



REPUBLIC OF BENIN
MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH OF BENIN

ABOMEY-CALAVI UNIVERSITY

**INSTITUTE OF TRAINING AND
RESEARCH IN COMPUTER SCIENCE**

BP 526 Cotonou Tel : +229 21 14 19 88
<http://www.ifri-uac.net> e-mail : contact@ifri.uac.bj



THESIS

for obtaining a

Master's degree in computer science

Option : Software Engineering

Presented by :

Wadoud O. A. Bienvenu MASSOU

Fairness in machine learning using pattern mining and constraint programming

Under supervision of : :

Dr (MA) Vinasétan Ratheil **HOUNDE**

Dr (MC) Arnaud **AHOUANDJINO**

Examination Comitee :

Dr (MC) Pelagie **HOUNGUE**

IFRI

President of the jury

Dr (MA) Amadou Tidjani **SANDA MAHAMA**

IFRI

Examiner

Dr (MA) Vinasétan Ratheil **HOUNDE**

IFRI

Reporter

Bon pour dépôt
libre
Pelagie HOUNGUE
(La présidente du jury)

Academic Year : 2023-2024

Summary

Dedication	ii
Acknowledgements	iii
Résumé	iv
Abstract	v
List of figures	vi
List of tables	viii
Introduction	1
1 State of the art and background	4
2 Approach and tools	21
3 Experimental results and discussion	36
Conclusion	54
Bibliography	55
Webography	56
Appendix	57
Contents	59

Dedication

To My Father, Mother and close family.

Acknowledgements

The successful completion of this work was made possible through the direct and indirect support of several individuals. I extend my heartfelt gratitude to everyone. In particular, I would like to sincerely thank:

- Prof. Eugène EZIN, Director of IFRI for his personal help and motivation boosts towards me, and his entire administration for granting us the opportunity to pursue our courses.
- Dr. Ratheil V. HOUNDEJI for his willingness to supervise me, his help and support throughout all the process.
- My father MASSOU Siaka and mother TCHIBOZO Brigitte for their consistent push and support.
- My close family, from which I got full help and support.
- My classmates for their help and efficiency.

Résumé

Les systèmes d'intelligence artificielle influencent de plus en plus les décisions critiques dans divers domaines, ce qui soulève des inquiétudes quant à l'équité et à la partialité de leurs résultats. La racine de ces problèmes se trouve souvent dans les données d'apprentissage, où les biais historiques et la sous-représentation de certains groupes peuvent conduire à des résultats discriminatoires. Cette thèse propose une nouvelle approche pour améliorer l'équité des ensembles de données en utilisant des techniques d'exploration de modèles couplées à des méthodes d'échantillonnage innovantes.

Nous présentons un cadre complet qui s'appuie sur l'exploration d'items fréquents pour découvrir les biais cachés dans les ensembles de données. Ces modèles découverts sont ensuite utilisés pour guider des techniques intelligentes de sous-échantillonnage et de suréchantillonnage, visant à équilibrer la représentation des différents groupes au sein des données.

La principale contribution de ce travail est une nouvelle bibliothèque qui intègre des algorithmes d'exploration de modèles avec des techniques d'échantillonnage personnalisées. Nous démontrons l'efficacité de notre méthode sur plusieurs ensembles de données du monde réel, montrant des améliorations significatives dans les mesures d'équité tout en maintenant l'utilité globale de l'ensemble de données. Des analyses comparatives avec les techniques existantes d'amélioration de l'équité mettent en évidence les avantages de notre approche basée sur les patterns.

Cette recherche contribue aux efforts en cours pour développer des systèmes d'IA responsables en fournissant aux scientifiques des données et aux praticiens de l'apprentissage automatique un outil puissant pour détecter et atténuer les biais à l'étape de la préparation des données. La méthode proposée offre une approche plus nuancée et contextuelle de l'équilibrage des ensembles de données, ce qui pourrait conduire à des modèles d'IA plus justes et plus équitables dans diverses applications.

Mots clés : L'équité dans l'IA, l'exploration de modèles, la partialité des ensembles de données, l'échantillonnage intelligent, l'IA responsable

Abstract

Artificial intelligence systems are increasingly influencing critical decisions in a variety of areas, raising concerns about the fairness and bias of their results. The root of these problems often lies in the training data, where historical biases and under-representation of certain groups can lead to discriminatory results. This thesis proposes a new approach to improving the fairness of datasets using model mining techniques coupled with innovative sampling methods.

