustice will be used [Fabris et al., 2022].

Datasets selected for this study can be found in the GitHub repository `https://github.com/juliettm/datasets`, which, as of January 30, 2024, is a private repository. It contains datasets that are not open for public use and can only be accessed by explicit request to the authors. Access can be requested and will be granted for research purposes. Nevertheless, all selected datasets can be found in public sources. For the evaluation of this work, only the datasets in their preprocessed and anonymized versions will be provided. Our goal is not to deeply describe and understand each particular dataset and its fairness implications, but it is important to at least understand their context. For that reason, we will now outline the prediction task and provide some other useful information about the datasets used in our study:

- **Adult dataset**:
  - **Dataset description**: This dataset contains information about U.S. citizens gathered during the March 1994 U.S. Current Population Survey. It includes demographic and socioeconomic dimensions, with features such as education, profession, age, sex, race, personal condition and financial condition. It is commonly used as a benchmark dataset for fairness in machine learning.
  - **Prediction task**: Predict whether a given individual earns above $50,000 annually or not.
  - **Protected attribute**: Race.
  - **Dataset shape after preprocessing**: 45222 instances and 14 attributes.
  - **Additional references**: [Becker and Kohavi, 1996]
- **Compas dataset**:

  - **Dataset description**: This dataset contains information about the COMPAS algorithm and its application to citizens of Broward County, Florida, during a time period ranging from 2013 to 2014. Selected attributes for use in our experiments were chosen as outlined in [Friedler et al., 2019], which are the following: sex, age, age_cat, race, juv_fel_count, juv_misd_count, juv_other_count, priors_count, c_charge_degree, c_charge_desc, decile_score, and score_text. It is commonly used as a benchmark dataset for fairness in machine learning.
  - **Prediction task**: Predict whether a certain individual will recidivate (and may the police notice that recidivism) within a time period ranging from their release until 2 years afterward.
  - **Protected attribute**: Race.
  - **Dataset shape after preprocessing**: 5278 instances and 9 attributes.
  - **Additional references**: [Julia et al., 2016] `https://github.com/propublica/compas-analysis`

- **Diabetes dataset**:
  - **Dataset description**: This dataset contains records of patients diagnosed with diabetes who underwent laboratory tests, received medications, and stayed for up to 14 days in hospitals. The data was collected from 1999 to 2008 at 130 US hospitals, including information from their integrated delivery networks. It includes demographic information, diagnoses, and other types of hospital procedures.
  - **Prediction task**: Predict if a patient will be readmitted within 30 days of discharge.
  - **Protected attribute**: Sex.
  - **Dataset shape after preprocessing**: 46176 instances and 16 columns.
  - **Additional references**: [Clore and Strack, 2014, Strack et al., 2014]

- **Dutch dataset**:
  - **Dataset description**: This dataset contains information derived from the 2001 census conducted by the Dutch Central Bureau for Statistics to collect data about family composition, economic activities, levels of education, and occupation of Dutch citizens and foreigners from various countries of origin.
  - **Prediction task**: Predict whether a certain individual has a high-income or low-income profession.
  - **Protected attribute**: Sex.
  - **Dataset shape after preprocessing**: 60420 instances and 10 attributes.
  - **Additional references**:
    [Centraal Bureau voor de Statistiek (CBS) (Statistics Netherlands), 2018]

- **German dataset**:
  - **Dataset description**: This dataset contains financial information of loan applicants from 1973 to 1975. It includes details about their financial situation, credit history, and personal situation.

  - **Prediction task**: Predict whether a given individual has a low or high risk of repaying a certain loan.
  - **Protected attribute**: Age.
  - **Dataset shape after preprocessing**: 1000 instances and 10 attributes.
  - **Additional references**: [Hofmann, 1994]

- **Insurance dataset**:
  - **Dataset description**: This dataset contains information from a popular Italian car insurance comparison website, including quotes provided by nine companies, collected in 2020. It includes information such as gender, age, vehicle details, and a summary of claim history.
  - **Prediction task**: Predict whether the insurance charges are above or below 40000€.
  - **Protected attribute**: Sex.
  - **Dataset shape after preprocessing**: 1338 instances and 9 attributes.
  - **Additional references**: [Fabris et al., 2021]

- **Obesity dataset**:
  - **Dataset description**: This dataset includes data for estimating obesity levels in individuals from the countries of Mexico, Peru, and Colombia, based on their eating habits and physical condition.
  - **Prediction task**: Predict whether the individual is obese (with obesity types I, II, or III) or not (with insufficient weight, normal weight, overweight level I, or II).
  - **Protected attribute**: Gender.
  - **Dataset shape after preprocessing**: 2111 instances and 23 attributes.
  - **Additional references**: [obe, 2019]

- **Parkinson dataset**:
  - **Dataset description**: This dataset contains biomedical voice measurements collected at home from 42 individuals with early-stage Parkinson's disease who were recruited for a six-month trial of a telemonitoring device for remote symptom progression monitoring. Attributes include information such as subject age, gender, and recruitment date.
  - **Prediction task**: Predict whether the total UPDRS score for a certain patient's voice recording is below 17.1 or not.
  - **Protected attribute**: Sex.
  - **Dataset shape after preprocessing**: 5875 instances and 19 columns.
  - **Additional references**: [Tsanas and Little, 2009]

- **Ricci dataset**:
  - **Dataset description**: This dataset contains information on 118 New Haven (Connecticut) firefighter promotion tests, including the scores and race of the individuals. The case escalated to the US Supreme Court labor case on discrimination, Ricci vs. DeStefano (2009), due to disparate impact.

  - **Prediction task**: Predict whether a given individual will pass a promotion exam (scoring at least 70 points) or not.
  - **Protected attribute**: Race.
  - **Dataset shape after preprocessing**: 118 instances and 5 attributes.
  - **Additional references**: [pro, 2009], https://github.com/algofairness/fairness-comparison/tree/master/fairness/data/raw

- **Student dataset**:
  - **Dataset description**: This dataset contains information on student achievement from two Portuguese secondary schools, collected using questionnaires and school reports. The data attributes include student grades, demographic, social, and school-related features.
  - **Prediction task**: Predict whether the individual's final year grade during the 3rd term for the Portuguese subject will be greater or lower than 12.
  - **Protected attribute**: Sex.
  - **Dataset shape after preprocessing**: 649 instances and 39 attributes.
  - **Additional references**: [Cortez, 2014]

These datasets belong to diverse fields such as finance, legal, labor, and medicine, collectively constituting an arguably robust set for testing our algorithms. They exhibit a good variety in terms of the number of instances, attributes, and content. Although none of them are excessively large, they offer a diverse range of datasets to evaluate our algorithms. A scatterplot showing the number of features and samples for each dataset can be seen in Figure 9.1.

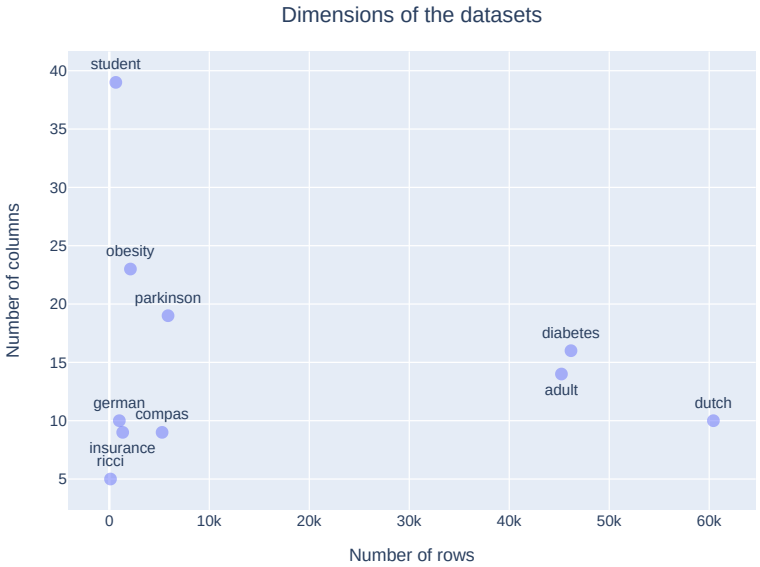


Figure 9.1.: Number of rows and columns for all datasets included in the experimentation process.

All datasets were preprocessed uniformly, meaning that the same principles were applied during the preprocessing of all datasets. Our objective is not to conduct an extensive preprocessing to achieve the best possible results for each dataset, but rather to preprocess them consistently to observe general behavioral patterns in our algorithm executions. This approach helps us determine the effectiveness of the algorithms for their proposed tasks. The preprocessing strategy we followed is similar to that proposed in [Valdivia et al., 2020]. It involves selecting attributes that contribute to good results, transforming categorical attributes into numerical ones, and converting binary labels into 0 or 1 values. If categorical attributes have an implicit order, they are numerically encoded; otherwise, a one-hot encoding set of attributes is generated.

9.3. Decision space, hyperparameters

The decision space cannot be standardized as a common set across all methods because it varies depending on the model chosen and the parameters that control its learning process. Therefore, the parameters considered for the optimization process (for algorithms using the NSGA-II hyperparameter optimization process, given that for Fair Genetic Pruning the decision space is the space of all possible pruning sets) will now be explained:

- **DT algorithm**:
  - **criterion**: This parameter represents the function that measures the quality of the split of an intermediate node in the decision tree. The decision tree generates new nodes from those already present at lower levels as this value becomes smaller. There are two possible values: "gini" and "entropy".
  - **max_depth**: This parameter specifies the maximum depth that the tree can reach. Deeper trees tend to be more complex. The possible values it can take are $\{n \in \mathbb{N} : n \leq m\}$, where m is the depth of the tree learned from the training sample without any growth restriction. (This tree will be the first model trained and it will belong to the initial population, so this value can be set from the beginning).
  - **min_samples_split**: This parameter specifies the minimum number of samples required to split a node. A larger value makes it more difficult for a node to split. The possible values considered are $\{n \in \mathbb{N} : 2 \leq n \leq 40\}$.
  - **max_leaf_nodes**: This parameter sets the maximum number of leaf nodes that the tree can have. A larger value increases the complexity of the tree, allowing for more diverse paths until reaching a final classification. The possible values are $\{n \in \mathbb{N} : n \leq l\}$, where l is the number of leaf nodes of the tree learned using the training sample without any growth restriction. (This tree will be the first model trained and it will belong to the initial population, so this value can be set from the beginning).
  - **class_weight**: This parameter assigns a weight to each class, which is useful for imbalanced datasets. The values it can take are $\left\{\frac{1}{n} : n \in \mathbb{N} \wedge 2 \leq n \leq 10\right\} \cup \{n \in \mathbb{N} : 1 \leq n \leq 10\}$. If this hyperparameter takes the value x it means that the negative class will have a weight of 1, while the positive class will have a weight of x (where $x = 1$ corresponds to equal weights).

- **FairDT algorithm**: Same parameters as Decision Trees (DT) with the following additional modifications/additions:
  - **criterion**: Instead of using "gini" and "entropy", it uses the newly created functions "gini_fair" and "entropy_fair". (Using gini_fair/entropy_fair with a fair_param of 0 is equivalent to using gini/entropy).
  - **fair_param**: This parameter controls the importance of fairness in the learning process. A low value could promote a fairer model, while a high value could overshadow the traditional impurity criterion, which ideally should lead the learning process. The possible values we will use are $\left\{\frac{n}{100} : n \in \mathbb{N} \wedge 0 \leq n \leq 100\right\}$.
- **FairLGBM algorithm**:
  - **num_leaves**: This parameter controls the complexity of the tree model. If set to $2^{\text{max\_depth}}$, it aims to obtain the same number of leaves as a depth-wise tree. However, in practice, as LightGBM uses leaf-wise trees, trees are typically much deeper than depth-wise trees for a fixed number of leaves. This parameter serves as a regularization parameter. Possible values we will use are $\{n \in \mathbb{N} : 2 \leq n \leq 62\}$.
  - **min_data_in_leaf**: This is another regularization parameter. Its optimal value highly depends on the number of training instances and num_leaves. It controls the minimum amount of data needed for a split to occur. Possible values we will use are $\{n \in \mathbb{N} : 0 \leq n \leq 20\}$.
  - **max_depth**: This parameter explicitly limits the maximum depth the tree can reach. Possible values that will be used for this parameter are $\{n \in \mathbb{N} : n \leq m\}$, where m is the maximum depth among all trees learned for an unrestricted LightGBM model in terms of size. (This LightGBM model will be the first model trained and it will belong to the initial population, so this value can be set from the beginning).
  - **learning_rate**: This parameter determines the step size at each iteration while moving toward the minimum of the loss function. Typically, a smaller learning rate requires more iterations to converge, resulting in a slower training process but potentially yielding a more accurate model. Possible values that will be used are $\left\{\frac{n}{100} : n \in \mathbb{N} \wedge 1 \leq n \leq 20\right\}$.
  - **n_estimators**: This parameter specifies the number of boosting stages to perform, and therefore the number of estimators trained. More estimators can enhance learning performance, but they can also increase the risk of overfitting. The selected values are $\{n \in \mathbb{N} : 50 \leq n \leq 200\}$.
  - **feature_fraction**: This parameter controls the fraction of randomly selected features used for training each individual tree. Possible values that will be used are $\{x \in \mathbb{R} : \frac{1}{\text{Num Features}} \leq x \leq 1\}$.
  - **class_weight**: This parameter assigns a weight to each class, which is useful for imbalanced datasets. The values it can take are $\left\{\frac{1}{n} : n \in \mathbb{N} \wedge 2 \leq n \leq 10\right\} \cup \{n \in \mathbb{N} : 1 \leq n \leq 10\}$. If this hyperparameter takes the value x it means that the negative class will have a weight of 1, while the positive class will have a weight of x (where $x = 1$ corresponds to equal weights). It internally uses the scale_pos_weight LightGBM parameter.

- **fair_param**: This parameter controls the importance of fairness in the learning process. A low value could promote a fairer model, while a high value could overshadow the traditional impurity criterion, which ideally should lead the learning process. The possible values we will use are $\left\{\frac{n}{100} : n \in \mathbb{N} \wedge 0 \leq n \leq 100\right\}$.

9.4. Objective space

The objective space will be defined by the functions we aim to optimize. These functions are intentionally chosen to have low correlation with each other. If they were highly correlated, optimizing one function would automatically optimize the others, potentially leading to a simplified model focusing on a single objective in that case.

For the study to be conducted, we will only consider the measures of inverted G-mean and difference in false positive rate (FPR_{diff}). These two objectives are chosen to identify models that not only achieve high classification accuracy but also maintain adherence to a specific fairness criterion.

However, several other objective functions can be considered. We will present the objective functions that were considered and implemented (even if not used in the final experimentation) across different categories: error objectives, fairness objectives, and interpretability objectives:

- **Error objectives**: We aim for our models to closely match the real values of the response attribute Y, minimizing errors in the predicted values.
  - **Error rate**: This is a classic error objective, defined as $\frac{\text{FP}+\text{FN}}{\text{P}+\text{N}}$. The goal is to minimize this objective function, which can take values in the range $[0, 1]$.
  - **Inverted G-mean** (1−G-mean): We will use the geometric mean criterion, which combines the measures of the true positive rate ($P[p = 1|Y = 1]$) and the true negative rate ($P[p = 0|Y = 0]$). By using the geometric mean criterion, we ensure that a significant improvement in this function requires a joint improvement in both measures. The G-mean measure is defined as $\sqrt{P[p = 1|Y = 1]P[p = 0|Y = 0]} = \sqrt{\text{TPR} \cdot \text{TNR}}$, and our aim is to maximize it. However, since we treat the problem as a minimization one, we will use inverted G-mean = $1 - \sqrt{\text{TPR} \cdot \text{TNR}}$. This function can take values in the range $[0, 1]$.
- **Fairness objectives**: We aim for our models to generate fair predictions without discrimination based on an individual's sensitive attribute. Since there are various mutually exclusive families of fairness, as we have already outlined, we will define several measures to address them.
  - **Difference in False Positive Rate (FPR_{diff})**: We aim to minimize the difference between false positives ($P[p = 1|Y = 0]$) conditioned on the value of both classes, i.e., the difference between $P[p = 1|Y = 0, A = 0]$ and $P[p = 1|Y = 0, A = 1]$. We will consider the absolute difference between the conditionings on each value of the sensitive attribute, i.e., $|P[p = 1|Y = 0, A = 0] - P[p = 1|Y = 0, A = 1]|$. Using an absolute difference instead of a relative one is preferable for several reasons: possible values for this measure are bounded to the range $[0, 1]$, and the minimum is reached when the difference is 0 (indicating the same probability

regardless of the value taken in the sensitive attribute). Additionally, considering a relative difference may equate values such as 0.02 and 0.01 with 0.5 and 1, which might not be preferable. This measure corresponds to the family of equalized odds.

- **Difference in Predictive Positive Value ($\mathbf{PPV_{diff}}$)**: We aim to minimize the difference in individuals who, having been predicted as positives, actually were positives ($P[Y = 1|p = 1]$), with respect to both classes, i.e., $P[Y = 1|p = 1, A = 1]$ and $P[Y = 1|p = 1, A = 0]$. Similar to before, we will consider the absolute difference, thus minimizing $|P[Y = 1|p = 1, A = 1] - P[Y = 1|p = 1, A = 0]|$. The minimum occurs when both probabilities are equal. This measure corresponds to the family of predictive rate parity. Possible values for this measure belong to the range $[0, 1]$.
- **Difference in Predictive Negative Rate ($\mathbf{PNR_{diff}}$)**: In this case, we aim to minimize the difference between the probabilities that an individual is predicted as negative ($P[p = 0]$), conditioned on whether the individual takes one value or another in the sensitive attribute, i.e., $P[p = 0|A = 0]$ and $P[p = 0|A = 1]$. Similar to before, we are interested in the absolute difference between the two: $|P[p = 0|A = 0] - P[p = 0|A = 1]|$. This measure corresponds to the demographic parity family. Possible values for this measure belong to the range $[0, 1]$.

- **Interpretability objectives** (not suitable for FairLGBM): We will aim for our models to be as simple as possible, as this greatly enhances its interpretability and understanding. This will allow users to comprehend it, enabling them to make decisions based on the method's outcomes consciously and responsibly.
  - **Number of leaves**: This is a measure that indicates the complexity and interpretability of the model quite well. The higher the number of leaves, the greater the number of decision paths that can be taken when classifying an individual, making the interpretation of each of these paths, or the tree as a whole, much more complex.
  - **Weighted average leaf depth based on the individuals falling into each leaf**: A higher depth for a particular leaf implies that it has passed through a greater number of intermediate decision processes (intermediate nodes), making the interpretation of that decision path more complicated. However, we are interested in knowing how many instances (during training or validation) fell into that leaf, as this will give us a relative importance of that leaf.If the majority of examples fell into shallow leaves, and only exceptions fell into very deep leaves, the overall complexity of the decision-making process would be lower than if the opposite were true.

After reviewing all implemented objectives, it is important to note several nuances. Not all objective functions are implemented for all algorithms: FDT supports all objectives, while FGP currently supports only error and fairness objectives, and FLGBM currently supports error objectives and FPR_{diff}. Any subset of these objectives can be used for our executions, enabling partial optimization studies or attempting complete optimization with all objectives. Due to the inherent challenges associated with increased dimensionality, it can be advantageous from an efficiency perspective to restrict the number of objectives considered necessary or relevant at any given moment.

9.5. Evaluation of models and runs

To evaluate the results obtained from the optimization process, we will implement a validation and testing system. Our dataset will be partitioned as follows: 75% of the data will be randomly selected for training, and the remaining 25% will be reserved for testing. This partition ratio is a standard practice in machine learning.

For validation purposes, a similar partitioning will be applied to the training data. Thus, 75% oof the training data will be used exclusively for model training, while the remaining 25% will be used for validation of the learned models. Therefore, the final distribution will be 56.25% of the total data for training, 18.75% for validation, and 25% for testing.

This partitioning will be executed using a specific random seed to ensure reproducibility of the experiments. The same seed will also control other random processes within the algorithms.

The use of a random seed has the potential to introduce bias into the results. Depending on the chosen seed, the dataset partition could exhibit a specific structure, leading to results that may not be representative of the general case. To mitigate this effect, 10 runs will be performed for each algorithm and dataset using different random seeds, ranging from 100 to 109.

Each run will return a set of Pareto-optimal solutions for the specified algorithm and dataset. These solutions are Pareto-optimal based on the objective function evaluated on the validation set. During the genetic optimization process, only the training data is used for training models, and the validation data is used for evaluating models. Once the Pareto-optimal set is obtained based on the validation set, these Pareto-optimal models will be tested using the test set.

After evaluating all these models from the 10 runs for each dataset with their respective test sets, all the solutions will be aggregated together, obtaining an average Pareto front using the test results. This approach ensures that we initially identify the best models during the search process using validation data, and then we obtain the final performance using the test results.

Additionally, quality indicators will be calculated using Pareto fronts found to ease the comparison among solutions obtained by different algorithms for the same dataset. Furthermore, various quality indicators will be evaluated for each generation of every run, including the runtime of each run and general structural measures of the algorithms identified in each generation.

The overall final structure of the experimentation conducted can be seen in Figure 9.2.

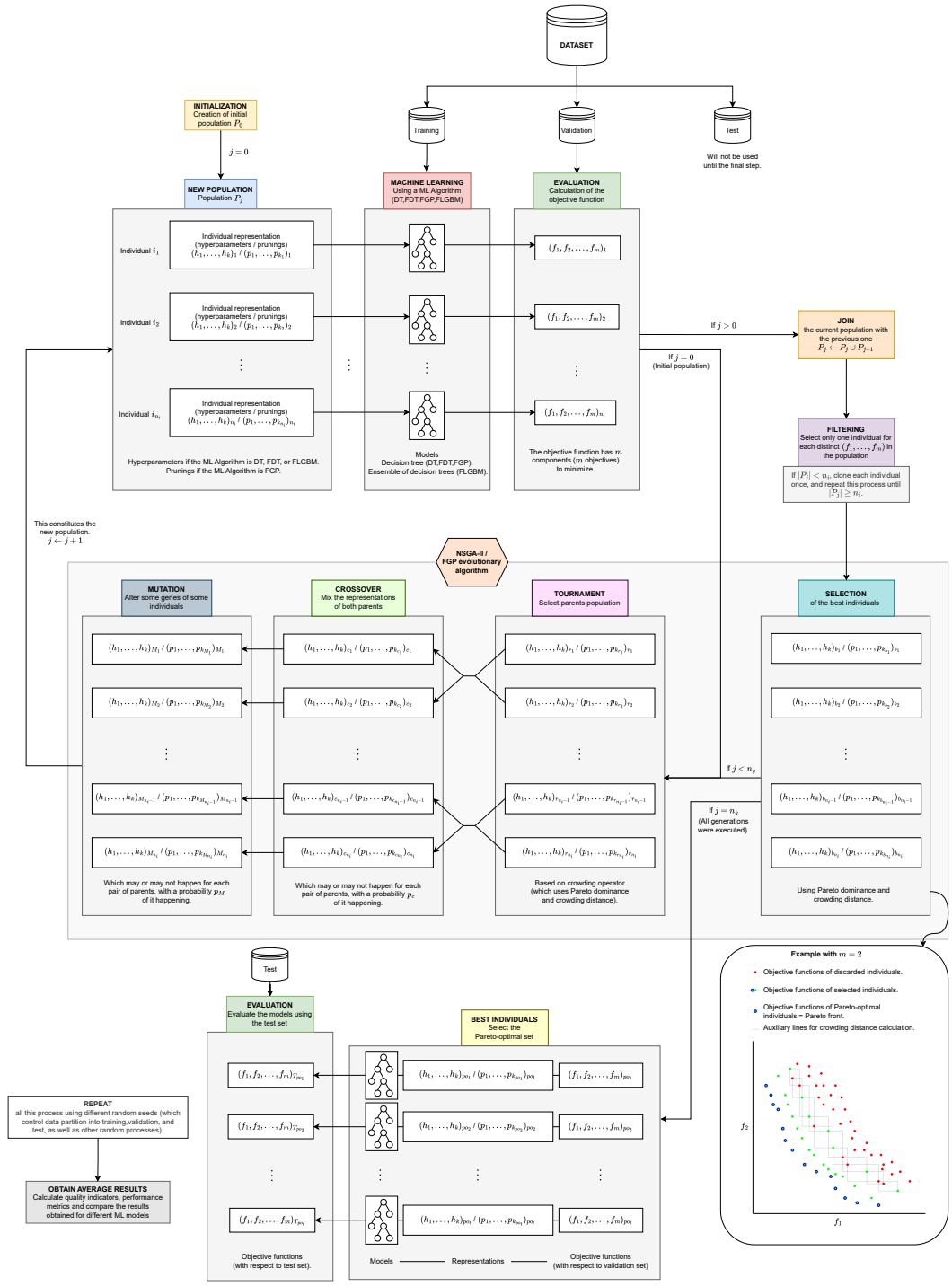


Figure 9.2.: Overall structure of the experimentation conducted for a given dataset. In our case, random seeds will be in $\{100, \ldots, 109\}$, $n_i = 150$, $n_g = 300$, $p_c = 1$, $p_M = 0.3$, $m = 2$, $f_1 =$ Inverted G-mean, and $f_2 = \text{FPR}_{\text{diff}}$

.