We present a comprehensive framework that relies on frequent item mining to discover hidden biases in datasets. These discovered patterns are then used to guide intelligent subsampling and oversampling techniques, aimed at balancing the representation of different groups within the data.

The main contribution of this work is a new library that integrates pattern mining algorithms with custom sampling techniques. We demonstrate the effectiveness of our method on several real-world datasets, showing significant improvements in fairness measures while maintaining the overall utility of the dataset. Comparative analyses with existing fairness improvement techniques highlight the advantages of our pattern-based approach.

This research contributes to ongoing efforts to develop responsible AI systems by providing data scientists and machine learning practitioners with a powerful tool to detect and mitigate biases at the data preparation stage. The proposed method offers a more nuanced and contextual approach to balancing datasets, which could lead to fairer and more equitable AI models in various applications.

Keywords: fairness in AI, pattern mining, dataset bias, intelligent sampling, responsible AI

List of figures

1.1	Data Split [2]	5
1.2	Supervised learning[3]	6
1.3	Unsupervised Learning [1]	7
1.4	Reinforcement Learning[11]	7
1.5	Overgeneralization Bias [7]	9
2.1	FairnessOptimizer Web Application user process	32
2.2	FairnessOptimizer Sequence Diagram	33
2.3	FairnessOptimizer Use Case Diagram	34
2.4	FairnessOptimizer Class Diagram	35
3.1	The study dataset	36
3.2	"Region" Feature initial Distribution	37
3.3	"PreviousCompanies" Feature initial Distribution	37
3.4	"Religion" Feature initial Distribution	38
3.5	2nd Example Dataset	38
3.6	'Gender" Feature Initial Distribution	39
3.7	"Region" Feature Random split Distribution	39
3.8	"Religion" Feature Random split Distribution	40
3.9	"Previous Companies" Feature Random split Distribution	40
3.10	"Region" Feature FairnessOptimizer split Distribution	41
3.11	"Religion" Feature FairnessOptimizer split Distribution	42
3.12	"Previouscompany" Feature FairnessOptimizer split Distribution	43
3.13	"Region" Feature FairMaker split Distribution	43
3.14	"Religion" Feature FairMaker split Distribution	44
3.15	"Previouscompany" Feature FairMaker split Distribution	44
3.16	"Gender" Feature Fairmakersplit Distribution Dataset 2	45
3.17	"Gender" Feature FairnessOptimizer split Distribution Dataset 2	45
3.18	Oversampled GAN Approach dataset shape	46
3.19	Oversampled FairnessOptimizer Approach dataset shape	46
3.20	"Previouscompany" Feature Random Oversample Distribution	47
3.21	"Region" Feature Random Oversample Distribution	47
3.22	"Religion" Feature Random Oversample Distribution	48
3.23	"Previouscompany" Feature Fairness Optimizer Oversample Distribution	48
3.24	"Religion" Feature FairnessOptimizer Oversample Distribution	49
3.25	"Region" Feature FairnessOptimizer Oversample Distribution	50
3.26	Oversampling with ADASYN comparison on different features	51

3.27 "FairnessOptimizer Web Application page Step 1	51
3.28 "FairnessOptimizer Web Application page Step 2	52
3.29 "FairnessOptimizer Web Application page	52
3.30 "FairnessOptimizer Web Application page	53

List of tables

1.1	Summary of Pattern Mining Techniques for Bias Detection	17
1.2	Summary of Sampling Methods for Dataset Balancing	18
1.3	Summary of Existing Libraries and Tools for Fairness in AI	19
2.1	Demographics categories	25
2.2	Dataset distribution by gender and ethnicity	25
2.3	Initial Dataset	29
2.4	Resulting Dataset after Undersampling	30

Introduction

Context

In the rapidly evolving landscape of data-driven decision-making, the imperative for fairness in machine learning models has become increasingly pronounced. As algorithmic systems permeate various facets of society, from healthcare to finance, the potential for perpetuating or exacerbating existing biases has emerged as a critical concern. This thesis presents a novel approach to addressing this challenge through the development of a Fairness Optimizer system, which leverages pattern mining techniques to enhance dataset fairness with respect to sensitive attributes.

The ubiquity of machine learning applications in contemporary society underscores the urgency of ensuring equitable outcomes across diverse demographic groups. However, the intrinsic biases present in historical data often lead to skewed model predictions, potentially reinforcing societal inequities. This research aims to mitigate these issues by introducing a comprehensive framework for dataset optimization, with a particular focus on maintaining utility while improving fairness metrics.

Problem statement

Central to the approach is the innovative application of pattern mining methodologies. By extracting meaningful patterns from the original dataset, we gain crucial insights into the underlying structure and relationships within the data. This pattern-centric perspective forms the foundation of the constraint generation process, enabling a more nuanced and context-aware approach to fairness optimization.

Objectives

The main objective of this work is to contribute to the broader discourse on ethical AI and responsible data science practices through the tool provided. The specific objectives of this study are as follows:

1. To develop a robust Fairness Optimizer system that effectively improves dataset fairness while preserving data utility.
2. To explore the efficiency of pattern mining techniques in identifying and addressing fairness-related issues within datasets.
3. To provide a comprehensive toolkit for data scientists and machine learning practitioners to evaluate and enhance the fairness of their datasets.

Contributions

This research makes several key contributions to the field:

1. A novel fairness optimization framework that integrates pattern mining techniques with constraint solving methodologies.
2. An adaptive approach to data adjustment, incorporating both undersampling and oversampling strategies based on optimal coefficients.
3. A comprehensive evaluation methodology that assesses fairness improvements and pattern preservation in optimized datasets.
4. An accessible web interface that facilitates the practical application of the fairness optimization techniques.

Thesis structure

This thesis is structured into three main chapters:

1. **State of the art and background:** This chapter provides a comprehensive review of existing fairness metrics, optimization techniques, and pattern mining methodologies in the context of machine learning fairness.
2. **Tools and approach:** Here, we detail the tools used to build the fairness optimizer system, then present the implementation details, demonstrating the practical realization of the theoretical approach.
3. **Experimental results and discussion:** The last chapter presents the experimental findings, demonstrating the effectiveness of the approach across various datasets and fairness scenarios. We also discuss the implications of the research, its limitations, and potential avenues for future investigation in the realm of fair machine learning.

By addressing the critical issue of dataset fairness through the lens of pattern mining and optimization, this research aims to contribute significantly to the development of more equitable and responsible AI systems. The fairness optimizer not only provides a practical tool for improving dataset fairness but also opens new avenues for exploring the intersection of pattern recognition and ethical machine learning practices.

State of the art and background

Introduction

In this chapter, we will go through the literature regarding AI and fairness.

1.1 Artificial intelligence

Artificial Intelligence (AI) is the study of how to build or program computers to enable them to do what human minds can do. It refers to the capability of computer systems to perform tasks that typically require human intelligence. This includes functions such as reasoning, problem-solving, understanding natural language, and recognizing patterns. AI encompasses a variety of technologies and can be categorized into two main types: Weak AI (or Narrow AI), which is designed to perform specific tasks, and Strong AI (or Artificial General Intelligence), which would possess human-like cognitive abilities across a wide range of activities.

AI has become increasingly prevalent in our daily lives, influencing various sectors such as healthcare, finance, transportation, and entertainment. Some common applications of AI include:

- **Natural Language Processing (NLP):** It is a scientific discipline which is found at the interface of computer science, artificial intelligence and cognitive psychology [9]. It enables machines to understand, interpret, and generate human language. Applications include chatbots, language translation, and sentiment analysis.
- **Computer Vision:** Allows machines to interpret and understand visual information from the world. This technology is used in facial recognition systems, autonomous vehicles, and medical image analysis. AI and computer vision technologies are paramount to prevent exacerbating existing disparities and inequalities [13].
- **Expert systems:** An expert system, also known as a knowledge based system, is a computer program that contains some of the subject-specific knowledge of one or more human experts. [12]. These AI systems emulate the decision-making ability of human experts in specific domains, such as medical diagnosis or financial planning.

The rapid advancement of AI technologies has also raised important ethical and societal questions, including concerns about job displacement, privacy, and the potential for AI systems to perpetuate or amplify existing biases. As AI continues to evolve, addressing these challenges will be crucial for ensuring its responsible and beneficial development.

1.2 What is Machine Learning, and how does it work ?

Machine learning (ML) is a subset of artificial intelligence (AI) focused on the development of algorithms that enable computers to learn from and make predictions based on data. Unlike traditional programming, where explicit instructions are provided, machine learning allows systems to improve their performance autonomously as they are exposed to more data.