10. Implementation details

In this chapter, various aspects of experimentation at the implementation and deployment levels will be discussed. It will include details on the developed software and the hardware used to test the methods.

10.1. Software specifications for experimentation

In this section, the structure of the software developed for experimentation will be explained. Specifically, it will cover the general structure and contents of the folders and files created.

The complete code is available in the following public repository: https://github.com/Daalma7/FairTreesAlgorithms. We will now proceed to explain its main contents.

The main structure of the repository is illustrated in Figure 10.1. Details of the items shown are provided below:

- **FairGeneticPruning**: Folder containing the code related to the algorithm with the same name.
- **HyperparameterOptimization**: Folder containing all the code related to the hyperparameter optimization process for the DT, FairDT, and FairLGBM algorithms using NSGA-II.
- **datasets**: Folder containing datasets used in the experimentation
- **exec_scripts**: Folder containing the execution scripts of the methods for the conducted experimentation.
- **README.md**: File with a simple description of the project.
- **LICENSE**: License of the repository.
- **MasterThesis.pdf**: This document.

The contents of the GeneticPruning folder can be seen in Figure 10.2. Details of the files shown in the figure are provided below:

- **genetic.py**: File that defines the entire genetic optimization process, including selection, crossover, mutation, and replacement criteria.
- **individual.py**: File that defines two crucial classes: one for the structure of the matrix tree and another for an individual consisting of a set of prunings on the matrix tree. All methods for representation and calculation of metrics on individuals are implemented here.

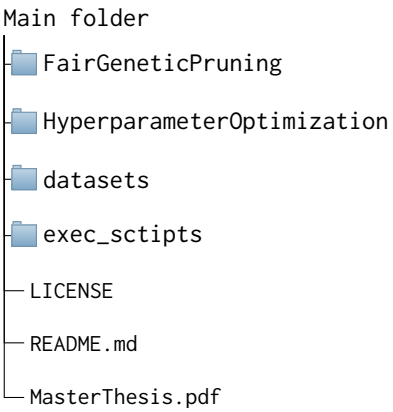


Figure 10.1.: Main contents of the root folder of the project.

- **main.py**: Main file that controls the execution of the method, accepting parameters such as the training set or the number of generations and population size.
- **ml.py**: File containing all logic related to machine learning, including training of individuals, validation, and saving information about the algorithm's performance and results.
- **test.py**: File for testing.

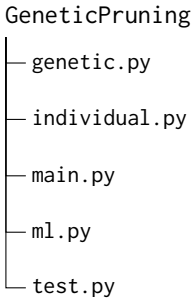


Figure 10.2.: Main contents of the GeneticPruning folder.

The main contents of the HyperparameterOptimization folder are illustrated in Figure 10.3. Details of the files shown are provided below:

- **algorithms**: Folder where different multi-objective optimization algorithms are defined (currently only nsga2 works).
  - **nsga2**: Folder containing the code of the NSGA-II algorithm.
  - **evolution.py**: Main file defining the main evolutionary process of NSGA-II.
  - **utils.py**: File defining functions related to NSGA-II, such as initial population creation, mutation, or crossover.
- **bin**: Folder with different execution scripts.

  - **main.py**: File controlling the main execution of the multi-objective optimization process, specifying the algorithm and parameters.

- **general**: Folder containing general classes and utilities.
  - **individual.py**: File defining classes for individuals used in the optimization process.
  - **ml.py**: File defining all machine learning processes, objective calculation, and model hyperparameter control.
  - **population.py**: File defining classes for populations of individuals in the optimization process.
  - **problem.py**: File containing a class that defines an optimization problem and various general utilities. It centralizes the generation of individuals, calculation of objectives and creation of result files.

- **models**: Folder containing the definition of the two algorithms created to run in this optimization process: FairLGBM and FairDT.
  - **FLGBM (FLGBM.py)**: File defining the FairLGBM algorithm.
  - **FairDT**: Folder containing the definition of the FairDT algorithm. It extends the DecisionTreeClassifier learning algorithm from the scikit-learn library, incorporating relevant fairness modifications. It has a package structure.

The main contents of the exec_scripts folder are illustrated in Figure 10.4. Details of the files shown are provided below::

- **CalculateMeasures**: Folder containing the code related to measure calculations and plots.
  - **calculatemeasures_aux.py**: Functions for measure calculations, plots, tables, and average Pareto-fronts.
  - **main.py**: Main file for executing measure calculations and plots.
  - **qualitymeasures.py**: File containing the definition of quality indicators on Pareto-optimal solutions, as defined in Section 5.3

- **exescript_x**: This script controls the execution of files ending in DT, FDT, FGP and FLGBM, managing the corresponding algorithm executions. The file ending in Measures controls the execution of the measure calculation.

10.2. Implementation details for algorithms

In this section, we will discuss some details about the implementation that are believed to be relevant. Let us now explain some general implementation principles that help define high-quality code, ensuring high performance and efficient resource usage:

- **Parallelism**: The creation of individuals in each population has been implemented using CPU-level parallelism with Python's joblib module, employing its Parallel and delayed methods. Parallel processing is used in both the creation of the initial population and in the crossover and training of offspring individuals. This approach is

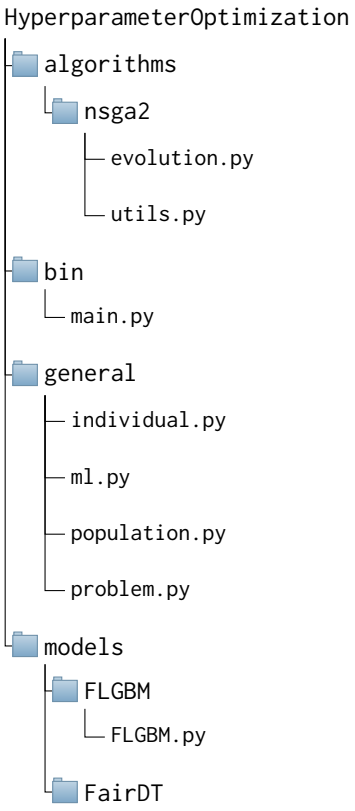


Figure 10.3.: Main contents of the HyperparameterOptimization folder.

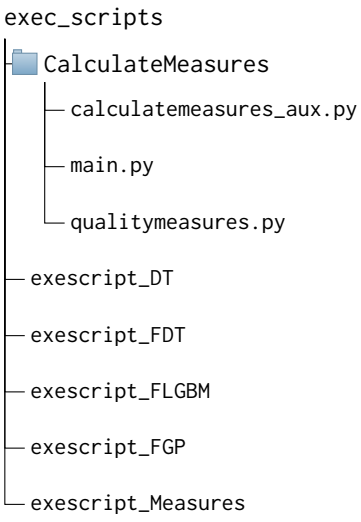


Figure 10.4.: Main contents of the GeneticPruning folder.

highly effective because the parameter specification (whether random in the initial population or selected through crossover and mutations in subsequent populations) and the training of individuals are entirely independent processes. This optimization significantly enhances efficiency by eliminating unnecessary sequential calculations, resulting in noticeable improvements in execution time.

- **No validation repetition**: Suppose the crossover probability is not 1, meaning that the same parent individuals are not returned to the offspring population. In this scenario, the individuals will not be retrained, thus saving computation time.
- **Saving individuals only when necessary**: The individuals generated during the optimization process will be stored in result files. Storing these individuals after each generation can increase computational cost due to potential bottlenecks in writing and reading result files. Therefore, it is appropriate to save individuals only at the end of the hyperparameter optimization process. Files will be saved for all individuals considered, including those in the Pareto-optimal set.
- **Efficient execution scripts**: All algorithms are executed using Bash scripts optimized for speed. These scripts launch all runs for the same dataset simultaneously, then wait for their completion before starting the next batch.

Next, we will discuss details about the implementation of each specific algorithm:

10.2.1. Implementation of the FairDT algorithm

To implement the FairDT algorithm, the DecisionTreeClassifier version from the scikit-learn library (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html) was modified. The code was downloaded from Github (https://github.com/scikit-learn/scikit-learn/tree/f07e0138bfee41cd2c0a5d0251dc3fe03e6e1084/sklearn/tree, accessed: April 2022),and relevant files such as _criterion.pyx, .pxd, _classes.py, and _tree.pyx were adjusted, written in Python and Cython. Successful compilation requires installing the Cython package using a version prior to 3.x.x (version 0.29.37 was used in this case). There is a file named build.sh within the folder containing the developed FairDT code, which contains the necessary code to execute in order to compile the algorithm package and make it usable.

The result is the new DecisionTreeClassifier class, which functions similarly to the scikit-learn version but with several modifications. It includes the fairness criteria gini_fair and entropy_fair, in addition to the gini and entropy criteria. These new criteria allow for the use of the fairness parameter and function; if gini or entropy are selected, these fairness criteria will not be used. The parameter f_lambda, which is the parameter controlling the relative importance given to the fairness function as described in Section 6.1. Lastly, the fair_fun parameter allows us to select the fairness function to apply. By default, it is set to fpr_diff, but it can also be any of the other objectives specified in the objectives section (fpr_diff, ppv_diff, pnr_diff). Additionally, two more were implemented: tpr_diff and tnr_diff.

10.2.2. Implementation of the Fair Genetic Pruning algorithm

The implementation of this algorithm has been entirely done using the Python language. Classes were created to represent the matrix tree, each individual (a set of prunings over the

matrix tree or subtree sharing the root node), and the genetic optimization process itself.

Thanks to the different representations implemented and discussed in Chapter 7, the method's efficiency is enhanced. Classes were created to represent the matrix tree, enabling faster access to critical information. For example, dictionaries were implemented to track the number of instances divided by class and protected attribute within each node, as well as the total number of instances. An array containing the initial leaves' representation of this matrix tree was also implemented. Storing this information scales linearly with the matrix tree's size (in terms of the number of nodes), significantly improving efficiency for methods affected by this data storage by avoiding repeated calculations. In the case of methods using dictionaries, efficiency improves to $\mathcal{O}(1)$, while for those using the array, efficiency becomes $\mathcal{O}(n)$, where n is the length of the array.

The DecisionTreeClassifier class from the scikit-learn library has been used in order to learn the matrix decision tree. The library's tree definition and its methods were used to access necessary information. The implementation of the NSGA-II method is entirely analogous to that done for the hyperparameter optimization methods.

10.2.3. Implementation of the FairLGBM algorithm

For the implementation, the code from the LightGBM library (`https://lightgbm.readthedocs.io/en/stable/`) was used. The functionality was integrated into a class named FairLGBM, having the essential functionalities required for the algorithm's operation. Modifications were made to the loss function, as well as other functions.

To modify the loss function, it is necessary to specify the *objective* parameter in the LightGBM parameter list and assign it a defined function. This function should accept two parameters: an array with predictions and a LightGBM dataset containing the necessary information for training. The function should return the gradient and Hessian of the loss function. This functionality has been implemented in the bce_fair_loss function. Additionally, a bce_fair_eval function has been implemented to calculate the loss function itself, especially useful in processes involving early stopping or validation sets.

In our case, we will indeed use early stopping with the validation set to achieve more accurate models and save training steps. This approach is crucial for hyperparameter optimization processes where the base algorithm is trained multiple times. However, training can also proceed without specifying these sets.

10.3. Hardware specifications for experimentation

In this section, the hardware used for the experimentation will be explained. The technical specifications of the equipment used are as follows:

- **Case**: Corsair Graphite 760T Black Full Tower.
- **Power supply**: 1000W ATX.
- **Motherboard**: ROG Corsair VIII Hero Socket AM4.

- **Processor**: Ryzen 9 5950X 3.4GHz.
- **Cooling**: Corsair Hydro H100X.
- **RAM**: 2 x 16GB DDR4 3200MHz PC4-25600 - CL16 - 1.35V.
- **SSD**: 1TB 2.5″.
- **Graphics Card**: Asus TUF RTX 3080 10GB.

Part V.

Results and Conclusions

In this part, all the results obtained through the experimentation will be showcased and discussed. Conclusions of the project as a whole will be drawn.

11. Results

In this chapter, the results obtained from the experimentation will be presented. The generated graphs and diagrams will be explained, along with their interpretations.

11.1. Analysis of the effect of the parameter λ in the FDT algorithm

The first study that will be conducted is a small study on the effect of the parameter λ on the structure of decision trees for the FDT model. Figures 11.1 , 11.2, 11.3, and 11.4 show different values of structural characteristics of FDT trees as functions of the parameter λ. Specifically, for each dataset and random seed (which controls data partitioning), FDTs have been trained on the training set unrestricted in terms of size but with 101 different values of the parameter λ, uniformly distributed in the range $[0, 1]$ ($\{0, 0.01, \ldots, 0.99, 1\}$). For every given combination of dataset and λ value, the results obtained for all runs (using the same random seeds as for the rest of the experimentation) have been summarized, showing the mean value of the runs as a point, with an error bar indicating a 95% confidence interval that the measure value lies within this interval. In Figure 11.1 the accuracy results on training and test sets can be seen. In Figure 11.2 analogous information is shown for the FPR_{diff} measure. In Figure 11.3 the depth of the trained trees can be observed. Finally, in Figure 11.4, the number of leaves of these trees can be seen.

The effect of the parameter λ that can be seen in these figures is clear. In Figure 11.1, it can be observed that as the value of the parameter λ increases, the accuracy decreases. This reduction is not constant, and it can be seen that for relatively low values of λ (around 0.1), there is some improvement in the test results. Generally, a sharp reduction begins from the value 0.6.

Values shown in Figure 11.2 are quite striking. It can be observed that as the value of λ increases, initially the FPR_{diff} worsens, which is unexpected. However, this phenomenon is not consistent; there are certain values where FPR_{diff} improves (which are precisely what we want to identify). Around the value 0.6 there is a very pronounced jump, with a similarly abrupt decrease around 0.8. In some cases, the value decreases significantly, achieving values lower than all previous ones. However, as seen in Figure 11.2, there is also a significant reduction in accuracy, as expected.

Finally, in Figures 11.3 and 11.4, trends related to the above but also highly influential in our experimentation are observed. As the value of λ increases, the resulting trees become simpler (with less depth and fewer leaves). From the value 0.8 onwards, the trees are extremely simple.

This last fact is undesirable in a multiobjective optimization process because, regardless of the values of other parameters (which, in fact, can only further limit tree growth), if the value of λ is high, the trees trained will be extremely simple. However, as the objective value in

FPR_{diff} is the most optimal (0), these trees will always appear as non-dominated individuals, and a wide combination of hyperparameters (decision space) will result in the same types of non-dominated trees. In initial experimentations, it was observed that the algorithm tended to accumulate these types of trees and return them at the end of the multiobjective optimization process. This effect has also been observed for every machine learning algorithm used, even for DT and FGP which do not use a λ parameter. Our goal is not only to find these extremely small trees but also to achieve a more complete exploration of the objective space rather than the decision space.

Consequently, the number of solutions with the same objective values in each generation of our multiobjective evolutionary algorithms will be limited, as explained in Section 5.4 and shown in Figure 9.2. After combining the populations from both the previous generation and the current one, only one individual for each distinct value in the objective function will be considered. If there are not enough individuals to complete the population size, those considered will be duplicated, and if necessary, duplicated again until the population reaches the required size.