ML operates through several key processes:

1.2.1 Data input

ML algorithms require large datasets to learn from. The quality and quantity of this data significantly affect the model's performance. The data treatment goes through multiple steps :

- **Data collection:** Gather data from various sources like databases, APIs, spreadsheets, or web scraping and ensure the collected data is relevant to the problem being solved
- **Data cleaning:** Remove or correct missing values, outliers, or inconsistencies in the data
- **Data transformation:** Transform the cleaned data into a format compatible with machine learning algorithms
- **Data splitting:** Split the prepared data into training, validation, and test sets. The next figure illustrates this splitting setup.

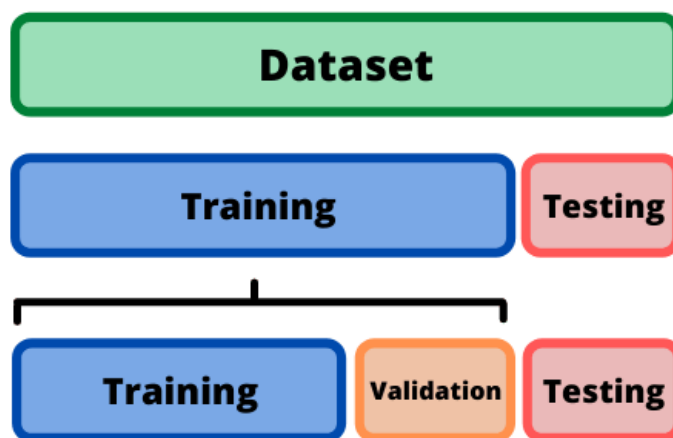


Figure 1.1: Data Split [2]

The figure 1.1 depicts an image illustrating the data split process in machine learning, showing how source data is divided into training (70 percent), validation (15 percent), and testing (15 percent) sets to ensure model performance evaluation.

1.2.2 Algorithm selection

Various algorithms are employed depending on the type of task at hand. The primary categories include:

Supervised learning: Involves training a model on a labeled dataset, where the input data is paired with the correct output. This method is commonly used for tasks such as classification (predicting categories) and regression (predicting numerical values). The next figure explains the supervised learning principle through an example.

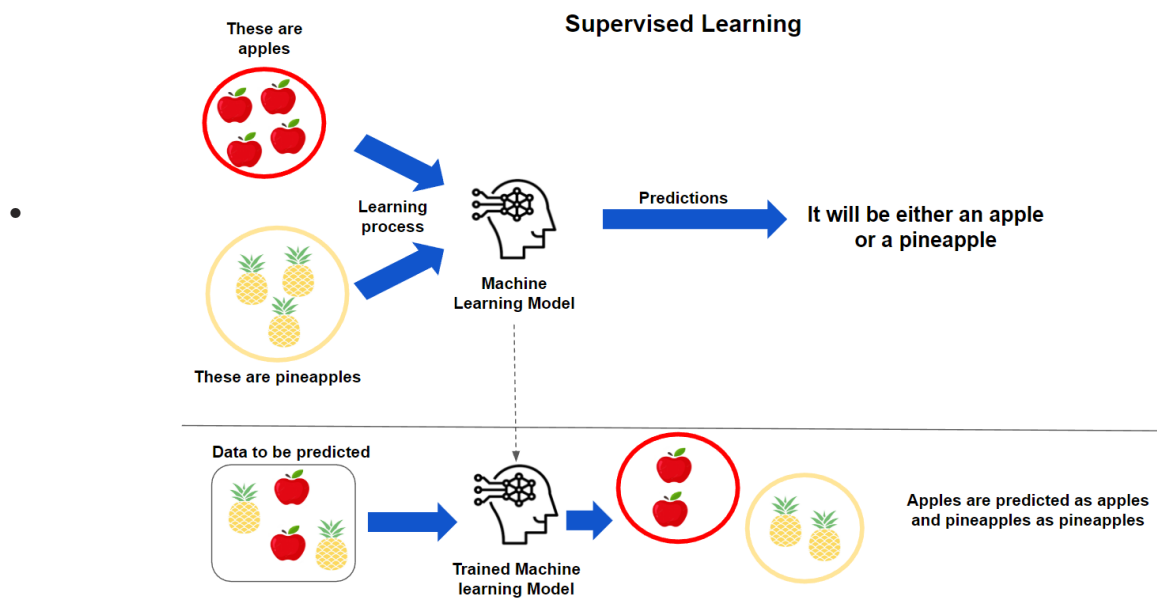


Figure 1.2: Supervised learning[3]

The figure illustrates the concept of supervised learning, where a machine learning model is trained to differentiate between apples and pineapples using labeled data, and then makes predictions on new data based on this learned distinction.