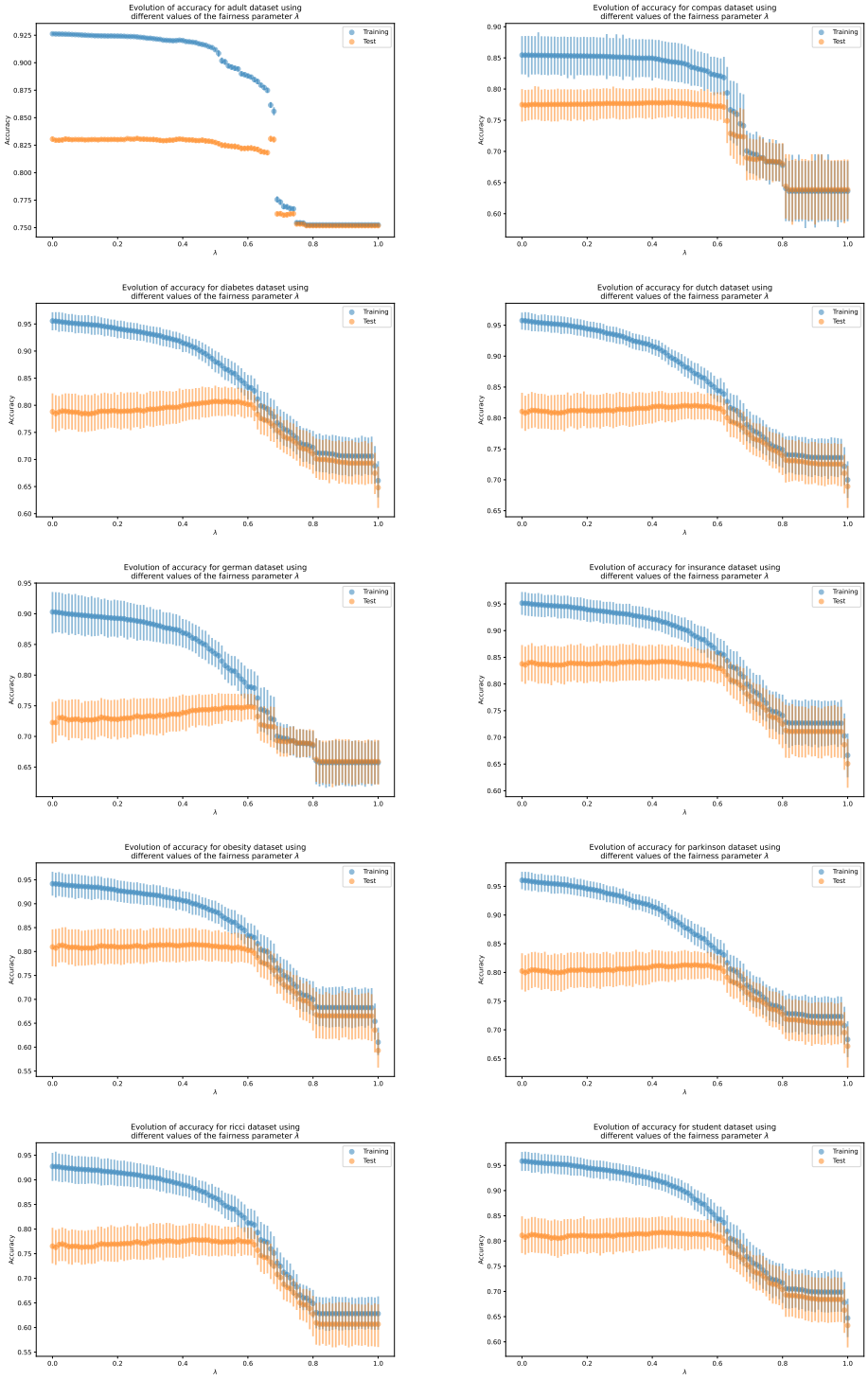


Figure 11.1.: Evolution of accuracy varying the fairness parameter

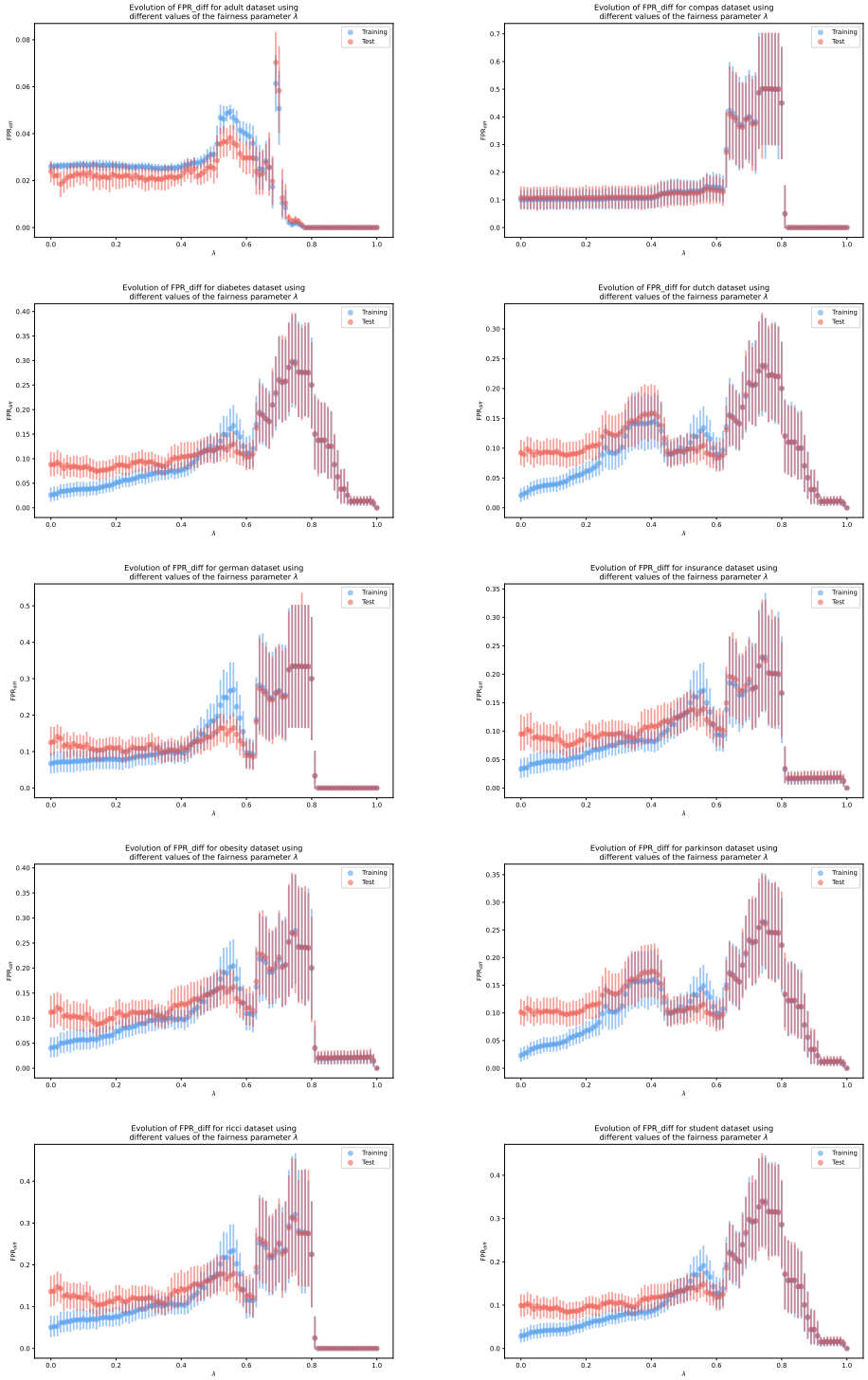


Figure 11.2.: Evolution of FPR_{diff} varying the fairness parameter

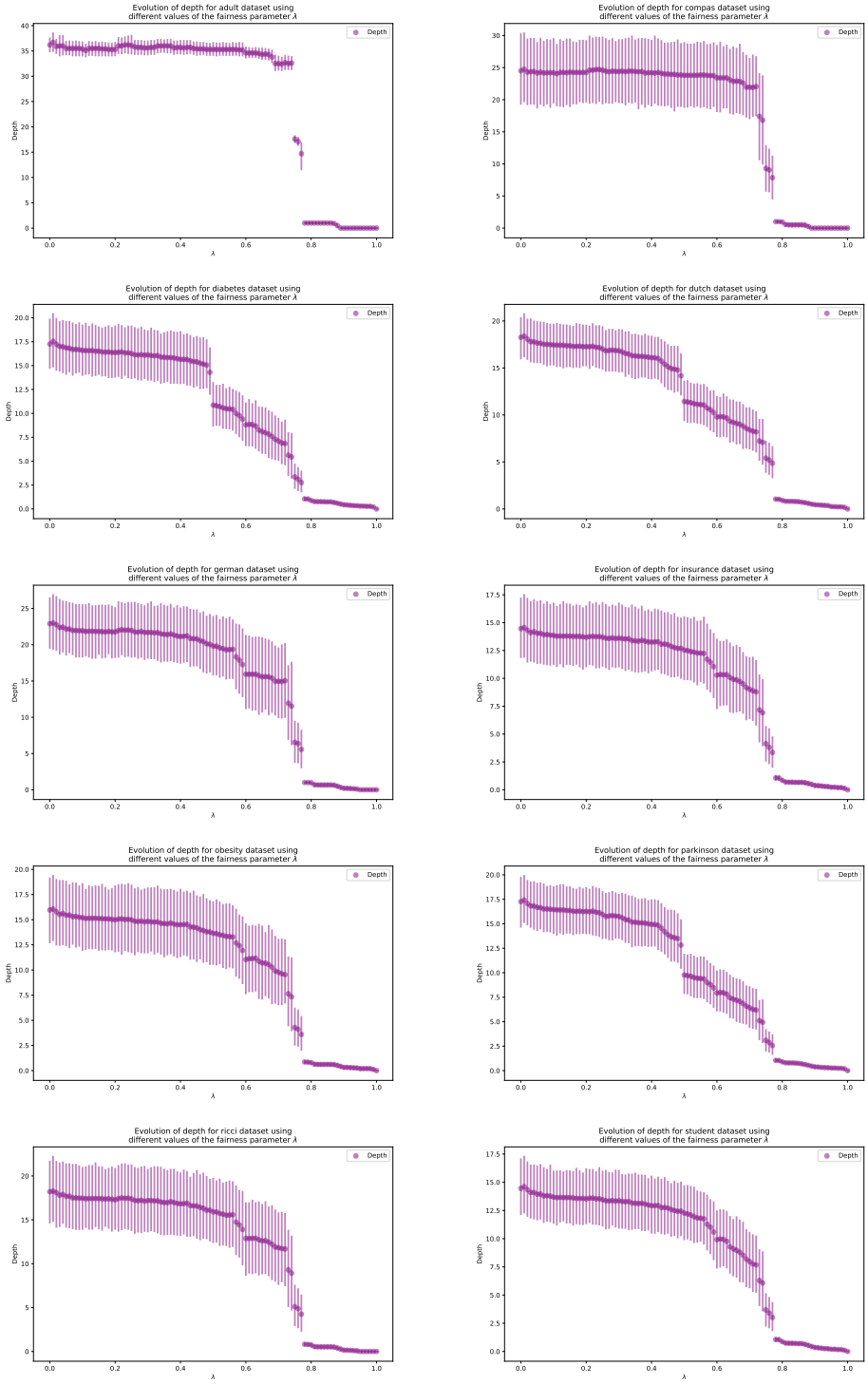


Figure 11.3.: Evolution of depth varying the fairness parameter

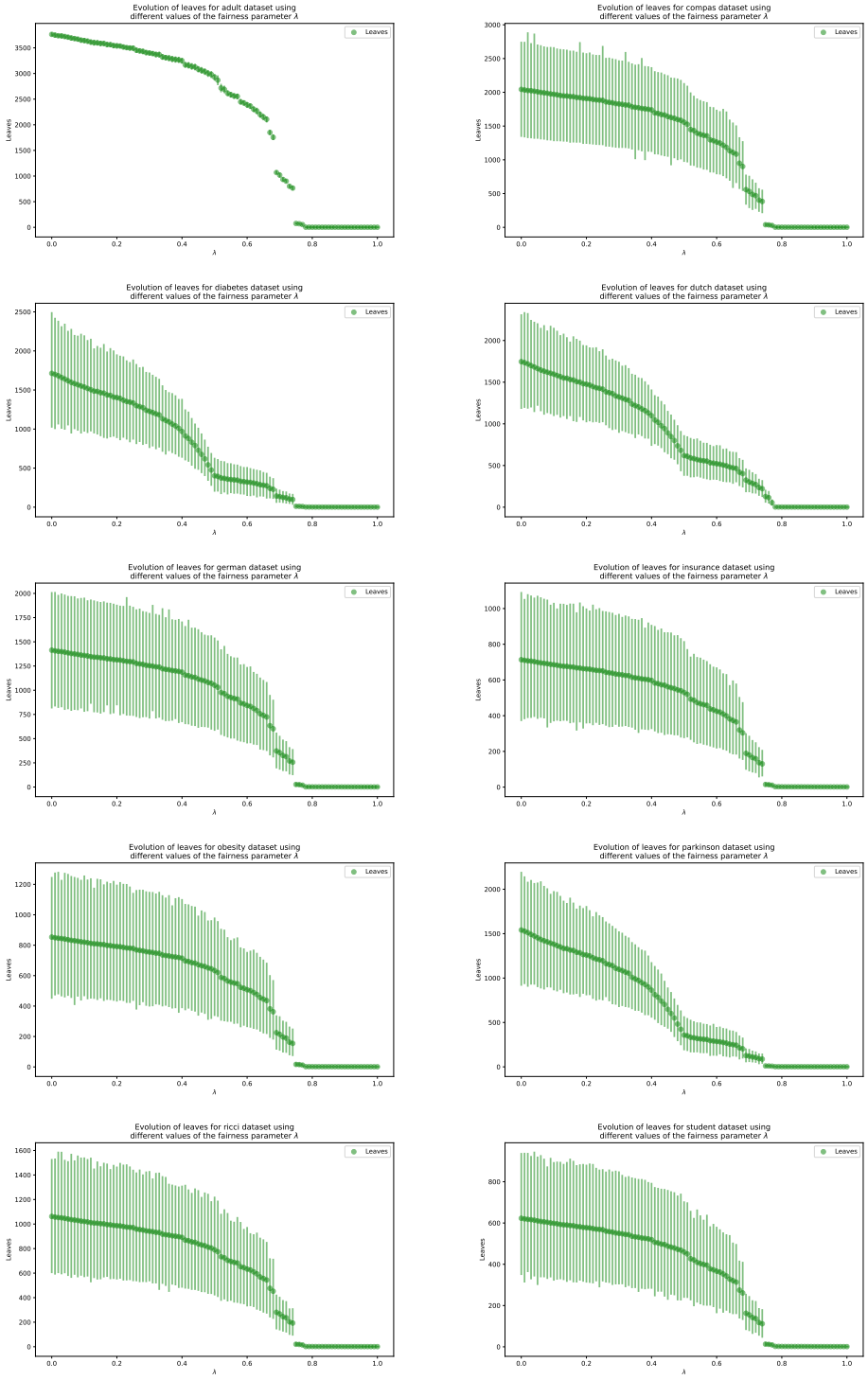


Figure 11.4.: Evolution of the number of leaves varying the fairness parameter

11.2. Comparative graphs and tables of general experimental results

We will continue by presenting general results regarding the runs of the multiobjective optimization process. They will be presented in the format of graphs and tables. From Figure 11.5 to Figure 11.34, various result graphs for each dataset can be found. We will explain the content of each of these graphs:

- **Pareto-optimal sets**: Figures 11.5, 11.8, 11.11, 11.14, 11.17, 11.20, 11.23, 11.26, 11.29 and 11.32 contain information about the Pareto-optimal sets of solutions.
  - **Scatterplots showing average Pareto fronts by algorithm**: The top-left graphs show a scatterplot where the test objective function of each individual found in each Pareto-optimal set of each run and algorithm is plotted using a soft tone. From these, an average Pareto front using these test results has been created for each algorithm. The number of individuals in this average set is the average of the number of Pareto-optimal individuals found in each run. On the other hand, the values in each objective are calculated using percentiles. With two objectives ($f = (f_1, f_2)$), and a population of n individuals, the objective values of these individuals will be $\left\{\left(p_{f_1}\left(\frac{i}{n-1}\right), p_{f_2}\left(1-\frac{i}{n-1}\right)\right), i \in \{0,\ldots,n-1\}\right\}$, where the functions p_{f_1} and p_{f_2} are the percentile functions of each test objective calculated over the population of Pareto-optimal solutions from all runs for that algorithm and dataset. In this way, average values are established using all runs. These average values appear in a stronger tone. Additionally, to indicate the approximate shape of the Pareto front, a Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) has been used, producing a continuous, smooth, and generally good approximation for this specific case. Additionally, in the case of the compas dataset, the result obtained by the real COMPAS model developed by Northpointe (now Equivant) is included for comparison with our algorithms [Valdivia et al., 2020].

    These scatterplots show average Pareto fronts with reasonably good shapes and objective function values using the test datasets. For some datasets, the average Pareto fronts of all algorithms are very similar (such as for compas dataset or dutch dataset), but there are still noticeable differences. For other datasets (obesity dataset, ricci dataset), the differences are more pronounced. Nevertheless, it can be seen that overall all algorithms function correctly, with some better suited to certain types of problems than others, which follows the No Free Lunch Theorem. Overall, it can be seen that the algorithms FDT and FLGBM perform slightly better than the other two, although there are some datasets where DT and FGP perform better (such as dutch dataset).

    In the case of the compas dataset, we can see how all algorithms find much better models than the model used by Northpointe. Specifically, for the same $1-$G-mean, the Northpointe model is approximately twice (100% more) as unfair as our models, and for the same FPR_{diff}, the Northpointe model has a $1-$G-mean 16% worse than our models.
  - **Scatterplot showing general average Pareto front**: The top-right graph is a scatterplot that uses the results from the left graph to select the general average

Pareto front from all values of the average Pareto fronts of all algorithms. Additionally, the remaining values of the average Pareto front of each algorithm that do not belong to the general average Pareto front are shown in a softer tone. The same Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) cubic monotonic interpolation is used. As in the previous scatterplot, in the case of the compas dataset, the result obtained by the real COMPAS model developed by Northpointe is also shown.This general average Pareto front will be useful for comparing how each algorithm performed.

– **Pieplot showing proportion of individuals**: The bottom-left graph corresponds to the number and proportion of individuals by algorithm in the general average Pareto front.

Here we begin to see some clearer trends. The algorithms FDT and FLGBM tend to have the highest representation in the general average Pareto front, indicating that they achieved better solutions compared to the other two algorithms. There are some datasets where this trend does not hold, such as in dutch or in ricci, but generally, this is the case.

– **Violinplot showing the ratio between test and validation results**: The bottom-center graph displays the ratio between test and validation objectives for the Pareto-optimal solutions of each algorithm. These ratios are calculated for every Pareto-optimal individual found in each run. Since each objective function takes values in the range $[0, 1]$, and lower values indicate better performance, if the violinplot shows values much greater than 1, it indicates overfitting for that specific objective and algorithm. Values less than 1 indicate that test results are better than validation results. The graph is presented in a logarithmic scale.

The results vary greatly between datasets. Generally, for larger datasets, there are minimal differences between validation and test results for $1-$G-mean (indicating good approximations). However, we can observe more significant differences in FPR_{diff}, although these are not particularly noticeable except for a few exceptions. In smaller datasets, the differences between validation and test results are larger. This may be because smaller datasets are less likely to be representative of the entire population, both for training and validating models. This makes it more challenging to achieve consistent results between validation and test sets. Nevertheless, the test results have been generally very satisfactory for both objectives in all datasets.

Overall, the differences in FPR_{diff} are larger than those for $1-$G-mean, which can also be attributed to the scale of the results obtained. Many FPR_{diff} results approach the optimal value (0), and a relative difference between 0.0001 and 0.01 is much larger than a relative difference between 0.3001 and 0.31, as may have occurred in the $1-$G-mean error objective.

– **Coverage heatmap**: The bottom-right graph displays a matrix of coverage indicators between the average Pareto fronts generated for each algorithm. Darker colors indicate higher values, while lighter tones indicate lower values (heatmap). A high value at a specific position is better for the algorithm of that row and

worse for the algorithm of that column.

Generally, the conclusions drawn from these results are similar to those derived from analyzing the scatterplots. In this case, we can see numerical values regarding this relative convergence comparison between average Pareto fronts.

- **Structure of individuals across generations**: Figures 11.6, 11.9, 11.12, 11.15, 11.18, 11.21, 11.24, 11.26, 11.30 and 11.33 contain information about different metrics across generations for each algorithm. These metrics provide insights into the structural characteristics of the models generated over generations. The results shown for each metric are statistical measures calculated over all individuals of the population in each generation. For each metric, the maximum, mean, and minimum values are shown using lines, and standard deviation values are displayed in a shaded area. The structural characteristics measured for each algorithm are as follows:
  - **DT and FairDT**: Number of leaves, depth, depth weighted by the number of instances that fall into each leave, and unbalance (calculated as depth of the shallower leaf divided by the depth of the deepest leaf).
  - **Fair Genetic Pruning**: Same values as above, as well as the amount of prunings performed.
  - **FairLGBM**: Number of decisions in the final ensemble model (n_estimators), as well as the standard deviation of the feature importances (indicating the importance of each feature in creating the model, feature_importance_std).

  Using these graphs, we can verify whether the general structure of the models learned by the optimization process converges or not, and if it so, how quickly it does so, along with the nature of the structure of the models.

  In general, the algorithms have shown fairly rapid structural convergence in the models found. As seen in the evolution graphs, individual structures typically stabilize before 50 generations. The algorithm that tends to stabilize the latest is FGP, possibly due to exploring configurations with a higher number of prunings at deeper levels. These results could indicate that algorithms find effective models without needing 300 generations. However, further studies and analysis regarding the convergence of each objective function and quality indicators across the solution sets of each generation are needed to confirm this assumption.

- **Barplots of quality indicators and CPU processing times**: Figures 11.7, 11.10, 11.13, 11.16, 11.19, 11.22, 11.25, 11.28, 11.31 and 11.34 show barplots containing information on quality indicators for the average Pareto fronts calculated for each algorithm. Additionally, it is indicated whether a low or high value is better for each indicator. Barplots containing the average total processing time in seconds considering all runs for each algorithm and dataset are also shown. This processing time is calculated using the *time.proces_time()* function, which returns the CPU processing time spent exclusively for that process, excluding other external influences. These graphs are highly informative for comparing the performance of the algorithms across these datasets.

Additionally, Figure 11.35 shows barplots containing rankings performed on each of the quality indicators, as well as on the processing times. In particular, if an algorithm has achieved

the best score among the 4 for a specific quality indicator and dataset, it will receive 3 points. If it achieved second place, it will receive 2 points; if third, 1 point, and if it ranked last, it will receive no points. The ranking has been conducted by summing up all the points obtained for each dataset. The same ranking rules will be applied to the processing time.

Based on all the barplots, we can extract some interesting results. Firstly, the algorithms that generally performed better were FDT and FLGBM. FLGBM achieved the best results in hypervolume, error ratio, generational distance, inverted generational distance, and overall Pareto front spread, which are 5 out of the 7 unary quality indicators used, and it was not the slowest algorithm either. FDT either matches or surpasses DT in almost all rankings shown in Figure 11.35, except for generational distance (by a small margin) and CPU processing time (where it performed worst, but looking at each algorithm's barplots reveals the time difference is minimal compared to DT). This extremely small time difference to achieve equal or better results across 6 out of 7 unary quality indicators makes FDT an algorithm worth considering over the base algorithm in this multiobjective optimization scenario. Additionally, FDT outperforms FLGBM in the overall rankings in the quality indicators where FLGBM does not win.

The case of FGP is special. Despite not achieving wins in almost any quality indicator, it does excel in CPU processing time, although there are certain nuances to consider. Firstly, FGP is not an algorithm that has performed poorly; on the contrary, it obtains very good results in some datasets like dutch, and results very similar to others in almost all other datasets. It is true that in some datasets it obtained worse results, such as in Parkinson or German, but these are exceptions. However, when compared in terms of quality indicators, it unquestionably behaves worse. Although it is capable of obtaining good solutions, it also finds some not-so-good solutions that affect the calculation of the average Pareto front. Adaptations should be made to make the algorithm more stable in finding good solutions. On the other hand, despite generally achieving the lowest CPU processing times, caution is necessary, as it performed quite poorly in one of the largest datasets (diabetes) in terms of CPU processing time. While it achieved fairly good results for other large datasets (dutch and adult), it may encounter some scaling issues not related to dataset size, but rather to the size of the matrix tree. To confirm this assumption, more experimentation is needed.

All the images have been inserted in PDF format (not only in this section, but throughout the entire document), so you can zoom in to see them in detail.

Results for adult dataset

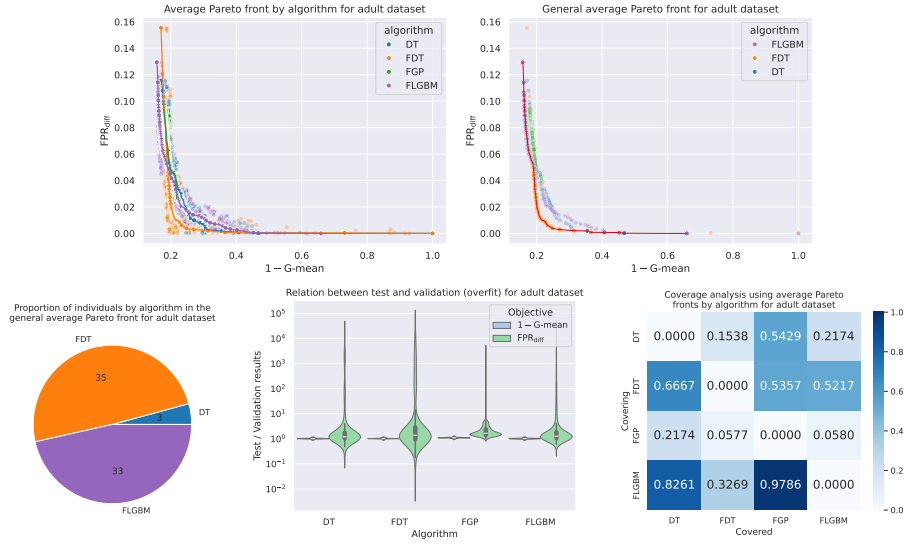


Figure 11.5.: General metrics for Pareto-optimal solution sets for adult dataset.

Figure 11.6.: Evolution of structural metrics of models for adult dataset.

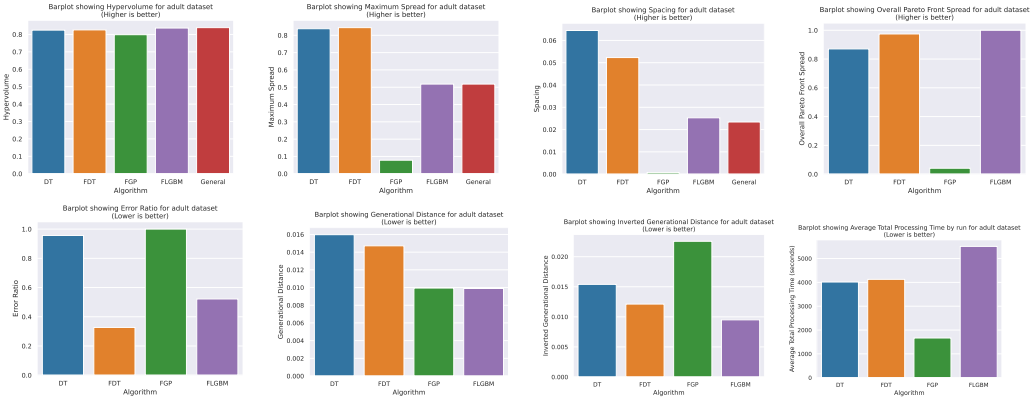


Figure 11.7.: Quality indicators for avg. Pareto fronts and processing times for adult dataset.

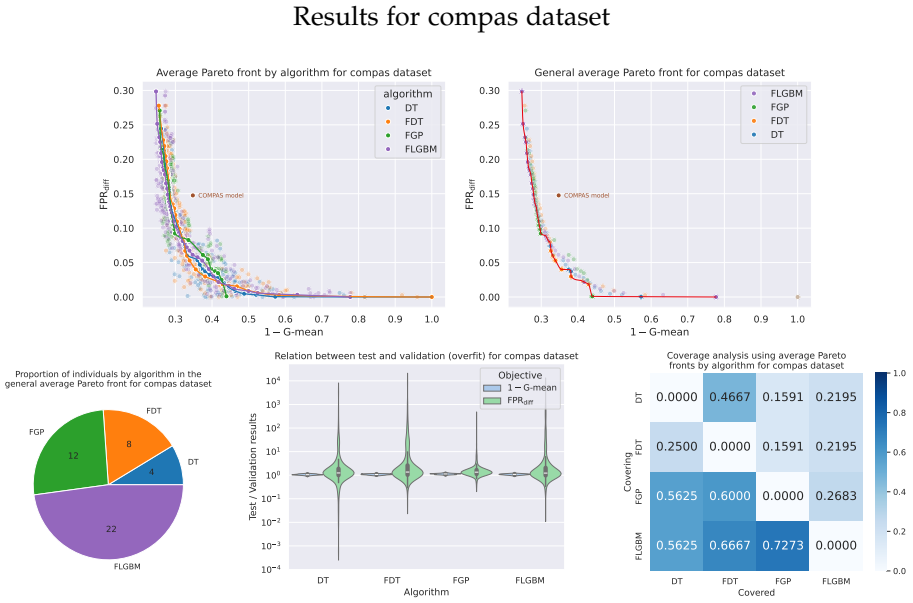


Figure 11.8.: General metrics for Pareto-optimal solution sets for compas dataset.

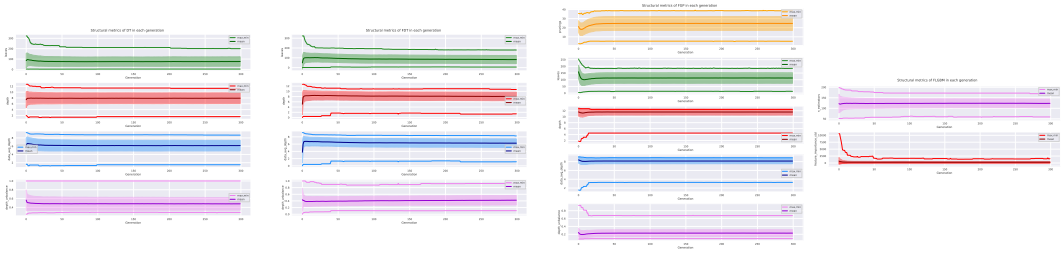


Figure 11.9.: Evolution of structural metrics of models for compas dataset.

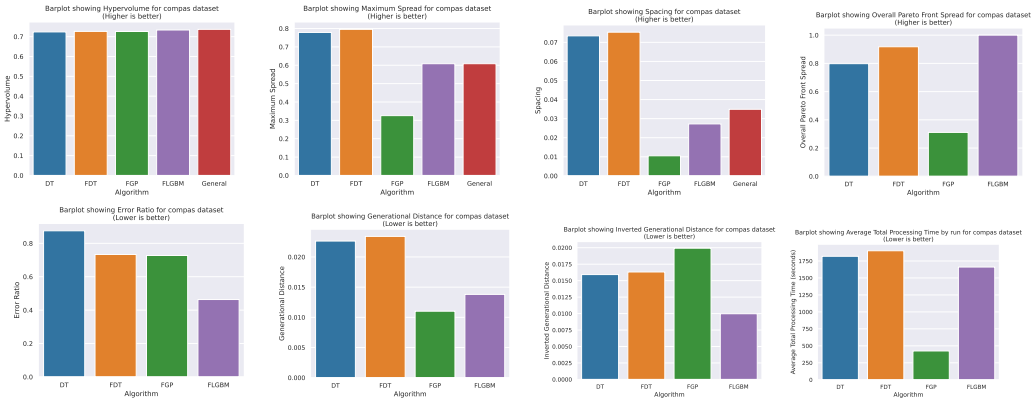


Figure 11.10.: Quality indicators for avg. Pareto fronts and processing times for compas dataset.

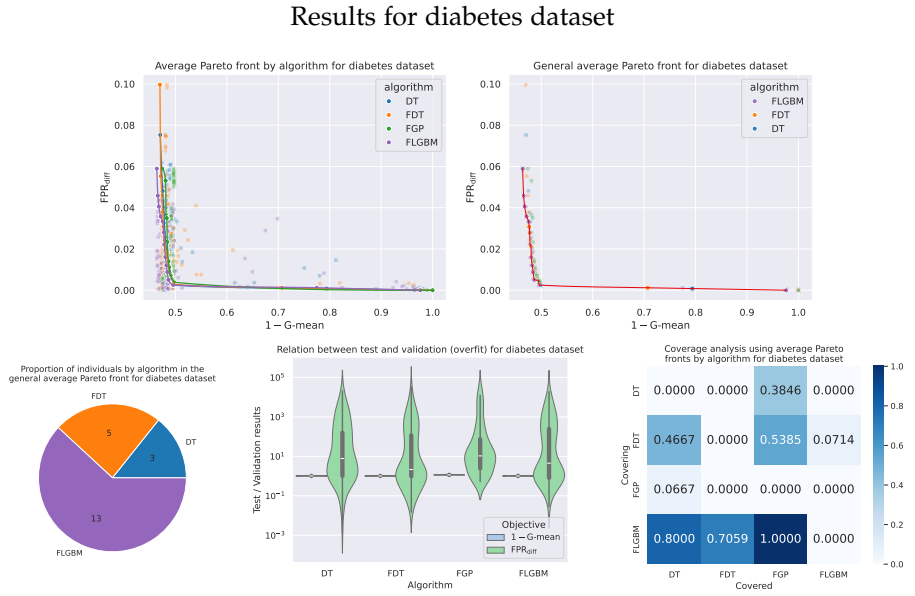


Figure 11.11.: General metrics for Pareto-optimal solution sets for diabetes dataset.

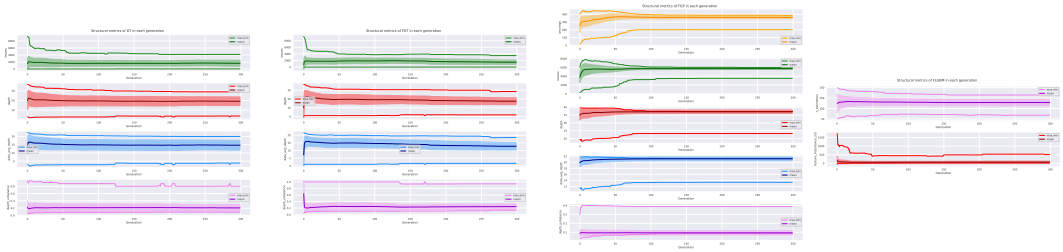


Figure 11.12.: Evolution of structural metrics of models for diabetes dataset.

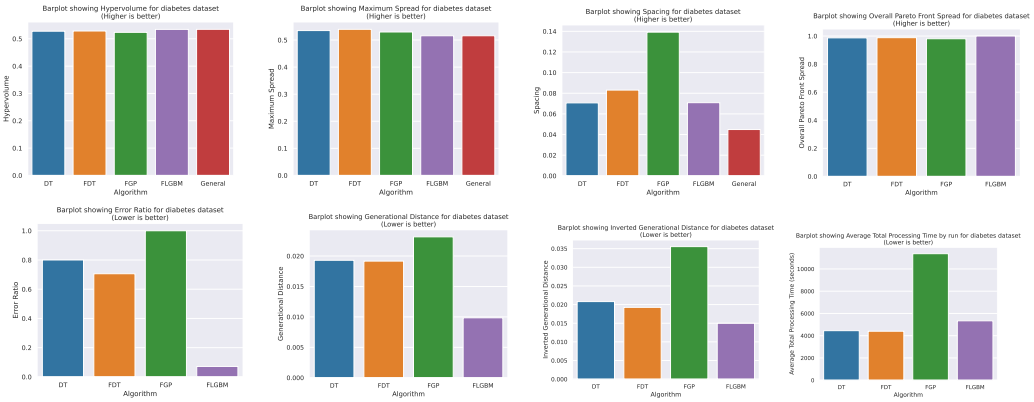


Figure 11.13.: Quality indicators for avg. Pareto fronts and processing times for diabetes dataset.

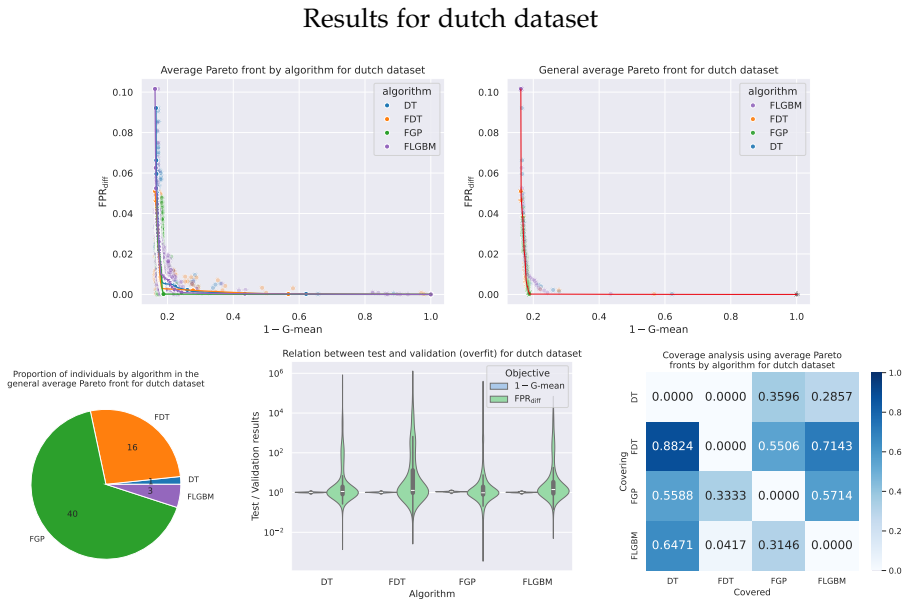


Figure 11.14.: General metrics for Pareto-optimal solution sets for dutch dataset.

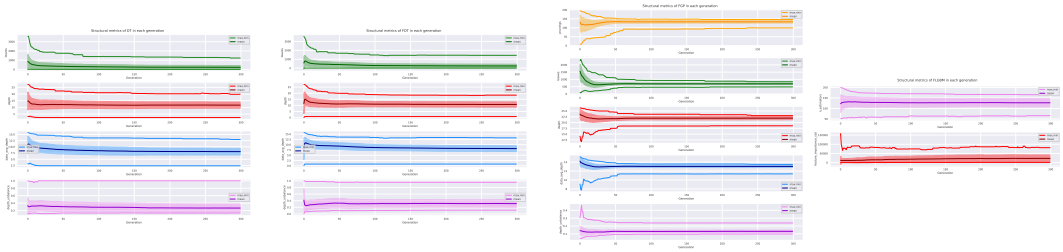


Figure 11.15.: Evolution of structural metrics of models for dutch dataset.

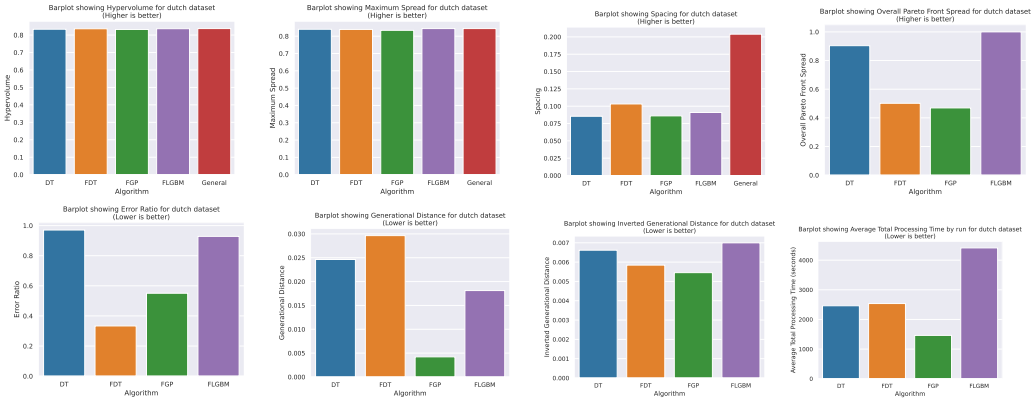


Figure 11.16.: Quality indicators for avg. Pareto fronts and processing times for dutch dataset.

Results for german dataset

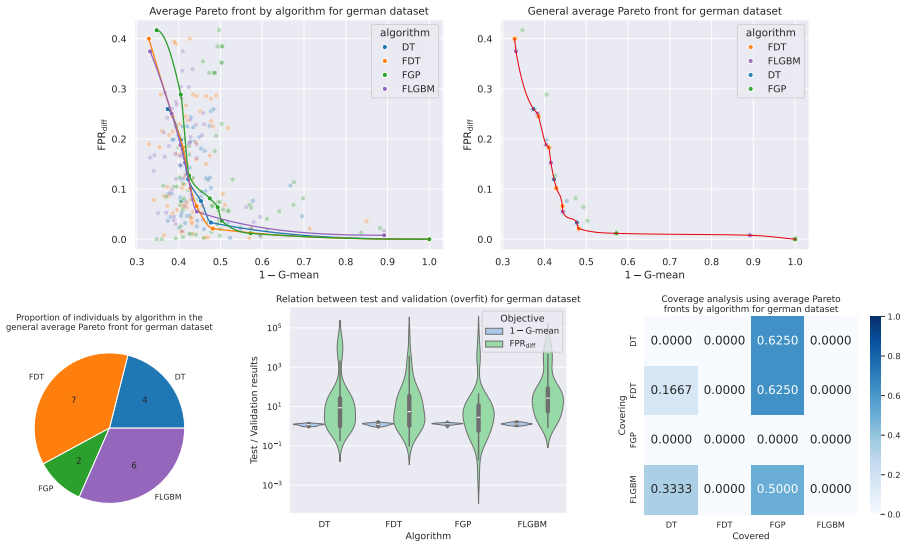


Figure 11.17.: General metrics for Pareto-optimal solution sets for german dataset.

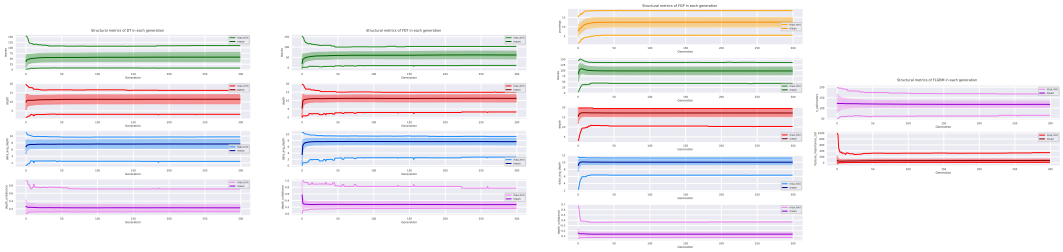


Figure 11.18.: Evolution of structural metrics of models for german dataset.

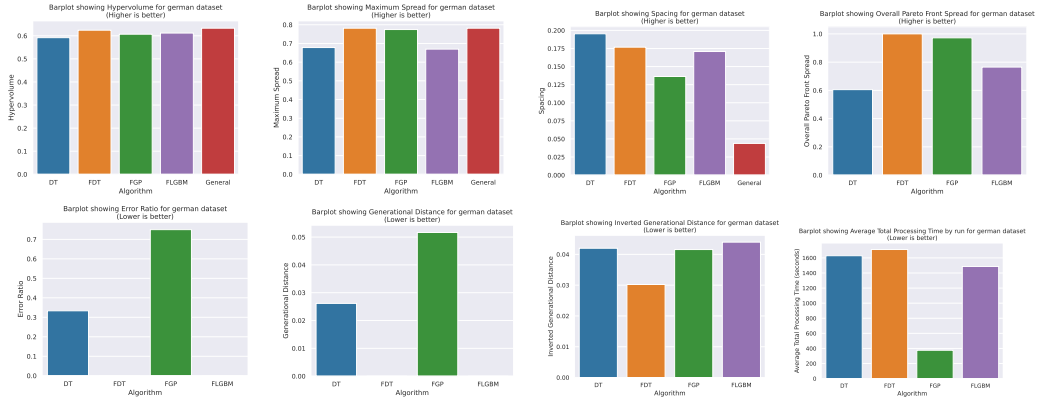


Figure 11.19.: Quality indicators for avg. Pareto fronts and processing times for german dataset.

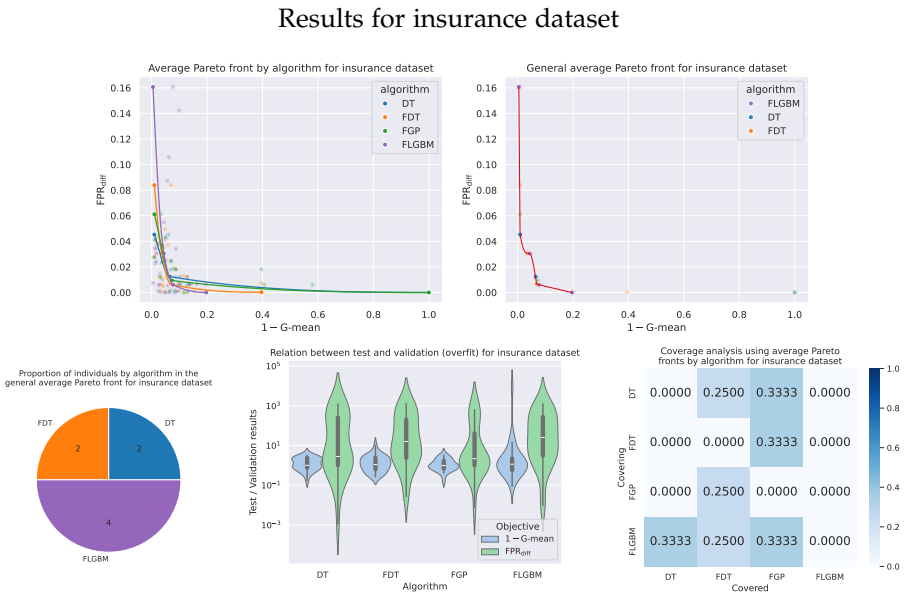


Figure 11.20.: General metrics for Pareto-optimal solution sets for insurance dataset.

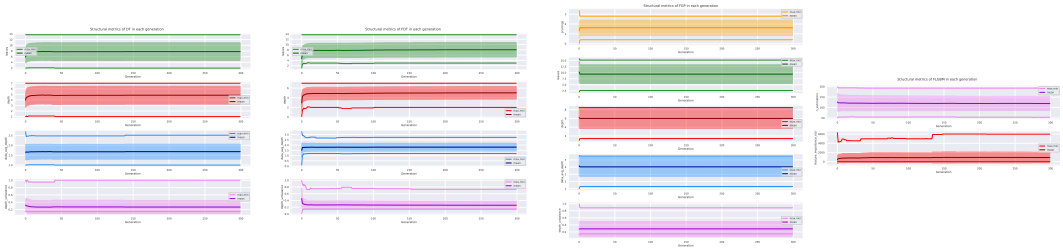


Figure 11.21.: Evolution of structural metrics of models for insurance dataset.

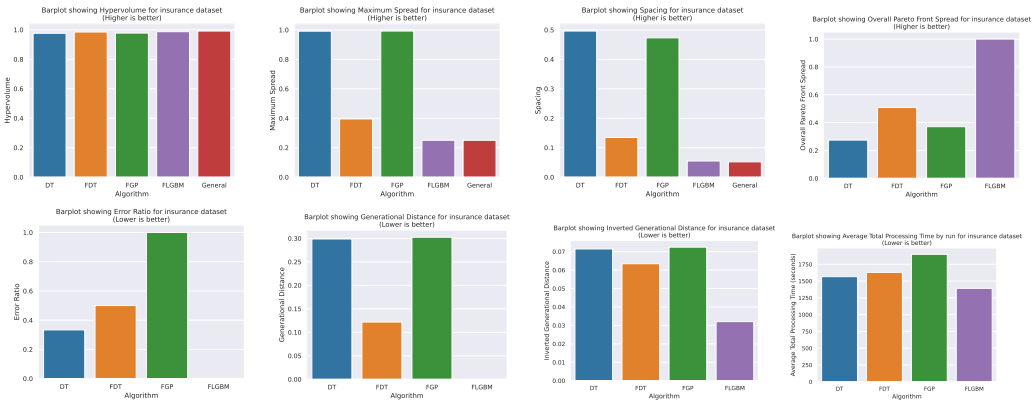


Figure 11.22.: Quality indicators for avg. Pareto fronts and processing times for insurance dataset.

Results for obesity dataset

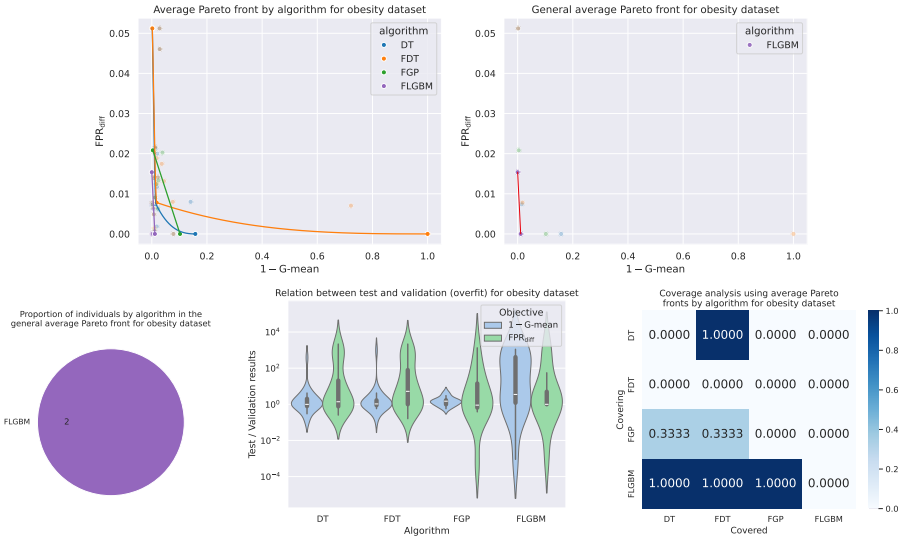


Figure 11.23.: General metrics for Pareto-optimal solution sets for obesity dataset.

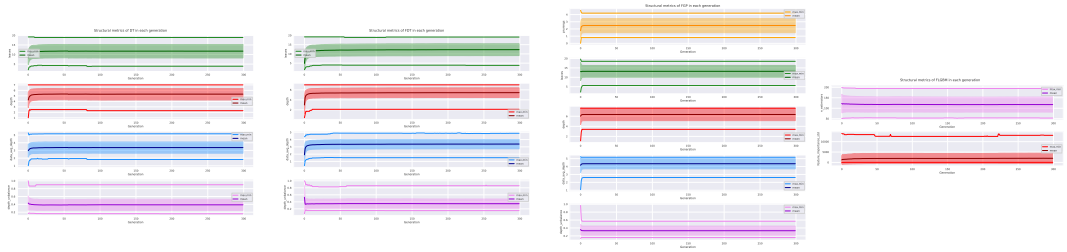


Figure 11.24.: Evolution of structural metrics of models for obesity dataset.

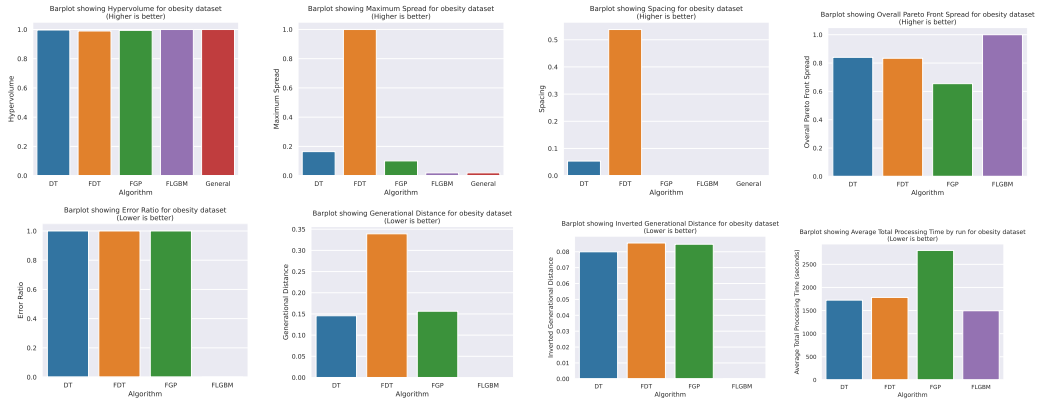


Figure 11.25.: Quality indicators for avg. Pareto fronts and processing times for obesity dataset.

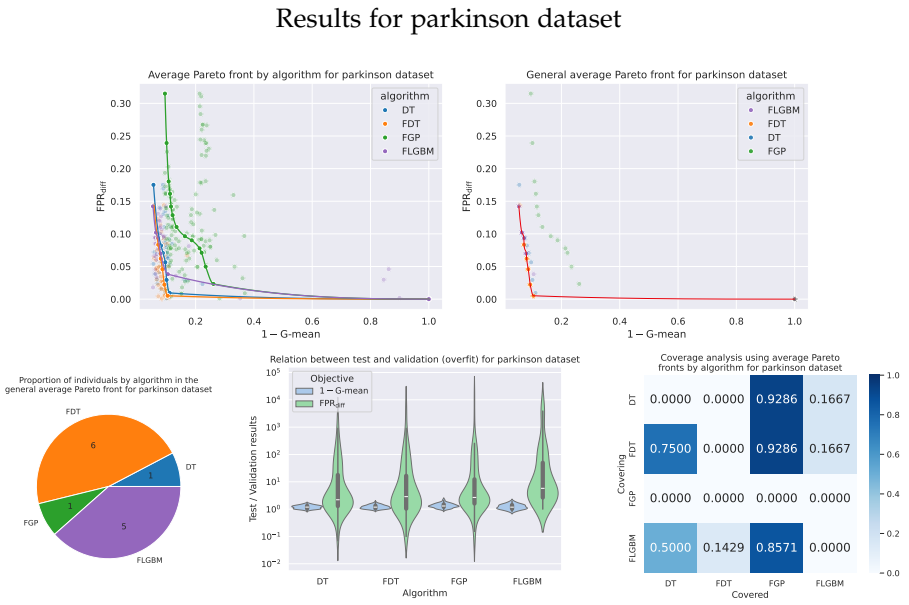


Figure 11.26.: General metrics for Pareto-optimal solution sets for parkinson dataset.