- **Unsupervised learning:** The model analyzes input data without labeled responses, identifying patterns and structures within the data. This approach is useful for clustering and association tasks. The next figure explains the unsupervised learning principle through an example.

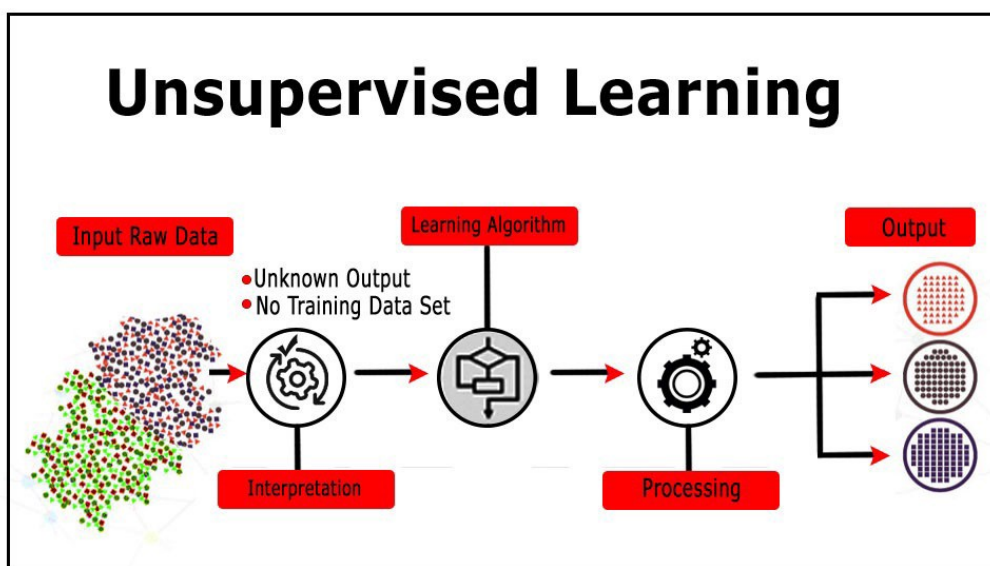


Figure 1.3: Unsupervised Learning [1]

The figure 1.3 represents unsupervised learning, where a learning algorithm processes raw, unlabeled data, interprets patterns without prior training data, and organizes the data into meaningful clusters or structures based on similarities.

- **Reinforcement learning:** An agent learns to make decisions by receiving rewards or penalties based on its actions, refining its strategy over time. The next figure illustrates reinforcement learning.

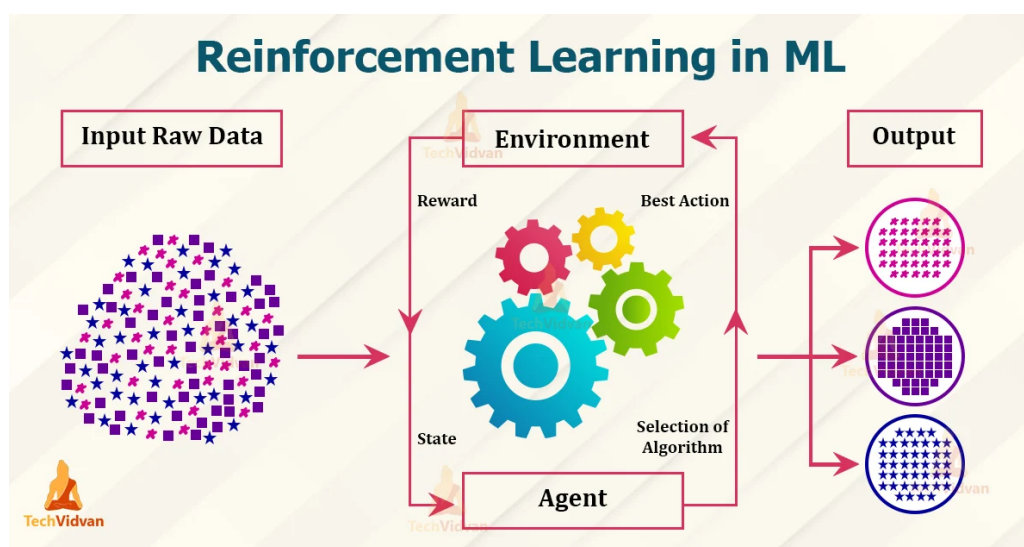


Figure 1.4: Reinforcement Learning[11]

The figure 1.4 visually represents how reinforcement learning transforms raw data through interaction with an environment to produce learned outputs.