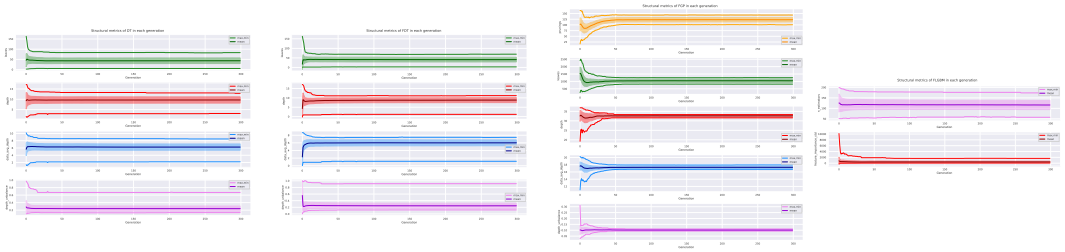


Figure 11.27.: Evolution of structural metrics of models for parkinson dataset.

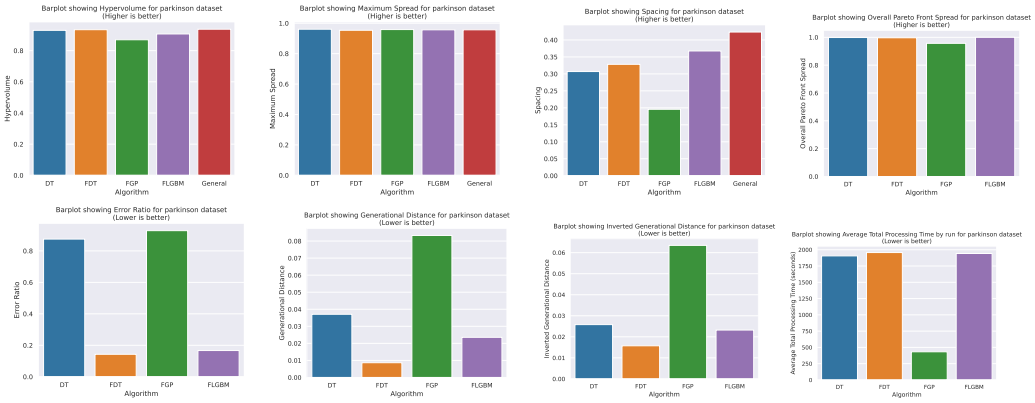


Figure 11.28.: Quality indicators for avg. Pareto fronts and processing times for parkinson dataset.

Results for ricci dataset

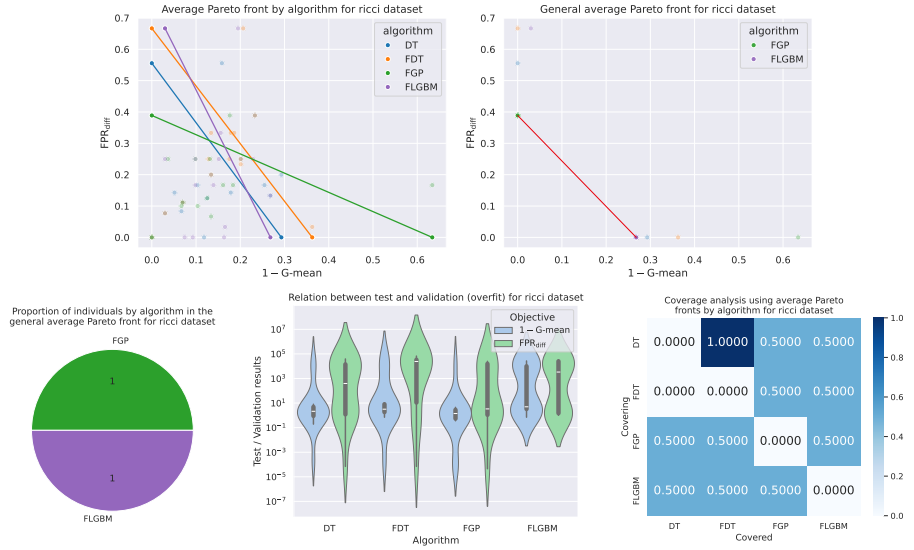


Figure 11.29.: General metrics for Pareto-optimal solution sets for ricci dataset.

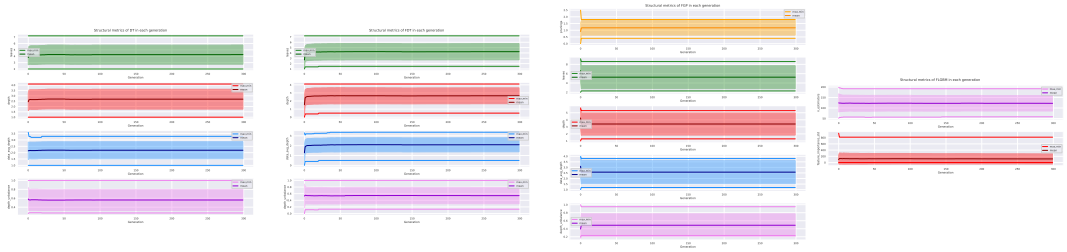


Figure 11.30.: Evolution of structural metrics of models for ricci dataset.

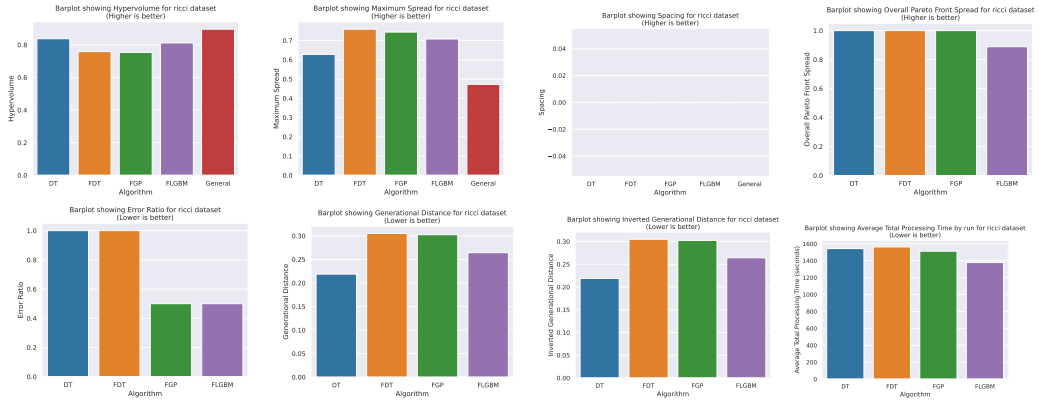


Figure 11.31.: Quality indicators for avg. Pareto fronts and processing times for ricci dataset.

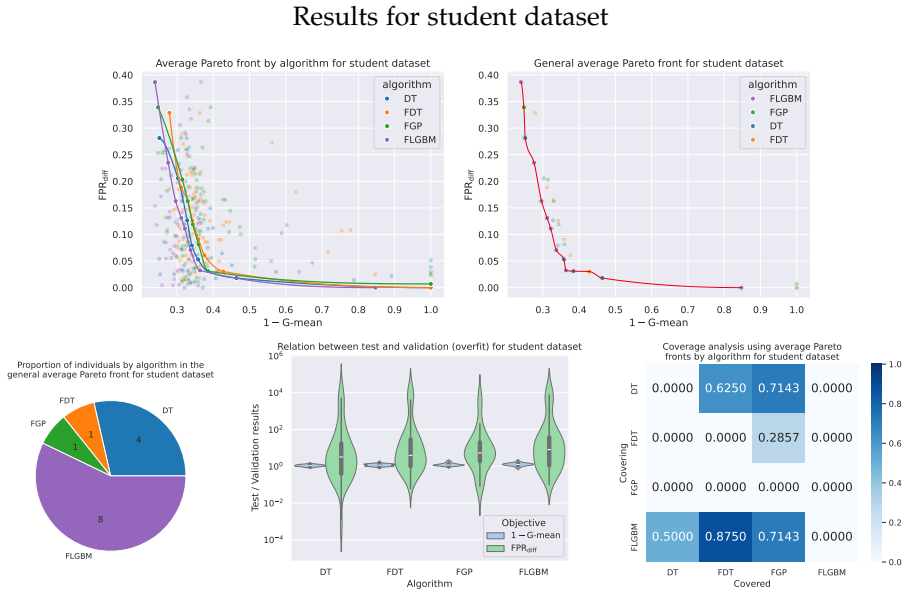


Figure 11.32.: General metrics for Pareto-optimal solution sets for student dataset.

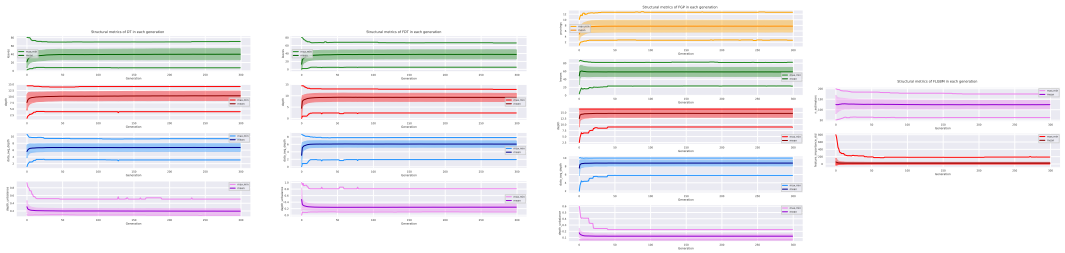


Figure 11.33.: Evolution of structural metrics of models for student dataset.

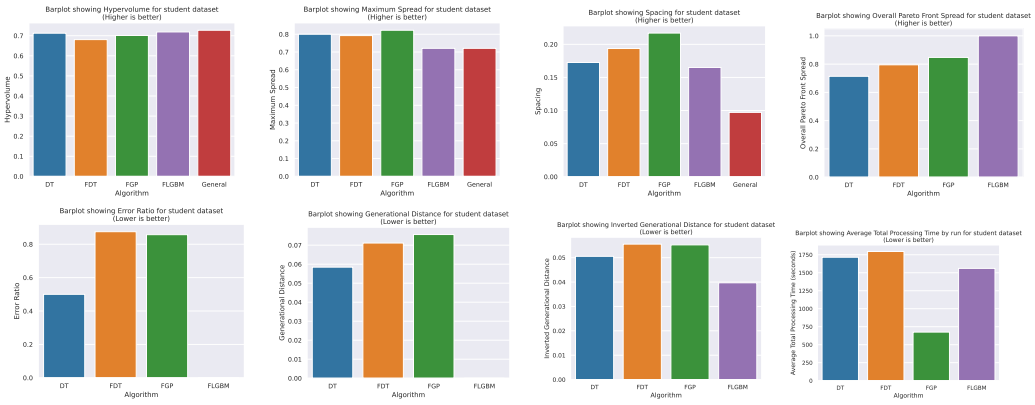


Figure 11.34.: Quality indicators for avg. Pareto fronts and processing times for student dataset.

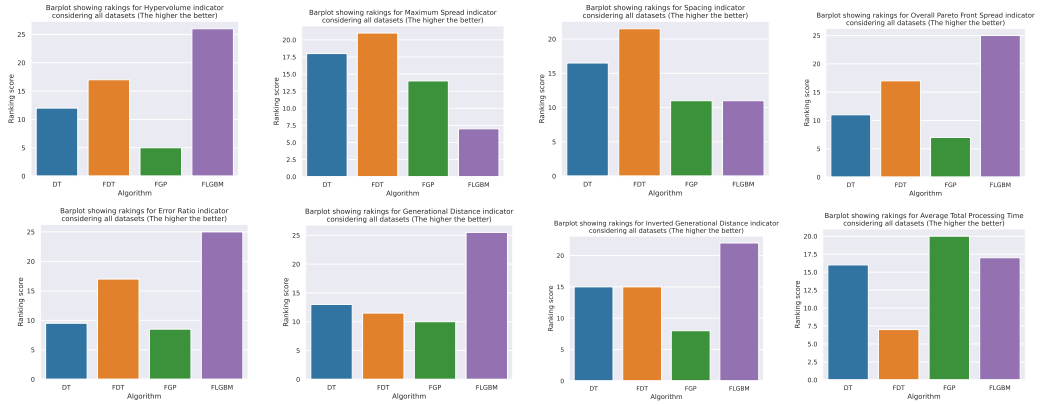


Figure 11.35.: Rankings of quality indicators and CPU processing times among all datasets for the studied algorithms.

Additionally, a table with the results of the quality indicators for the average Pareto fronts is included. This way, the results can be seen numerically, allowing for a better comparison between them. The measures for the general average Pareto front are also included. These results can be observed in Table 11.1.These results complement those seen in the previous quality indicator barplots, enabling more precise comparisons.

Table 11.2 shows the average CPU processing times and their standard deviations (in seconds) of each generation for each dataset and algorithm. Therefore, this information does not represent the average CPU processing time and standard deviation of a single execution, but rather the average CPU processing time and standard deviation for each generation (in our case, the number of generations executed was $n_g = 300$, and each generation has a total of $n_i = 150$ individuals). These results were calculated using the execution times of each generation for each run and algorithm. The results obtained are very similar to those observed in the previous barplots. FGP is consistently the fastest algorithm with low standard deviations, but it encounters issues with the diabetes dataset (where it also exhibits a high standard deviation). FDT is slightly slower compared to DT, while FLGBM also achieved good execution times. FLGBM was slightly slower than DT and FDT for large datasets but slightly faster for small datasets.

Another pertinent analysis performed was a comparative analysis using statistical tests. The Wilcoxon signed-rank test for paired samples was used to compare the results obtained. This study evaluated the algorithms in pairs to determine if there were significant differences between their outcomes. Specifically, the null hypothesis H_0 is that the differences of the paired samples are symmetric around $\mu = 0$, while the alternative hypothesis H_1 is that differences are symmetric around $\mu \neq 0$.

The results of this statistical test will be shown in tables. If the result is statistically significant (p-value < 0.05, in the upper diagonal), it will show which algorithm performed better and which performed worse (lower diagonal). If the algorithm in the row has better results, it will be indicated with the symbol $\oplus$, while if it performed worse, it will be indicated with the symbol $\ominus$. Similarly, even if the results are not significant, if the algorithm in the row has

Table 11.1.: Average Pareto fronts and general average Pareto fronts quality indicators for each dataset.

Data	Quality Indicator	DT	FDT	FGP	FLGBM	G.a.P.front
adult	Hypervolume	0.82476	0.82600	0.79842	0.83667	0.83895
	Spacing	0.06447	0.05231	0.00071	0.02528	0.02340
	Maximum Spread	0.83789	0.84417	0.07826	0.51810	0.51810
	Overall PF Spread	0.87022	0.97417	0.04092	1.00000	1.00000
	Error Ratio	0.95652	0.32692	1.00000	0.52174	0.00000
	GD	0.01597	0.01471	0.00994	0.00989	0.00000
	Inverted GD	0.01540	0.01212	0.02257	0.00950	0.00000
compas	Hypervolume	0.72382	0.72652	0.72596	0.73349	0.73680
	Spacing	0.07340	0.07533	0.01052	0.02723	0.03490
	Maximum Spread	0.77858	0.79543	0.32566	0.60837	0.60837
	Overall PF Spread	0.79787	0.91770	0.31096	1.00000	1.00000
	Error Ratio	0.87500	0.73333	0.72727	0.46341	0.00000
	GD	0.02263	0.02341	0.01102	0.01380	0.00000
	Inverted GD	0.01592	0.01632	0.01993	0.00997	0.00000
diabetes	Hypervolume	0.52781	0.52851	0.52365	0.53451	0.53470
	Spacing	0.07067	0.08307	0.13920	0.07086	0.04485
	Maximum Spread	0.53481	0.53945	0.52957	0.51564	0.51564
	Overall PF Spread	0.98685	0.98819	0.98058	1.00000	1.00000
	Error Ratio	0.80000	0.70588	1.00000	0.07143	0.00000
	GD	0.01930	0.01916	0.02315	0.00985	0.00000
	Inverted GD	0.02080	0.01926	0.03555	0.01498	0.00000
dutch	Hypervolume	0.83339	0.83596	0.83203	0.83635	0.83708
	Spacing	0.08547	0.10322	0.08602	0.09091	0.20391
	Maximum Spread	0.83998	0.83896	0.83388	0.84390	0.84390
	Overall PF Spread	0.90370	0.50168	0.46927	1.00000	1.00000
	Error Ratio	0.97059	0.33333	0.55056	0.92857	0.00000
	GD	0.02463	0.02964	0.00421	0.01811	0.00000
	Inverted GD	0.00662	0.00584	0.00545	0.00700	0.00000
german	Hypervolume	0.59186	0.62429	0.60693	0.61184	0.63295
	Spacing	0.19532	0.17682	0.13629	0.17081	0.04379
	Maximum Spread	0.67805	0.78164	0.77444	0.66965	0.78164
	Overall PF Spread	0.60522	1.00000	0.97198	0.76493	1.00000
	Error Ratio	0.33333	0.00000	0.75000	0.00000	0.00000
	GD	0.02616	0.00000	0.05165	0.00000	0.00000
	Inverted GD	0.04198	0.03021	0.04156	0.04396	0.00000
insurance	Hypervolume	0.97625	0.98491	0.97777	0.98720	0.99150
	Spacing	0.49596	0.13471	0.47287	0.05483	0.05194
	Maximum Spread	0.99140	0.39526	0.99225	0.25041	0.25041
	Overall PF Spread	0.27451	0.50841	0.37082	1.00000	1.00000
	Error Ratio	0.33333	0.50000	1.00000	0.00000	0.00000
	GD	0.29873	0.12190	0.30245	0.00000	0.00000
	Inverted GD	0.07145	0.06344	0.07242	0.03205	0.00000

obesity	Hypervolume	0.99649	0.98968	0.99412	0.99983	0.99983
	Spacing	0.05356	0.53762	0.00000	0.00000	0.00000
	Maximum Spread	0.16416	0.99947	0.10063	0.01896	0.01896
	Overall PF Spread	0.84039	0.83393	0.65450	1.00000	1.00000
	Error Ratio	1.00000	1.00000	1.00000	0.00000	0.00000
	GD	0.14567	0.33911	0.15641	0.00000	0.00000
	Inverted GD	0.07996	0.08550	0.08473	0.00000	0.00000
parkinson	Hypervolume	0.93043	0.93447	0.86985	0.90710	0.93757
	Spacing	0.30694	0.32824	0.19574	0.36758	0.42332
	Maximum Spread	0.96068	0.95377	0.95839	0.95668	0.95668
	Overall PF Spread	0.99844	0.99661	0.95678	1.00000	1.00000
	Error Ratio	0.87500	0.14286	0.92857	0.16667	0.00000
	GD	0.03705	0.00864	0.08327	0.02354	0.00000
	Inverted GD	0.02580	0.01573	0.06346	0.02321	0.00000
ricci	Hypervolume	0.83728	0.75825	0.75343	0.81133	0.89575
	Spacing	0.00000	0.00000	0.00000	0.00000	0.00000
	Maximum Spread	0.62804	0.75891	0.74380	0.70795	0.47233
	Overall PF Spread	1.00000	1.00000	1.00000	0.88862	1.00000
	Error Ratio	1.00000	1.00000	0.50000	0.50000	0.00000
	GD	0.21879	0.30509	0.30247	0.26428	0.00000
	Inverted GD	0.21879	0.30509	0.30247	0.26428	0.00000
student	Hypervolume	0.71258	0.68068	0.70129	0.71880	0.72726
	Spacing	0.17259	0.19366	0.21698	0.16507	0.09721
	Maximum Spread	0.79991	0.79241	0.82243	0.72069	0.72069
	Overall PF Spread	0.71382	0.79497	0.84737	1.00000	1.00000
	Error Ratio	0.50000	0.87500	0.85714	0.00000	0.00000
	GD	0.05843	0.07112	0.07559	0.00000	0.00000
	Inverted GD	0.05054	0.05555	0.05523	0.03969	0.00000

better results in 60% of the occasions, it will be indicated with the symbol $+$. If it is worse in 60% of the occasions, it will be indicated with the symbol $-$. If none of the above applies, it will be indicated with the symbol $=$, indicating that they have performed comparably well an equal number of times.

These results can be seen in Tables 11.3, 11.4 and 11.5. In Table 11.3, only the average Pareto fronts obtained by all algorithms using all runs (data partitions) for each given dataset will be used. Consequently, we will have 10 paired measurements (1 for each dataset) for each quality indicator and pair of algorithms to compare, which can be low for ensuring statistical significance. As results show, many tests obtained non-significant results. In general, no result for spacing is significant, and most significant results are obtained when comparing with FLGBM, either to indicate that it is better than the others in the measure being compared (hypervolume, overall Pareto front spread, error ratio, generational distance and inverted generational distance) or worse (maximum spread). The only other significant result is that FGP performed worse than FDT with respect to overall Pareto spread. Despite this, the p-values of the significant results are close to the non-rejection region of H_0 (close to 0.05),

Table 11.2.: Average and standard deviation execution times (in seconds) of each generation ($n_i = 150$) for each algorithm and dataset.

Data	DT		FDT		FGP		FGLBM	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
adult	13.3794	±3.2469	13.7548	±3.6269	5.5510	±0.7451	18.3624	±5.0958
compas	6.0641	±2.8333	6.3350	±2.9483	1.4155	±0.0982	5.5355	±2.2527
diabetes	14.8380	±3.3893	14.6732	±3.4842	37.9049	±11.0912	17.7911	±3.6953
dutch	8.2040	±3.4302	8.4544	±3.7945	4.8730	±0.6454	14.7113	±5.2672
german	5.4291	±2.7030	5.7034	±2.8335	1.2533	±0.0839	4.9551	±2.0875
insurance	5.2188	±2.6785	5.4318	±2.8009	6.3279	±2.2141	4.6394	±1.9561
obesity	5.7515	±2.8099	5.9475	±2.8483	9.3342	±2.5652	4.9849	±2.1742
parkinson	6.3512	±2.9206	6.5225	±3.0364	1.4408	±0.1065	6.4741	±2.8485
ricci	5.1444	±2.7464	5.2088	±2.7134	5.0419	±2.0133	4.5941	±2.0469
student	5.7184	±2.7713	5.9843	±2.8838	2.2501	±1.5600	5.2025	±2.2155

highlighting the need to increase the sample size of paired comparisons.

In Table 11.4, all results found for each algorithm and run were used. For each dataset and run, the Pareto fronts obtained for each algorithm were considered. Therefore, we had a total of 100 paired measurements for each quality indicator and pair of algorithms to compare (10 datasets with 10 different partitions each), which can be more benefitial to reach statistical significance. As shown, there are no $+$ or $-$ results in this table, and the p-values are generally lower, indicating that increasing the sample size improves the test results in terms of statistical significance. Almost all the tests that produced statistically significant results in Table 11.3 remained significant in Table 11.4 except for one: the comparison of generational distance between FLGBM and FGP. Otherwise, the results complement the rankings in Figure 11.35, showing which results are truly significant and which are not. Here, for example, it can be seen that the differences between DT and FDT are not significant in general (although there was a non-significant difference in hypervolume in Table 11.3). Even though there are no significant differences, we will prefer the FDT algorithm over DT for its generally better results in the rankings. The FGP algorithm performs significantly worse in all quality indicators except for generational distance compared to all other algorithms. FLGBM achieves significantly better results than the other algorithms except in the two quality indicators where it did not rank highest: spacing and maximum spread (even so, it is significantly better than FGP for these measures).

Finally, in Table 11.3, the results of the CPU processing times were compared. In the first subtable, the total CPU processing times for each run and algorithm are considered, resulting in a total of 100 paired samples for each pair of algorithms to compare. The second subtable, however, compares the execution time of each generation for each run and dataset, resulting in 30000 paired samples for each pair of algorithms to compare. The results are somewhat different from those obtained in Figure 11.35, and the same phenomenon is observed with an increased sample size. The best algorithm was FGP, which outperformed all the others, while the worst was FLGBM, which lost against all the others. In this case, as discussed before, FDT was significantly slower with respect to CPU processing time than DT.

Table 11.3.: Paired samples Wilcoxon signed-rank test for all quality indicators using average Pareto fronts for all datasets.

(a) Wilcoxon signed-rank test for Hypervolume indicator

	DT	FDT	FGP	FLGBM
DT		7.70×10^{-1}	1.31×10^{-1}	4.32×10^{-1}
FDT	$+$		1.05×10^{-1}	2.32×10^{-1}
FGP	$-$	$-$		1.95×10^{-3}
FLGBM	$+$	$+$	$\oplus$	

(b) Wilcoxon signed-rank test for Spacing indicator

	DT	FDT	FGP	FLGBM
DT		3.74×10^{-1}	2.14×10^{-1}	1.73×10^{-1}
FDT	$=$		3.74×10^{-1}	5.06×10^{-2}
FGP	$=$	$=$		8.89×10^{-1}
FLGBM	$=$	$-$	$=$	

(c) Wilcoxon signed-rank test for Maximum Spread indicator

	DT	FDT	FGP	FLGBM
DT		4.32×10^{-1}	5.57×10^{-1}	3.71×10^{-2}
FDT	$=$		2.32×10^{-1}	9.77×10^{-3}
FGP	$=$	$-$		4.32×10^{-1}
FLGBM	$\ominus$	$\ominus$	$-$	

(d) Wilcoxon signed-rank test for Overall Pareto Front Spread indicator

	DT	FDT	FGP	FLGBM
DT		3.14×10^{-1}	2.60×10^{-1}	1.37×10^{-2}
FDT	$=$		3.82×10^{-2}	1.60×10^{-1}
FGP	$=$	$\ominus$		4.88×10^{-2}
FLGBM	$\oplus$	$+$	$\oplus$	

(e) Wilcoxon signed-rank test for Error Ratio indicator

	DT	FDT	FGP	FLGBM
DT		1.61×10^{-1}	5.94×10^{-1}	1.95×10^{-3}
FDT	$=$		9.69×10^{-2}	1.09×10^{-1}
FGP	$=$	$=$		1.51×10^{-2}
FLGBM	$\oplus$	$=$	$\oplus$	

(f) Wilcoxon signed-rank test for Generational Distance indicator

	DT	FDT	FGP	FLGBM
DT		9.22×10^{-1}	2.32×10^{-1}	3.71×10^{-2}
FDT	$=$		9.22×10^{-1}	3.82×10^{-2}
FGP	$-$	$=$		2.73×10^{-2}
FLGBM	$\oplus$	$\oplus$	$\oplus$	

Table 11.3.: Paired samples Wilcoxon signed-rank test for all quality indicators using average Pareto fronts for all datasets cont.

(g) Wilcoxon signed-rank test for Inverted Generational Distance indicator

	DT	FDT	FGP	FLGBM
DT		6.25×10^{-1}	1.37×10^{-2}	1.31×10^{-1}
FDT	$=$		8.40×10^{-2}	1.31×10^{-1}
FGP	$\ominus$	$=$		9.77×10^{-3}
FLGBM	$+$	$+$	$\oplus$	

Table 11.4.: Paired samples Wilcoxon signed-rank test for all quality indicators using Pareto fronts from all runs and datasets.

(a) Wilcoxon signed-rank test for Hypervolume indicator

	DT	FDT	FGP	FLGBM
DT		8.55×10^{-1}	2.41×10^{-7}	2.33×10^{-2}
FDT	$=$		3.17×10^{-6}	3.21×10^{-3}
FGP	$\ominus$	$\ominus$		2.39×10^{-8}
FLGBM	$\oplus$	$\oplus$	$\oplus$	

(b) Wilcoxon signed-rank test for Spacing indicator

	DT	FDT	FGP	FLGBM
DT		2.58×10^{-1}	2.52×10^{-10}	3.02×10^{-6}
FDT	$=$		2.17×10^{-10}	6.59×10^{-8}
FGP	$\ominus$	$\ominus$		1.01×10^{-5}
FLGBM	$\ominus$	$\ominus$	$\oplus$	

(c) Wilcoxon signed-rank test for Maximum Spread indicator

	DT	FDT	FGP	FLGBM
DT		5.82×10^{-1}	2.94×10^{-11}	7.55×10^{-8}
FDT	$=$		4.33×10^{-11}	3.12×10^{-10}
FGP	$\ominus$	$\ominus$		4.20×10^{-7}
FLGBM	$\ominus$	$\ominus$	$\oplus$	

(d) Wilcoxon signed-rank test for Overall Pareto Front Spread indicator

	DT	FDT	FGP	FLGBM
DT		9.96×10^{-1}	8.40×10^{-7}	1.99×10^{-2}
FDT	$=$		1.00×10^{-9}	2.01×10^{-3}
FGP	$\ominus$	$\ominus$		1.27×10^{-6}
FLGBM	$\oplus$	$\oplus$	$\oplus$	

Table 11.4.: Paired samples Wilcoxon signed-rank test for all quality indicators using Pareto fronts from all runs and datasets cont.

(e) Wilcoxon signed-rank test for Error Ratio indicator

	DT	FDT	FGP	FLGBM
DT		7.55×10^{-1}	1.44×10^{-3}	5.29×10^{-4}
FDT	$=$		1.76×10^{-3}	2.70×10^{-3}
FGP	$\ominus$	$\ominus$		4.23×10^{-7}
FLGBM	$\oplus$	$\oplus$	$\oplus$	

(f) Wilcoxon signed-rank test for Generational Distance indicator

	DT	FDT	FGP	FLGBM
DT		3.11×10^{-1}	5.79×10^{-1}	4.04×10^{-2}
FDT	$=$		6.18×10^{-1}	2.72×10^{-3}
FGP	$=$	$=$		1.53×10^{-1}
FLGBM	$\oplus$	$\oplus$	$=$	

(g) Wilcoxon signed-rank test for Inverted Generational Distance indicator

	DT	FDT	FGP	FLGBM
DT		7.50×10^{-1}	1.01×10^{-7}	5.25×10^{-1}
FDT	$=$		4.99×10^{-6}	7.21×10^{-1}
FGP	$\ominus$	$\ominus$		3.03×10^{-6}
FLGBM	$=$	$=$	$\oplus$	

Table 11.5.: Paired samples Wilcoxon signed-rank test for all processing times using results for all datasets

(a) Wilcoxon signed-rank test using the total processing times from each run.

	DT	FDT	FGP	FLGBM
DT		1.26×10^{-11}	5.15×10^{-4}	4.13×10^{-1}
FDT	$\ominus$		2.86×10^{-4}	8.10×10^{-1}
FGP	$\oplus$	$\oplus$		6.98×10^{-4}
FLGBM	$+$	$+$	$\ominus$	

(b) Wilcoxon signed-rank test using the processing times of each generation from each run.

	DT	FDT	FGP	FLGBM
DT		0.00	0.00	8.25×10^{-160}
FDT	$\ominus$		0.00	4.88×10^{-47}
FGP	$\oplus$	$\oplus$		0.00
FLGBM	$\ominus$	$\ominus$	$\ominus$	

11.3. Graphs of hyperparameter values in the Pareto-optimal sets

Let us also include images showing the distribution of hyperparameters among the individuals belonging to the Pareto-optimal sets from all runs for each algorithm that uses hyperparameters (DT,FDT and FLGBM) using violin plots, as well as their correlation using heatmaps. We will explain the content of these graphs below:

- **DT algorithm**: Figures 11.36, 11.39, 11.42, 11.45, 11.48, 11.51, 11.54, 11.57, 11.60 and 11.63 show the distribution of hyperparameters in the Pareto-optimal sets found using the DT algorithm for each respective dataset using violin plots. These plots illustrate the range and distribution of hyperparameter values optimized during the process. Additionally, correlation plots using heatmaps are included to evaluate relationships between hyperparameter configurations for each specific problem. Results exhibit the following patterns:
  - **criterion**: This parameter does not exhibit any special structure; its value depends on the dataset and algorithm. It may tend more towards 0, towards 1, or not tend to either one.
  - **max_depth**: They tend to concentrate around the central values of their distribution, which vary depending on the dataset (usually 6 or 10).
  - **min_samples_split**: This hyperparameter is generally quite varied, tending towards medium values but being widely distributed.
  - **max_leaf_nodes**: This hyperparameter also takes varied values, although it is not common to find values in the high part of the assigned search range.
  - **class_weight**: This parameter depends heavily on the dataset, but they tend to concentrate significantly around a certain value (which probably is the one that provides the optimal balance between the two classes). The scale of this value is as follows: for values $i \in \{1, \ldots, 10\}$, the class weight used is $\frac{1}{11-i}$, while for values $i \in \{11, \ldots, 19\}$, the class weight used is $i - 9$.
  - **Correlation heatmaps**: There are no clear patterns; there are few high absolute values, which are also inconsistent across datasets. For compas dataset, it is interesting to see how all correlations approach very close to 0.
- **FDT algorithm**: Figures 11.37, 11.40, 11.43, 11.46, 11.49, 11.52, 11.55, 11.58, 11.61 and 11.64 include analogous information to the previous point but for the FDT algorithm. Results exhibit the following patterns:
  - All parameters shared with DT tend to exhibit a structure relatively similar to that found for that algorithm.
  - **fair_param**: This parameter tends to take relatively low values in some datasets (such as Parkinson or obesity) or relatively varied values in other datasets (such as insurance or German). Depending on the dataset, the algorithm focuses on low or more varied values to find the best models.
  - **Correlation heatmaps**: There are no clear patterns for all datasets either.
- **FLGBM algorithm**: Figures 11.38, 11.41, 11.44, 11.47, 11.50, 11.53, 11.56, 11.59, 11.62 and 11.65 include analogous information to the previous point but for the FLGBM algorithm. Results exhibit the following patterns:

  - **All parameters except fair_param**: These tend to consistently have varied values, not taking too many extreme values.
  - **fair_param**: This parameter tends to take relatively varied values for every dataset using FLGBM, in contrast to what happen with FDT algorithm.
  - **Correlation heatmaps**: The correlations are generally slightly higher compared to the other two algorithms, but there are no significant results either. In some datasets, moderate direct correlations are observed among all hyperparameters, but this trend is not consistent across all datasets.

- **Graphs using every individual**: Finally, in Figures 11.66, 11.67, 11.68, analogous information is presented, taking into account all Pareto-optimal individuals found across all datasets. This allows for a comprehensive view to determine whether there is a general distribution pattern for the hyperparameters and their relationships, independent of specific datasets. The results exhibit a behavior analogous to that commented on previously for the figures of each specific dataset.

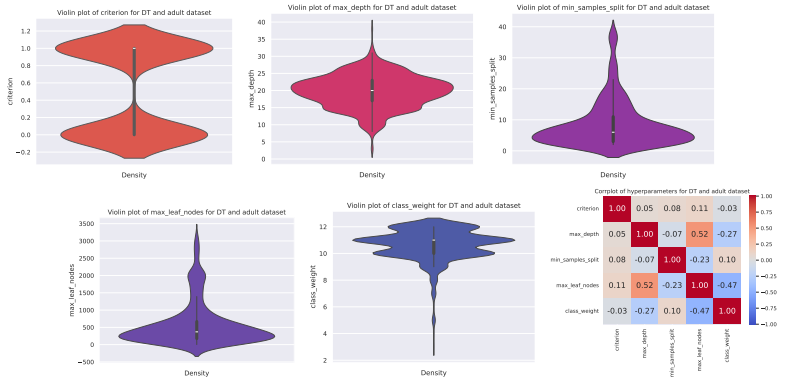


Figure 11.36.: Hyperparameter distribution for Pareto front found using DT for adult dataset.

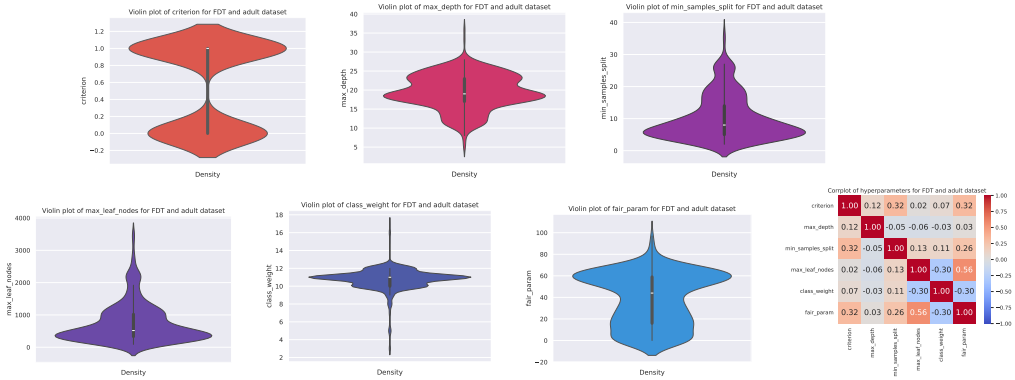


Figure 11.37.: Hyperparameter distribution for Pareto front found using FDT for adult dataset.

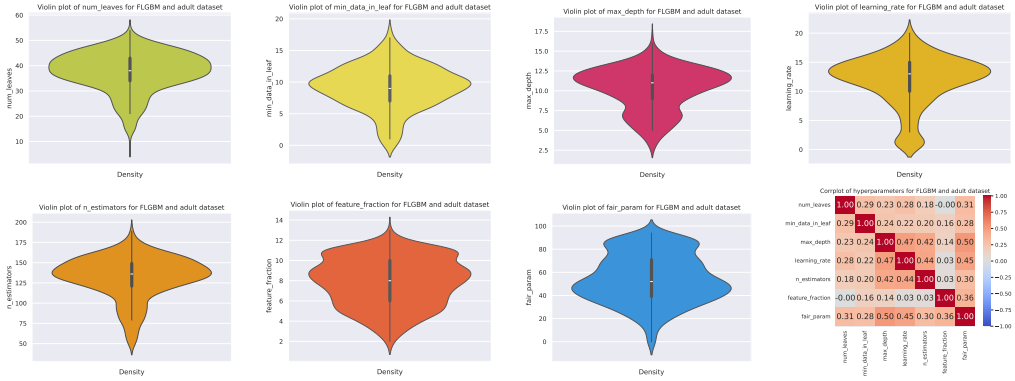


Figure 11.38.: Hyperparameter distribution for Pareto front found using FLGBM for adult dataset.

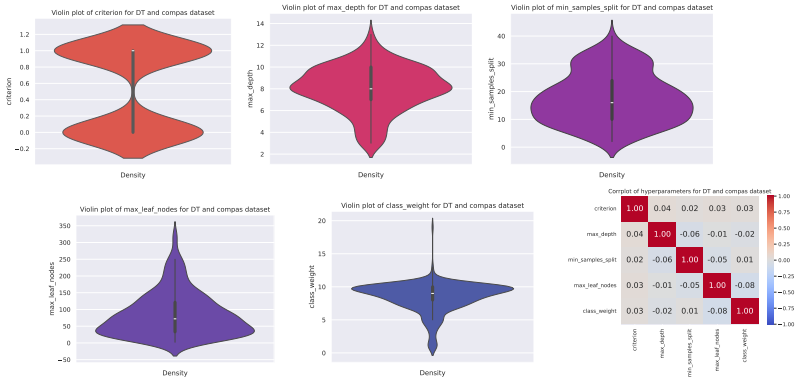


Figure 11.39.: Hyperparameter distribution for Pareto front found using DT for compas dataset.

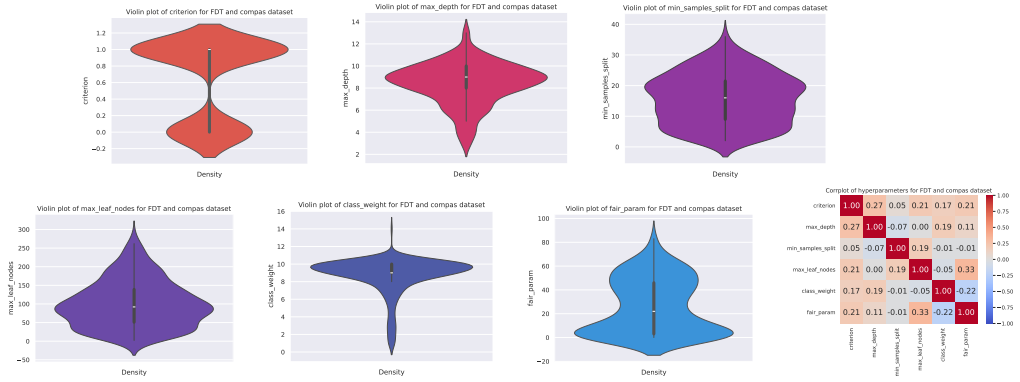


Figure 11.40.: Hyperparameter distribution for Pareto front found using FDT for compas dataset.

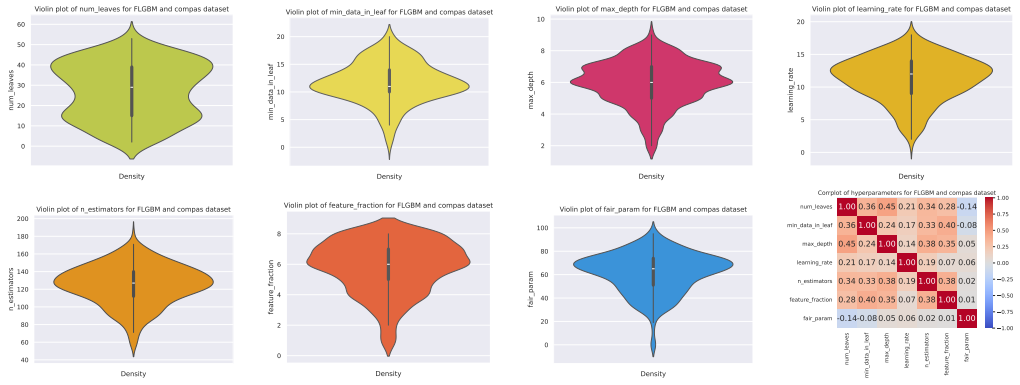


Figure 11.41.: Hyperparameter distribution for Pareto front found using FLGBM for compas dataset.

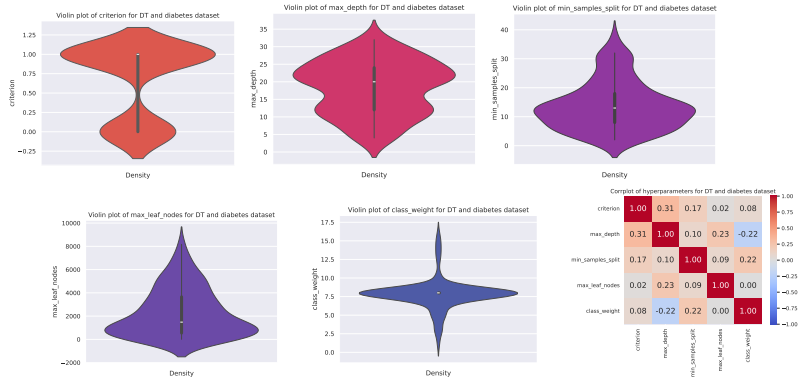


Figure 11.42.: Hyperparameter distribution for Pareto front found using DT for diabetes dataset.

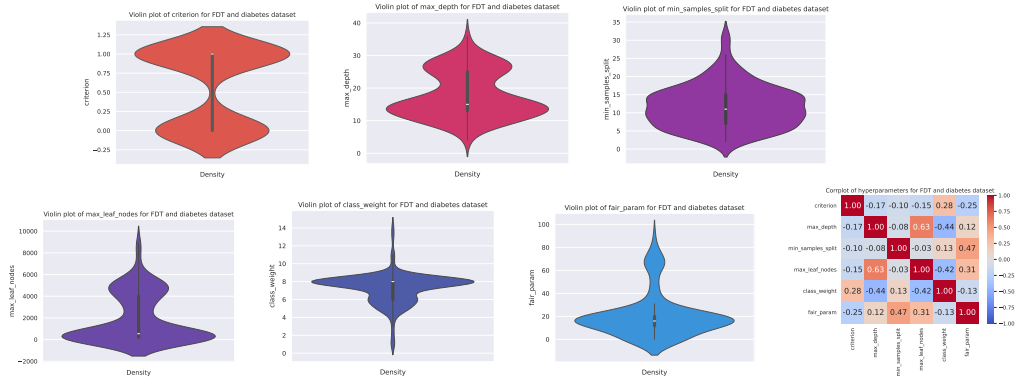


Figure 11.43.: Hyperparameter distribution for Pareto front found using FDT for diabetes dataset.

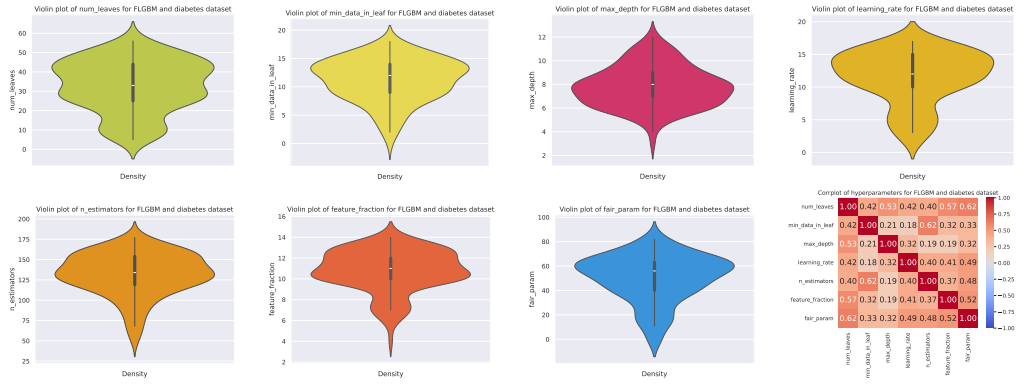


Figure 11.44.: Hyperparameter distribution for Pareto front found using FLGBM for diabetes dataset.

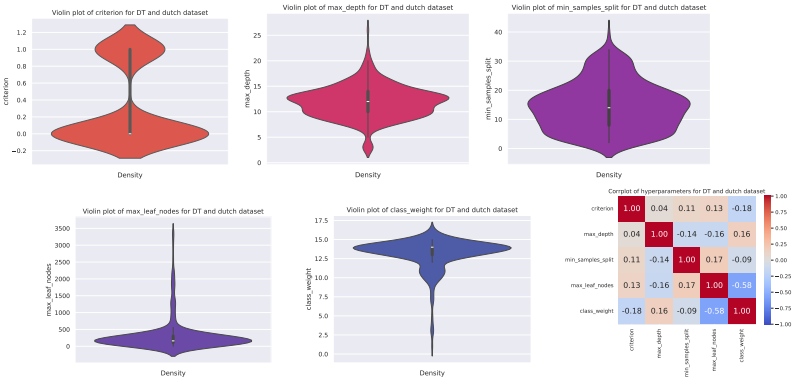


Figure 11.45.: Hyperparameter distribution for Pareto front found using DT for dutch dataset.