1.2.3 Model training

During training, the algorithm adjusts its parameters to minimize errors in predictions. This involves iterating over the dataset multiple times, allowing the model to learn complex relationships within

the data.

1.2.4 Evaluation and testing

After training, the model is tested on a separate dataset to evaluate its performance. Metrics such as accuracy, precision, and recall are commonly used to assess how well the model generalizes to unseen data.

1.2.5 Deployment and improvement

Once validated, the model can be deployed in real-world applications. Continuous learning from new data helps improve its accuracy and reliability over time.

1.3 Dataset bias and fairness in AI

Artificial intelligence (AI) and machine learning (ML) systems are increasingly influencing crucial decisions in our daily lives, from loan approvals to medical diagnoses. However, these systems are only as good as the data they are trained on. When training datasets contain biases, AI models can perpetuate and even amplify these biases, leading to unfair outcomes for certain groups. Data bias refers to systematic errors or distortions in datasets that can lead to inaccurate conclusions or unfair outcomes, particularly in machine learning and data analysis. This can manifest in various forms[8]:

1.3.1 Types of biases

- **Selection bias** : This occurs when the data collected is not representative of the population intended to be analyzed. It can arise from improper randomization during data collection, leading to skewed results. Types include:
 - **Sampling bias**: When the sample is not randomly selected.
 - **Participation bias**: When certain groups are underrepresented due to voluntary participation.
 - **Convergence bias**: When the sample is collected from a non-representative subset of the population.
- **Reporting bias**: This happens when only a subset of data is reported, often skewing results. Types include:
 - **Publication bias**: Positive results are more likely to be published than negative ones.
 - **Outcome reporting bias**: Selectively reporting certain outcomes while ignoring others.
- **Confirmation bias**: This bias occurs when researchers favor data that supports their existing beliefs or hypotheses, leading to selective analysis and interpretation. [8].
- **Measurement bias**: When the features or labels used do not accurately capture the phenomenon of interest for all groups.
- **Automation bias**: People may overly trust automated systems' outputs without verifying their accuracy, which can perpetuate errors present in the underlying data.

- **Overgeneralization bias** The next figure illustrates an instance of overgeneralization bias.

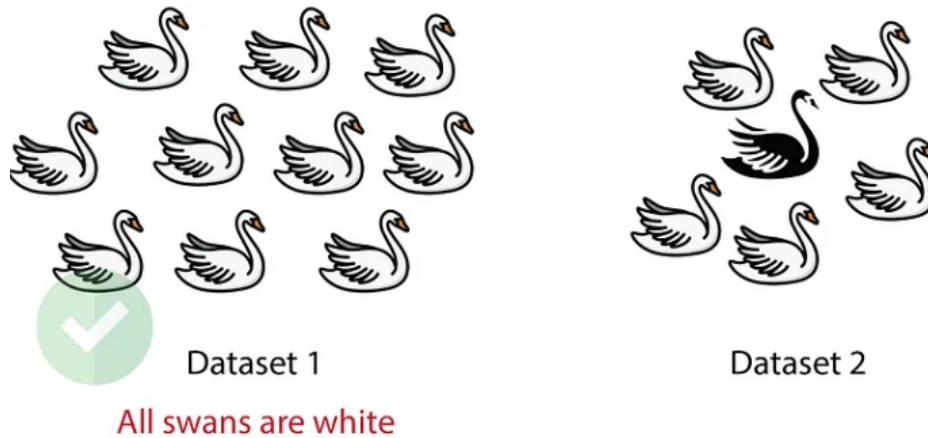


Figure 1.5: Overgeneralization Bias [7]

Overgeneralization occurs when you assume what you see in your dataset is what you would see if you looked in any other dataset meant to assess the same information, regardless of the size of the dataset.

- **Historical bias:** Existing societal biases are reflected in historical data, potentially reinforcing stereotypes and discrimination when used in predictive models.
- **Representation bias:** This occurs when certain groups are over- or under-represented in the dataset, leading to models that do not generalize well across diverse populations.
- **Implicit bias:** Assumptions made based on personal experiences or societal stereotypes can influence how data is interpreted or collected.