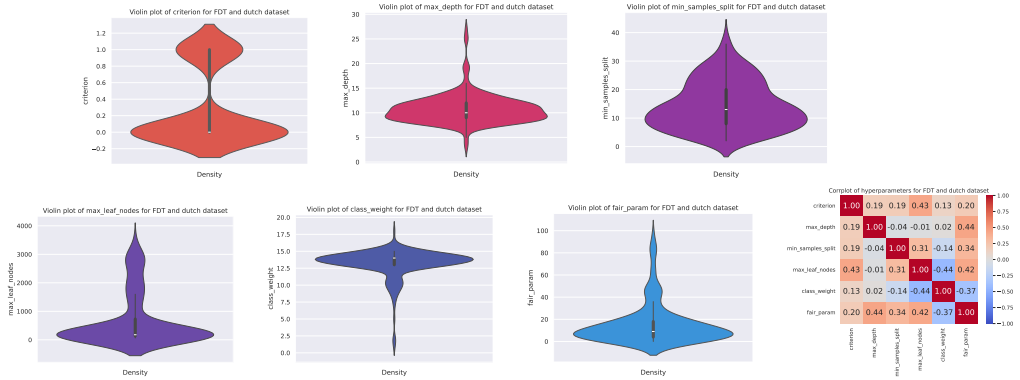


Figure 11.46.: Hyperparameter distribution for Pareto front found using FDT for dutch dataset.

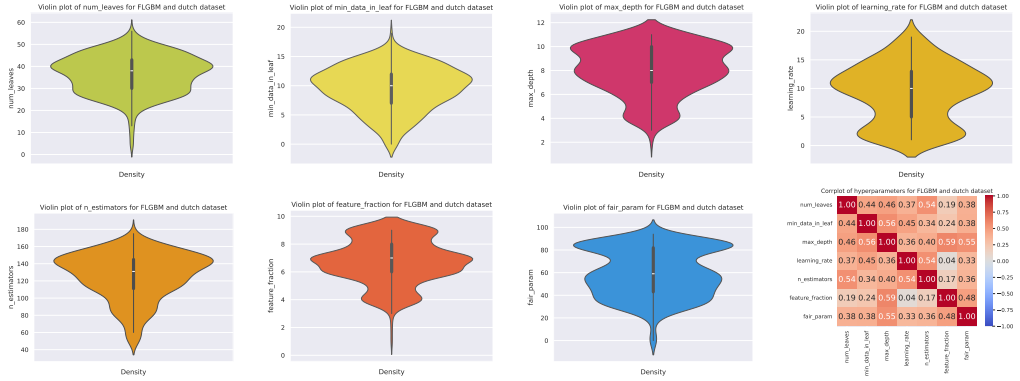


Figure 11.47.: Hyperparameter distribution for Pareto front found using FLGBM for dutch dataset.

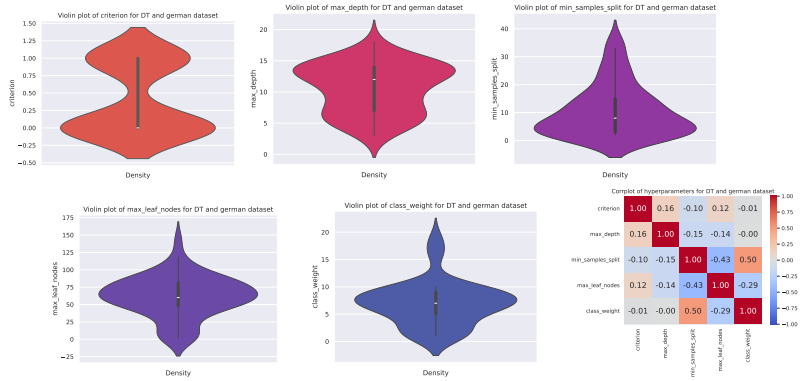


Figure 11.48.: Hyperparameter distribution for Pareto front found using DT for german dataset.

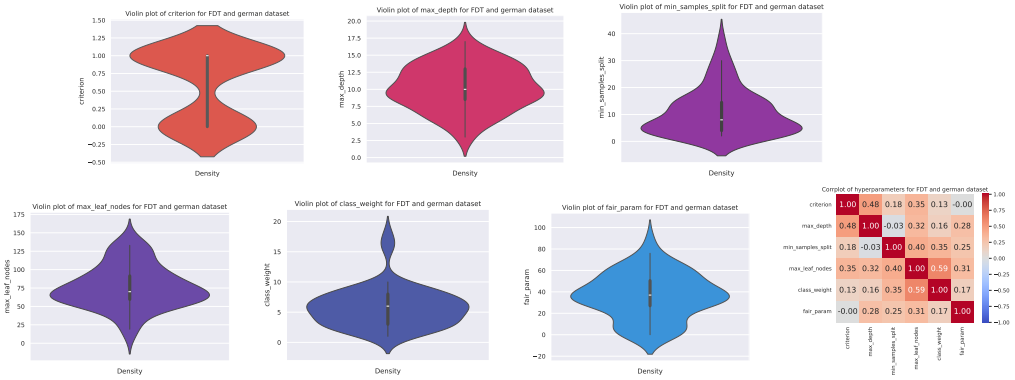


Figure 11.49.: Hyperparameter distribution for Pareto front found using FDT for german dataset.

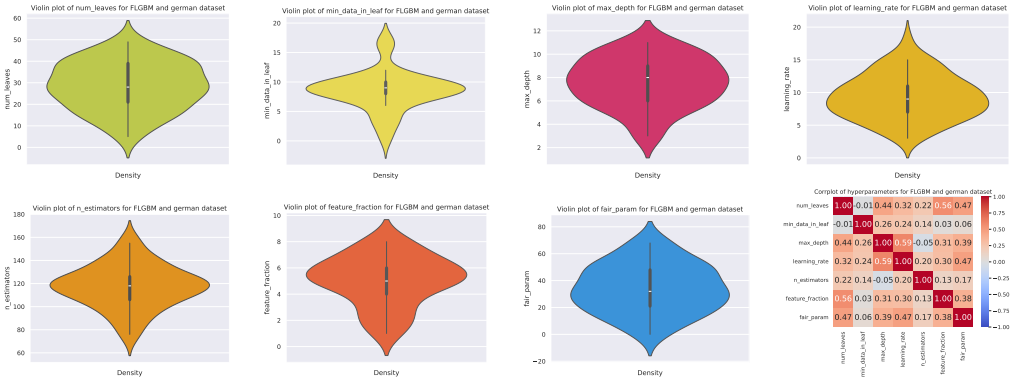


Figure 11.50.: Hyperparameter distribution for Pareto front found using FLGBM for german dataset.

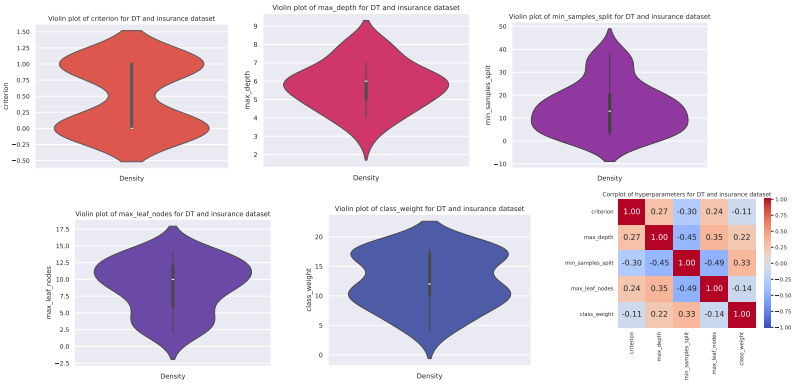


Figure 11.51.: Hyperparameter distribution for Pareto front found using DT for insurance dataset.

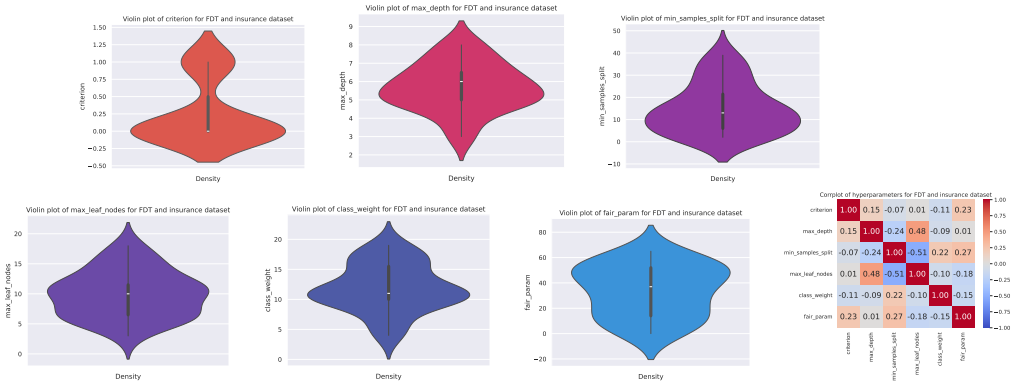


Figure 11.52.: Hyperparameter distribution for Pareto front found using FDT for insurance dataset.

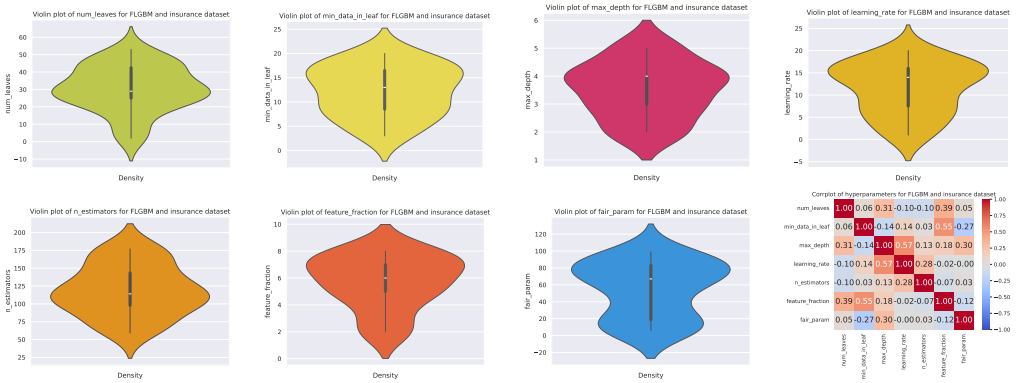


Figure 11.53.: Hyperparameter distribution for Pareto front found using FLGBM for insurance dataset.

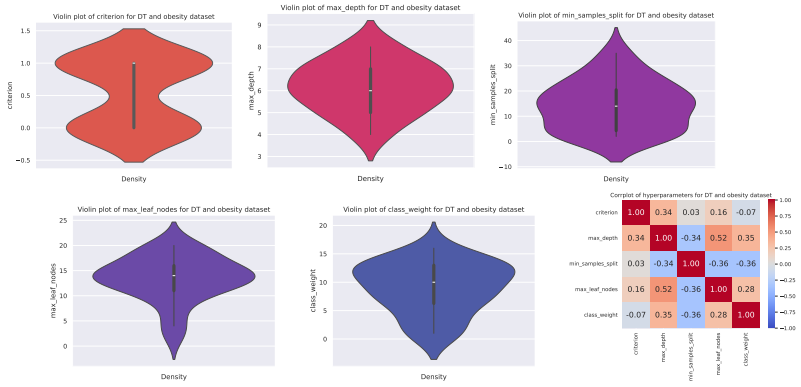


Figure 11.54.: Hyperparameter distribution for Pareto front found using DT for obesity dataset.

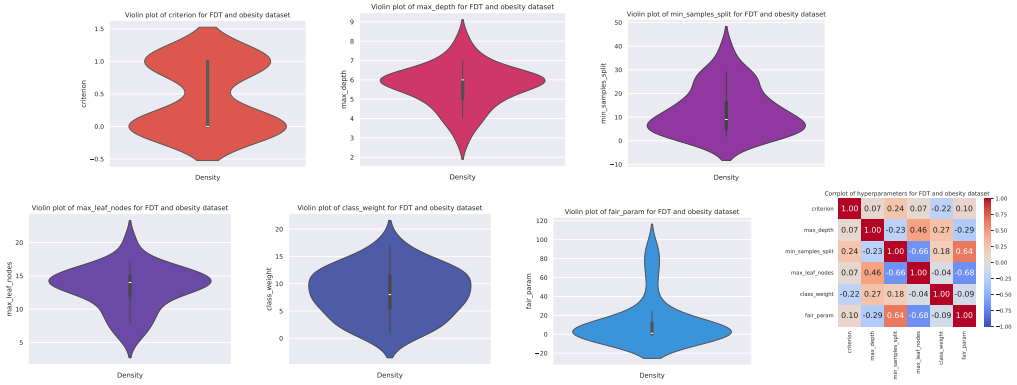


Figure 11.55.: Hyperparameter distribution for Pareto front found using FDT for insurance dataset.

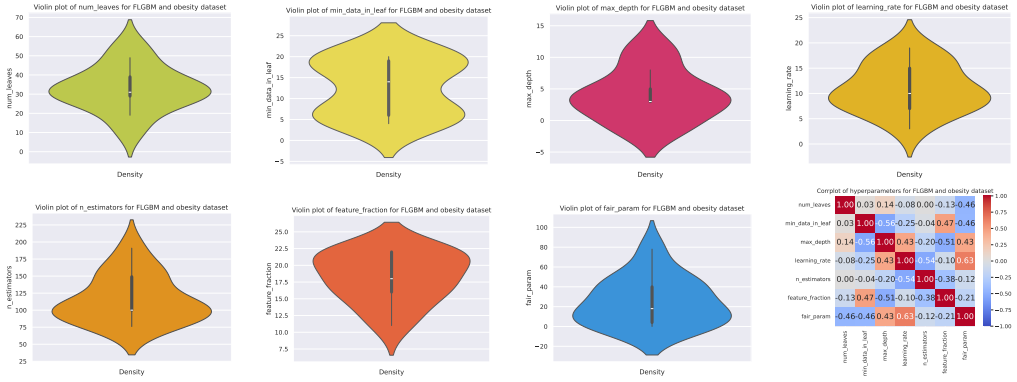


Figure 11.56.: Hyperparameter distribution for Pareto front found using FLGBM for insurance dataset.

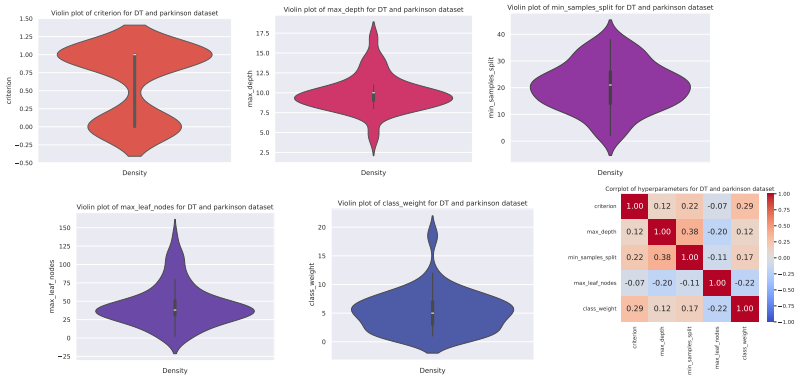


Figure 11.57.: Hyperparameter distribution for Pareto front found using DT for parkinson dataset.

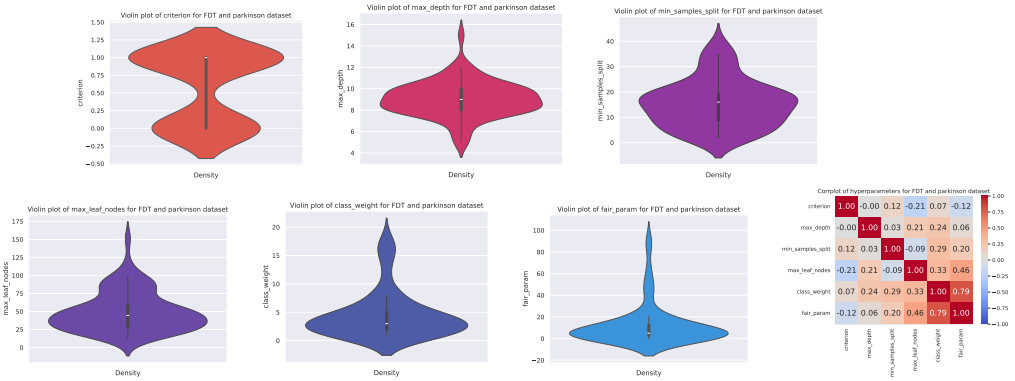


Figure 11.58.: Hyperparameter distribution for Pareto front found using FDT for parkinson dataset.

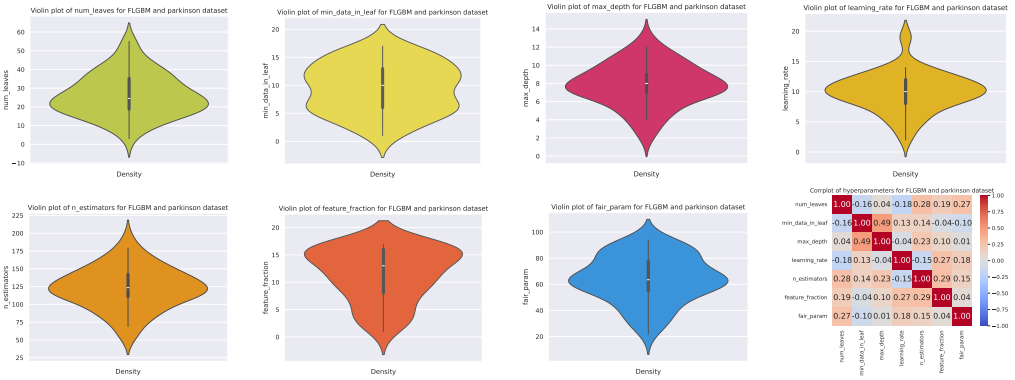


Figure 11.59.: Hyperparameter distribution for Pareto front found using FLGBM for parkinson dataset.

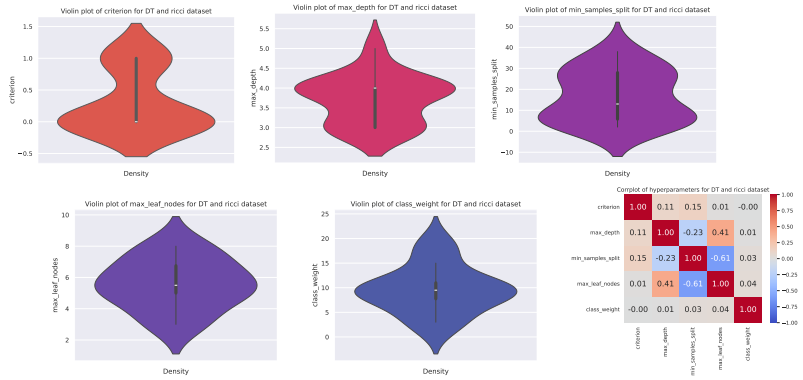


Figure 11.60.: Hyperparameter distribution for Pareto front found using DT for ricci dataset.

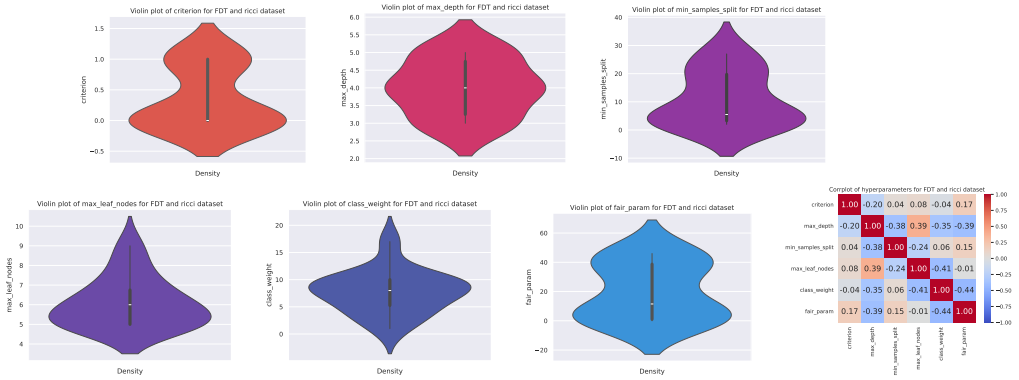


Figure 11.61.: Hyperparameter distribution for Pareto front found using FDT for ricci dataset.

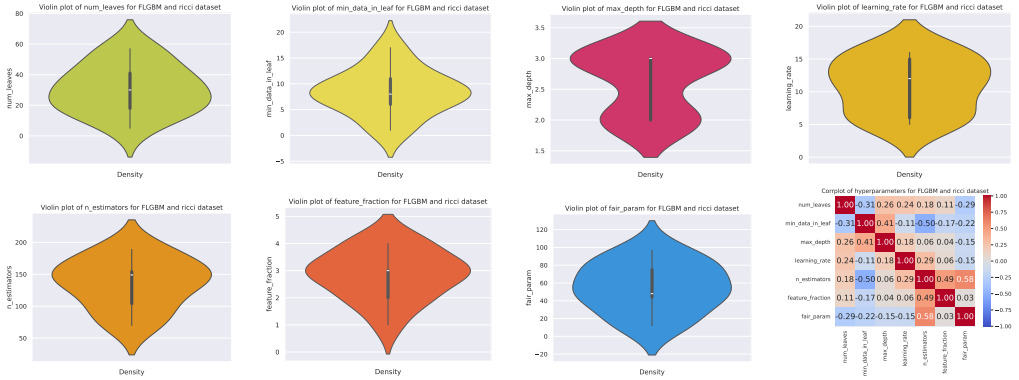


Figure 11.62.: Hyperparameter distribution for Pareto front found using FLGBM for ricci dataset.

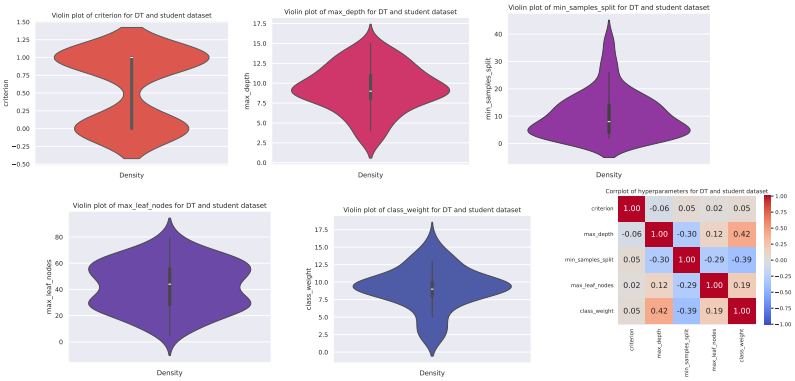


Figure 11.63.: Hyperparameter distribution for Pareto front found using DT for student dataset.

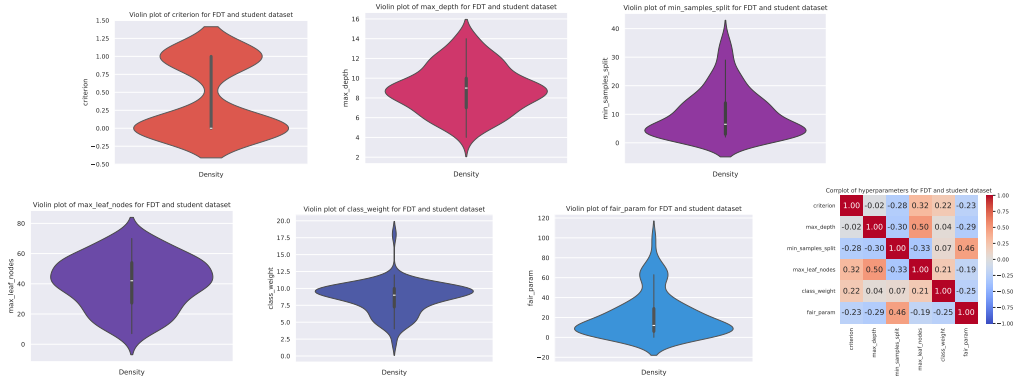


Figure 11.64.: Hyperparameter distribution for Pareto front found using FDT for student dataset.

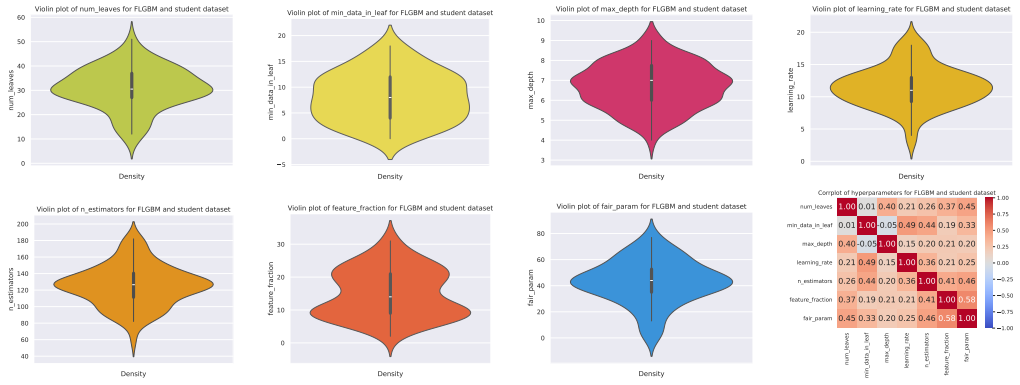


Figure 11.65.: Hyperparameter distribution for Pareto front found using FLGBM for student dataset.

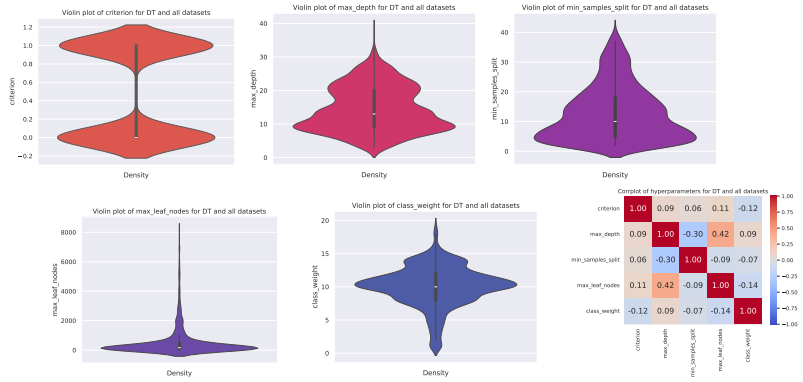


Figure 11.66.: Hyperparameter distribution for Pareto front found using DT for all datasets.

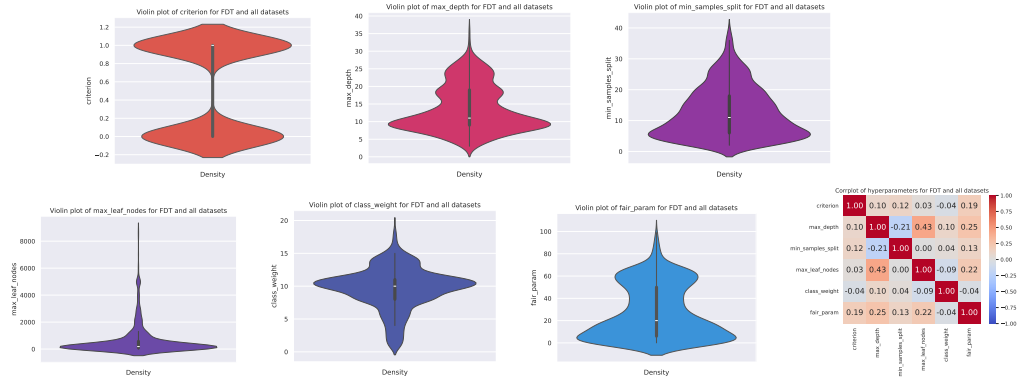


Figure 11.67.: Hyperparameter distribution for Pareto front found using FDT for all datasets.

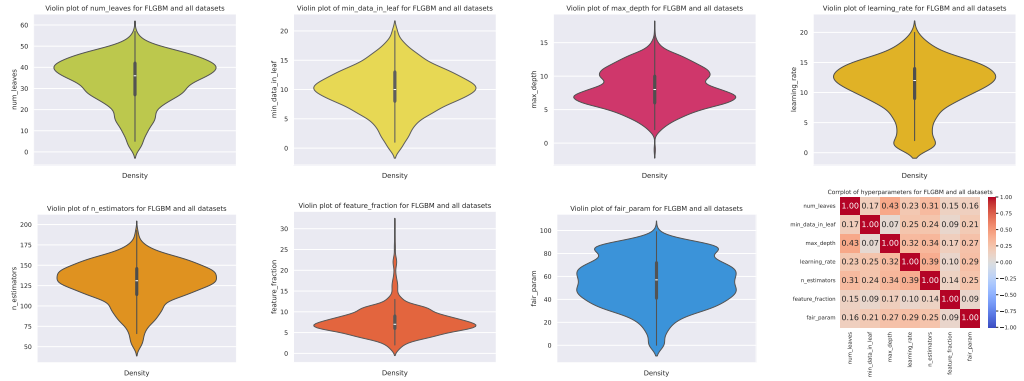


Figure 11.68.: Hyperparameter distribution for Pareto front found using FLGBM for all datasets.

12. Conclusions

After having analyzed the theoretical context, explained each developed algorithm in detail, and conducted the experimentation and analyzed the results, we can draw some clear conclusions.

To begin with, it is possible to improve upon the base algorithm. As we have observed, FDT and FLGBM algorithms exhibit behavior that is generally more optimal for a multi-objective optimization process than the DT algorithm. This can be confirmed by examining the quality indicators found for the average Pareto fronts produced by the algorithms. It is not possible to surpass the base algorithm in all quality indicators, but the improvement margin of the FLGBM algorithm compared to DT in hypervolume, overall Pareto front spread, error ratio, generational distance, and inverted generational distance is clear and statistically significant. Additionally, it does not have poor CPU processing times.

On the other hand, despite not showing statistically significant differences compared to the base algorithm, FDT algorithm is capable of improving or matching the base algorithm in almost all quality indicators (except generational distance). This gives us a reason to prefer its use over the base algorithm. We can thus see how the inclusion of a simple parameter that affects the learning process of the algorithm by incorporating a fairness measure can significantly improve the quality of the models found in multi-objective optimization processes, including that same fairness measure. The CPU processing time worsens, but the difference is extremely low, especially when compared to the better models with respect to error and fairness that it can learn.

Although FGP algorithm achieved reasonably similar results when observing the average Pareto fronts found by each algorithm, it did not obtain results as good as the other developed algorithms, performing even worse than the base algorithm. This is because, although it finds very good solutions for some data partitions, it cannot obtain equally good results for others, worsening the average Pareto fronts found. Despite being the winner in terms of minimum CPU processing time with statistically significant results when compared to all other algorithms, discretion is advised, as it may have some scaling issues depending on the structure of the tree matrix.

As can also be seen, it is possible to significantly improve some classic models such as Northpointe's COMPAS model by a wide margin in the two employed objectives.

The study conducted on the values of the hyperparameters found in the Pareto-optimal sets of the algorithms that use hyperparameters shows that, although some patterns are observed, particularly with the parameter λ in FDT and FLGBM and in the common parameters used in DT and FDT, there is generally not much correlation between their values.

This study demonstrates that there is still room for improvement in the multi-objective opti-

mization of accuracy and fairness. The development of learning algorithms that incorporate fairness in their internal processing is an area that is just beginning to be explored and is obtaining promising results, as found in this work. The developed methods are proposed as good alternatives (FDT and FLGBM) or potential alternatives (FGP) to algorithms that incorporate fairness measures during learning. The theoretical and practical study conducted on them supports their effectiveness, and these solid foundations enhance their potential use.

However, much research and experimentation are needed in this area of machine learning. In this study, we have only developed binary classification algorithms and tested them using datasets containing one binary sensitive attribute, employing one objective for accuracy and another for fairness. Yet, many other alternatives could be explored: experimenting with regression problems, testing individual or counterfactual fairness, using non-binary or even continuous sensitive attributes, considering more than one sensitive attribute, employing multiple mutually incompatible fairness objectives, and delving into the realm of Many-objectives optimization... The possibilities are vast, and the ideas presented in this work pave the way for future research to refine the models presented here or create new ones. There is still a long way to go to achieve fairness, but this work "just" takes a step in that "right" direction.

12.1. Future work

A wide variety of different future lines of research related to the current work can be proposed. Some of them are:

- **Experimentation with more datasets**: It would be beneficial to increase the number of datasets used, ensuring diversity in terms of the number of instances, attributes, and contexts they belong to. Especially, testing with higher-dimensional datasets should be attempted, although these are currently scarce in the field of fairness in machine learning.
- **Improvements of each algorithm**: Overall, striving to refine each algorithm to achieve even more optimized results, but focusing particularly on the following:
- **Improvements to FGP algorithm**: This algorithm can likely be enhanced for scalability when handling large matrix trees, as well as to achieve more stable results that are less dependent on the data partitioning.
- **Analysis of objective convergence through generations**: Examining the evolution of objective values over generations, similar to what is done with structural convergence.
- **Many-objectives approach**: Test these algorithms using a many-objectives environment, where at least one accuracy measure and two mutually incompatible fairness measures are considered. This way, we can verify how the algorithms behave in other, more complex execution environments.
- **Consider other problems**: Such as problems with more than one sensitive attribute, a non-binary sensitive attribute, regression problems, the use of other fairness measures...

Bibliography

References are listed in alphabetical order. References with more than one author are sorted according to the first author.

[pro, 2009] (2009). Ricci v. destefano. Technical Report 557 U.S. 557, S. C. of the United States. [Citado en págs. 7 and 101]

[obe, 2019] (2019). Estimation of Obesity Levels Based On Eating Habits and Physical Condition . UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5H31Z. [Citado en pág. 100]

[Abu-Mostafa et al., 2012] Abu-Mostafa, Y. S., Magdon-Ismail, M., and Lin, H.-T. (2012). *Learning from data*, volume 4. AMLBook New York, NY, USA:. [Citado en pág. 13]

[Agarwal et al., 2018] Agarwal, A., Beygelzimer, A., Dudík, M., Langford, J., and Wallach, H. (2018). A reductions approach to fair classification. *arXiv preprint arXiv:1803.02453*. [Citado en pág. 10]

[Angwin et al., 2016] Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine bias. *ProPublica, May*, 23:2016. [Citado en pág. 7]

[Audet et al., 2020] Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., and Salomon, L. (2020). Performance indicators in multiobjective optimization. *European journal of operational research*. [Citado en pág. 51]

[Balashankar et al., 2019] Balashankar, A., Lees, A., Welty, C., and Subramanian, L. (2019). Pareto-efficient fairness for skewed subgroup data. In *International Conference on Machine Learning AI for Social Good Workshop. Long Beach, United States*, volume 8. [Citado en pág. 10]

[Becker and Kohavi, 1996] Becker, B. and Kohavi, R. (1996). Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20. [Citado en pág. 98]

[Biddle, 2006] Biddle, D. (2006). *Adverse impact and test validation: A practitioner's guide to valid and defensible employment testing*. Gower Publishing, Ltd. [Citado en pág. 27]

[Binns, 2020] Binns, R. (2020). On the apparent conflict between individual and group fairness. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 514–524. [Citado en pág. 26]

[Binns et al., 2018] Binns, R., Van Kleek, M., Veale, M., Lyngs, U., Zhao, J., and Shadbolt, N. (2018). 'it's reducing a human being to a percentage' perceptions of justice in algorithmic decisions. In *Proceedings of the 2018 Chi conference on human factors in computing systems*, pages 1–14. [Citado en pág. 7]

[Bolukbasi et al., 2016] Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., and Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*, pages 4349–4357. [Citado en págs. 7 and 40]

[Castelnovo, 2022] Castelnovo, A. (2022). Extending decision tree to handle multiple fairness criteria. In Raedt, L. D., editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5839–5840. International Joint Conferences on Artificial Intelligence Organization. Doctoral Consortium. [Citado en págs. 11 and 69]

[Centraal Bureau voor de Statistiek (CBS) (Statistics Netherlands), 2018] Centraal Bureau voor de Statistiek (CBS) (Statistics Netherlands), M. P. C. (2018). The dutch virtual census of 2001 - ipums subset. The World Bank Microdata repository. [Citado en pág. 99]

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. [Citado en pág. 88]

[Chouldechova, 2017] Chouldechova, A. (2017). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163. [Citado en pág. 12]

[Clore and Strack, 2014] Clore, John, C. K. D. J. and Strack, B. (2014). Diabetes 130-us hospitals for years 1999-2008. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5230J. [Citado en pág. 99]

[Copas, 1983] Copas, J. (1983). Regression, prediction and shrinkage. *Journal of the royal statistical society series b-methodological*, 45:311–335. [Citado en pág. 87]

[Corbett-Davies and Goel, 2018] Corbett-Davies, S. and Goel, S. (2018). The measure and mismeasure of fairness: A critical review of fair machine learning. [Citado en pág. 31]

[Corne and Knowles, 2007] Corne, D. W. and Knowles, J. D. (2007). Techniques for highly multiobjective optimisation: some nondominated points are better than others. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 773–780. [Citado en pág. 61]

[Cortez, 2014] Cortez, P. (2014). Student Performance. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5TG7T. [Citado en pág. 101]

[Cruz et al., 2023] Cruz, A., Belém, C. G., Bravo, J., Saleiro, P., and Bizarro, P. (2023). FairGBM: Gradient boosting with fairness constraints. In *The Eleventh International Conference on Learning Representations*. [Citado en págs. 11 and 94]

[Danner et al., 2016] Danner, M. J., VanNostrand, M., and Spruance, L. M. (2016). Race and gender neutral pretrial risk assessment, release recommendations, and supervision: Vprai and praxis revised. *Luminosity, Inc*. [Citado en pág. 31]

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197. [Citado en pág. 60]

[del Barrio et al., 2020] del Barrio, E., Gordaliza, P., and Loubes, J.-M. (2020). Review of mathematical frameworks for fairness in machine learning. [Citado en pág. 33]

[Deza and Deza, 2009] Deza, M. M. and Deza, E. (2009). Encyclopedia of distances. In *Encyclopedia of distances*, pages 1–583. Springer. [Citado en pág. 39]

[Domingos, 2000] Domingos, P. (2000). A unified bias-variance decomposition. In *Proceedings of 17th International Conference on Machine Learning*, pages 231–238. [Citado en pág. 16]

[Dwork et al., 2011] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. (2011). Fairness through awareness. [Citado en págs. 36 and 37]

[Eubanks, 2018] Eubanks, V. (2018). *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press. [Citado en pág. 7]

[Fabris et al., 2022] Fabris, A., Messina, S., Silvello, G., and Susto, G. A. (2022). Algorithmic fairness datasets: the story so far. *Data Min. Knowl. Discov.*, 36(6):2074–2152. [Citado en pág. 98]

[Fabris et al., 2021] Fabris, A., Mishler, A., Gottardi, S., Carletti, M., Daicampi, M., Susto, G. A., and Silvello, G. (2021). Algorithmic audit of italian car insurance: Evidence of unfairness in access and pricing. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, page 458–468, New York, NY, USA. Association for Computing Machinery. [Citado en pág. 100]

[Fonseca and Fleming, 1998] Fonseca, C. M. and Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 28(1):26–37. [Citado en pág. 61]

[Friedler et al., 2019] Friedler, S. A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E. P., and Roth, D. (2019). A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 329–338. [Citado en págs. 7 and 99]

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232. [Citado en págs. 83 and 88]

[Galles and Pearl, 1998] Galles, D. and Pearl, J. (1998). An axiomatic characterization of causal counterfactuals. *Foundations of Science*, 3(1):151–182. [Citado en pág. 42]

[Garg et al., 2020] Garg, P., Villasenor, J., and Foggo, V. (2020). Fairness metrics: A comparative analysis. [Citado en pág. 31]

[Hofmann, 1994] Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77. [Citado en pág. 100]

[Hu and Chen, 2018] Hu, L. and Chen, Y. (2018). A short-term intervention for long-term fairness in the labor market. *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. [Citado en pág. 27]

[Hu and Chen, 2020] Hu, L. and Chen, Y. (2020). Fair classification and social welfare. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 535–545. [Citado en pág. 10]

[Ilvento, 2020] Ilvento, C. (2020). Metric learning for individual fairness. [Citado en pág. 39]

[Johnson and Zhang, 2014] Johnson, R. and Zhang, T. (2014). Learning nonlinear functions using regularized greedy forest. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5):942–954. [Citado en pág. 88]

[Julia et al., 2016] Julia, Angwin, J., and Larson (2016). Machine bias. [Citado en págs. 9 and 99]

[Kamiran et al., 2010] Kamiran, F., Calders, T., and Pechenizkiy, M. (2010). Discrimination aware decision tree learning. In *2010 IEEE International Conference on Data Mining*, pages 869–874. IEEE. [Citado en pág. 10]

[Ke et al., 2017] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3149–3157, Red Hook, NY, USA. Curran Associates Inc. [Citado en pág. 88]

[Kleinberg et al., 2016] Kleinberg, J., Mullainathan, S., and Raghavan, M. (2016). Inherent trade-offs in the fair determination of risk scores. [Citado en pág. 32]

[Knowles and Corne, 2007] Knowles, J. and Corne, D. (2007). Quantifying the effects of objective space dimension in evolutionary multiobjective optimization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 757–771. Springer. [Citado en pág. 61]

[Kusner et al., 2018] Kusner, M. J., Loftus, J. R., Russell, C., and Silva, R. (2018). Counterfactual fairness. [Citado en pág. 41]

[Li et al., 2015] Li, B., Li, J., Tang, K., and Yao, X. (2015). Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 48(1):1–35. [Citado en págs. 12 and 61]

[Li et al., 2014] Li, M., Yang, S., and Liu, X. (2014). Diversity comparison of pareto front approximations in many-objective optimization. *IEEE Transactions on Cybernetics*, 44(12):2568–2584. [Citado en pág. 51]

[Martos Venturini, 2015] Martos Venturini, G. A. (2015). Statistical distances and probability metrics for multivariate data, ensembles and probability distributions. [Citado en pág. 37]

[Menon and Williamson, 2018] Menon, A. K. and Williamson, R. C. (2018). The cost of fairness in binary classification. In *Conference on Fairness, Accountability and Transparency*, pages 107–118. [Citado en pág. 7]

[Mukherjee et al., 2020] Mukherjee, D., Yurochkin, M., Banerjee, M., and Sun, Y. (2020). Two simple ways to learn individual fairness metrics from data. [Citado en pág. 39]

[O'neil, 2016] O'neil, C. (2016). *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books. [Citado en pág. 7]

[Pearl, 1999] Pearl, J. (1999). Probabilities of causation: three counterfactual interpretations and their identification. *Synthese*, 121(1):93–149. [Citado en pág. 44]

[Pleiss et al., 2017] Pleiss, G., Raghavan, M., Wu, F., Kleinberg, J., and Weinberger, K. Q. (2017). On fairness and calibration. [Citado en pág. 32]

[Purshouse and Fleming, 2007] Purshouse, R. C. and Fleming, P. J. (2007). On the evolutionary optimization of many conflicting objectives. *IEEE transactions on evolutionary computation*, 11(6):770–784. [Citado en pág. 61]

[Ranzato et al., 2021] Ranzato, F., Urban, C., and Zanella, M. (2021). Fairness-aware training of decision trees by abstract interpretation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 1508–1517, New York, NY, USA. Association for Computing Machinery. [Citado en págs. 11 and 81]

[Sani et al., 2018] Sani, H. M., Lei, C., and Neagu, D. (2018). Computational complexity analysis of decision tree algorithms. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 191–197. Springer. [Citado en pág. 20]

[Selbst et al., 2019] Selbst, A. D., Boyd, D., Friedler, S. A., Venkatasubramanian, S., and Vertesi, J. (2019). Fairness and abstraction in sociotechnical systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 59–68. [Citado en pág. 7]

[Strack et al., 2014] Strack, B., DeShazo, J. P., Gennings, C., Olmo, J. L., Ventura, S., Cios, K. J., and Clore, J. N. (2014). Impact of hba1c measurement on hospital readmission rates: Analysis of 70, 000 clinical database patient records. *BioMed Research International*, 2014:1–11. [Citado en pág. 99]

[Tang and Zhang, 2020] Tang, Z. and Zhang, K. (2020). Attainability and optimality: The equalized-odds fairness revisited. [Citado en págs. 28 and 29]

[Truong et al., 2024] Truong, V.-H., Tangaramvong, S., and Papazafeiropoulos, G. (2024). An efficient lightgbm-based differential evolution method for nonlinear inelastic truss optimization. *Expert Systems with Applications*, 237:121530. [Citado en pág. 89]

[Tsanas and Little, 2009] Tsanas, A. and Little, M. (2009). Parkinsons Telemonitoring. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5ZS3N. [Citado en pág. 100]

[Valdivia et al., 2020] Valdivia, A., Sánchez-Monedero, J., and Casillas, J. (2020). How fair can we go in machine learning? assessing the boundaries of fairness in decision trees. *arXiv preprint arXiv:2006.12399*. [Citado en págs. 7, 11, 102, and 125]

[van der Linden et al., 2022] van der Linden, J., de Weerdt, M., and Demirović, E. (2022). Fair and optimal decision trees: A dynamic programming approach. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 38899–38911. Curran Associates, Inc. [Citado en págs. 11 and 81]

[Van Veldhuizen, 1999] Van Veldhuizen, D. A. (1999). Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH SCHOOL OF ENGINEERING. [Citado en pág. 51]

[Vapnik, 1971] Vapnik, V. (1971). Chervonenkis: On the uniform convergence of relative frequencies of events to their probabilities. [Citado en pág. 15]

[Verma and Rubin, 2018] Verma, S. and Rubin, J. (2018). Fairness definitions explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, pages 1–7. IEEE. [Citado en págs. 7, 26, and 32]

[Xing et al., 2024] Xing, C., Liu, M., Peng, J., Wang, Y., Liu, Y., Gao, S., Zheng, Z., and Liao, J. (2024). Disturbance frequency trajectory prediction in power systems based on lightgbm spearman. *Electronics*, 13(3):597. [Citado en pág. 89]

[Zafar et al., 2017a] Zafar, M. B., Valera, I., Gomez Rodriguez, M., and Gummadi, K. P. (2017a). Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th international conference on world wide web*, pages 1171–1180. [Citado en págs. 7 and 29]

[Zafar et al., 2019] Zafar, M. B., Valera, I., Gomez-Rodriguez, M., and Gummadi, K. P. (2019). Fairness constraints: A flexible approach for fair classification. *J. Mach. Learn. Res.*, 20(75):1–42. [Citado en pág. 10]

[Zafar et al., 2017b] Zafar, M. B., Valera, I., Rogriguez, M. G., and Gummadi, K. P. (2017b). Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics*, pages 962–970. PMLR. [Citado en pág. 10]

[Zehlike et al., 2017] Zehlike, M., Bonchi, F., Castillo, C., Hajian, S., Megahed, M., and Baeza-Yates, R. (2017). Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1569–1578. [Citado en pág. 7]

[Zehlike and Castillo, 2020] Zehlike, M. and Castillo, C. (2020). Reducing disparate exposure in ranking: A learning to rank approach. In *Proceedings of The Web Conference 2020*, pages 2849–2855. [Citado en pág. 7]

[Zhang and Zhao, 2020] Zhang, W. and Zhao, L. (2020). Online decision trees with fairness. *ArXiv*, abs/2010.08146. [Citado en págs. 11 and 69]

[Zhuang et al., 2024] Zhuang, H., Lehner, F., and DeGaetano, A. T. (2024). Improved diagnosis of precipitation type with lightgbm machine learning. *Journal of Applied Meteorology and Climatology*, 63(3):437–453. [Citado en pág. 89]