1.3.2 Existing fairness metrics

Fairness in AI aims to mitigate these biases and ensure that AI systems do not discriminate against individuals or groups based on sensitive attributes such as race, gender, or age. Several bias measurements metrics have been proposed in the literature. [5] Amongst the most popular ones, we can count :

- **Demographic parity:** This metric requires that the proportion of positive outcomes is the same across different demographic groups. It assesses whether the model's predictions are independent of sensitive attributes, such as race or gender. A common formulation is that the probability of a positive prediction should be equal across groups.
- **Equal opportunity:** Equal opportunity focuses on ensuring that the true positive rate (TPR) is equal across groups. This means that if individuals are qualified, they should have an equal chance of being positively classified, regardless of their group membership.
- **Equalized odds:** This metric extends equal opportunity by requiring both the true positive rate and the false positive rate (FPR) to be equal across groups. It aims to ensure that similar individuals receive similar treatment, regardless of their group affiliation.

- **Predictive parity:** Predictive parity ensures that the positive predictive value (PPV) is equal across groups. This means that among those predicted as positive, the proportion who are actually positive should be the same for all groups.

Addressing dataset bias and ensuring fairness is crucial for building trustworthy and ethical AI systems. This has led to increased research in fairness-aware machine learning and the development of various techniques to detect and mitigate bias in datasets.

1.4 Pattern mining techniques

Pattern mining is a crucial data analysis technique aimed at discovering interesting, unexpected, and valuable regularities in large datasets. In the context of fairness and bias detection, pattern mining can be a powerful tool to uncover hidden biases and understand the underlying structure of the data. This section will delve deeper into various pattern mining techniques, their applications, and their relevance to fairness in AI.

1.4.1 Frequent itemset mining

Frequent itemset mining aims to find sets of items that frequently co-occur in a dataset. This technique is fundamental to many pattern mining tasks and has applications in market basket analysis, web usage mining, and bioinformatics.

1.4.1.1 Apriori algorithm

The Apriori algorithm is one of the earliest and most well-known algorithms for frequent itemset mining. It uses a bottom-up approach, extending frequent itemsets one item at a time. The algorithm leverages the anti-monotonicity property: if an itemset is infrequent, all its supersets must also be infrequent.

Key steps of the Apriori algorithm:

1. Generate candidate itemsets of length $k+1$ from frequent itemsets of length k .
2. Prune candidate itemsets that contain infrequent subsets.
3. Scan the database to determine the support of each candidate itemset.
4. Eliminate candidates that do not meet the minimum support threshold.

While Apriori is intuitive and easy to implement, it can be computationally expensive for large datasets due to multiple database scans and the generation of many candidate itemsets.

1.4.1.2 FP-Growth algorithm

The FP-Growth (Frequent Pattern Growth) algorithm addresses some of the limitations of Apriori by using a divide-and-conquer strategy and a special data structure called the FP-tree (Frequent Pattern tree).

Key features of FP-Growth:

- Constructs a compact data structure (FP-tree) that stores quantitative information about frequent patterns.
- Avoids the need for candidate generation by using a pattern fragment growth method.
- Reduces the number of database scans to just two: one for counting item frequencies and another for constructing the FP-tree.

FP-Growth is generally more efficient than Apriori, especially for large datasets with many frequent patterns. However, it can be memory-intensive for very large datasets that don't fit into main memory.

1.4.2 Sequential pattern mining

Sequential pattern mining extends itemset mining by considering the order of items. This technique is particularly useful in scenarios where the temporal aspect of data is important, such as analyzing customer purchase sequences or studying disease progression patterns.

1.4.2.1 PrefixSpan algorithm

The PrefixSpan (Prefix-projected Sequential Pattern mining) algorithm is an efficient method for mining sequential patterns. It uses a pattern-growth approach, recursively projecting the sequence database into smaller databases and growing subsequences.

Key features of PrefixSpan:

- Projects the sequence database based on frequent prefixes to reduce search space.
- Avoids candidate generation and testing.
- Can be more scalable than alternative algorithms like GSP (Generalized Sequential Patterns).

1.4.2.2 SPADE algorithm

The SPADE (Sequential Pattern Discovery using Equivalence classes) algorithm is another approach to sequential pattern mining that uses a vertical data format and lattice-theoretic approach.

Key features of SPADE:

- Uses a vertical id-list database format, which allows for efficient support counting.
- Decomposes the original problem into smaller sub-problems that can be independently solved in main memory.
- Can be easily parallelized for improved performance on multicore systems.

1.4.3 Association rule mining

Association rule mining aims to discover interesting relations between variables in large databases. It's often used in market basket analysis but can be adapted for bias detection and fairness analysis in AI systems.

1.4.3.1 Apriori-based algorithms

These algorithms first find frequent itemsets using the Apriori algorithm and then generate rules from these itemsets. The process involves two main steps:

1. Find all frequent itemsets: Sets of items that appear together in a database with frequency no less than a predetermined minimum support threshold.
2. Generate strong association rules from the frequent itemsets: Rules that satisfy both minimum support and minimum confidence thresholds.

1.4.3.2 FP-Growth-based algorithms

Association rule mining can also be performed using the FP-Growth algorithm as a base. These approaches generally follow a similar two-step process but use the FP-tree structure for more efficient frequent itemset discovery.

1.4.4 Emerging pattern mining

Emerging patterns are itemsets whose support increases significantly from one dataset to another. This technique can be particularly useful for identifying discriminative features between different groups or classes, which is highly relevant to fairness and bias detection in AI. [10]

Key concepts in emerging pattern mining:

- **Growth rate:** The ratio of the support of an itemset in one dataset to its support in another dataset.
- **Jumping emerging patterns:** Patterns with an infinite growth rate (i.e., they appear in one dataset but not in the other).
- **Chi-Square-based emerging patterns:** Patterns selected based on the chi-square test, which measures the deviation from the expected distribution.

1.4.5 Applying pattern mining to fairness and bias detection

Pattern mining techniques can be powerful tools for uncovering hidden biases and ensuring fairness in AI systems. Some potential applications include:

- **Identifying discriminative patterns:** Use emerging pattern mining to discover features or combinations of features that disproportionately affect certain protected groups.
- **Analyzing decision paths:** Apply sequential pattern mining to study the sequence of decisions made by AI systems and identify potentially biased decision paths.
- **Discovering correlations:** Use association rule mining to uncover unexpected correlations between protected attributes and decision outcomes.
- **Temporal bias analysis:** Employ sequential pattern mining to study how biases in AI systems may evolve over time.
- **Feature interaction analysis:** Utilize frequent itemset mining to identify complex feature interactions that may contribute to unfair outcomes.

Using these pattern mining techniques, researchers and practitioners can gain deeper insights into the underlying structures and relationships within datasets used to train AI systems. This knowledge can then be used to develop more effective bias mitigation strategies and fairness-aware machine learning algorithms.

1.5 Sampling methods for dataset balancing

Dataset balancing is a crucial step in addressing bias and improving fairness in machine learning datasets. Various sampling methods have been developed to adjust the distribution of classes or groups within a dataset. These methods generally fall into two categories: undersampling and oversampling.

1.5.1 Undersampling techniques

Undersampling involves reducing the number of samples in the majority class to balance it with the minority class. Some key undersampling techniques include:

- **Random undersampling:** Randomly removes samples from the majority class. While simple, this method risks losing important information.
- **Tomek links:** Tomek-link defines the concept boundary pairs in a data distribution. When two instances, a and b are nearest of each-other, with class (a) class (b); then the pair (a,b) is termed as Tomek-link pair. These pairs are noise-promoting in the data distribution. Tomek links remove unwanted overlap between classes where majority class links are removed until all minimally distanced nearest neighbor pairs are of the same class. A Tomek link is defined as follows: given an instance pair (x_i, x_j) , where $x_i \in S_{\min}, x_j \in S_{\max}$ and $d(x_i, x_j)$ is the distance between x_i and x_j , then the pair (x_i, x_j) is called a Tomek link if there's no instance x_k such that $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j, x_k) < d(x_i, x_j)$. In this way, if two instances form a Tomek link then either one of these instances is noise or both are near a border. Thus, one can use Tomek links to clean up overlap between classes. By removing overlapping examples, one can establish well-defined clusters in the training set and lead to improved classification performance.
- **Cluster centroids:** Applies K-means clustering to the majority class and uses the cluster centroids as the new samples.
- **Near miss:** Selects majority class samples based on their distance to minority class samples. Several variants exist, each with different selection criteria .

1.5.2 Oversampling techniques

Oversampling increases the number of samples in the minority class to balance it with the majority class:

- **Random oversampling:** Randomly duplicates samples from the minority class. This can lead to overfitting. Amongst those techniques, the most popular ones are Generative Adversarial Networks (GANs) which are a class of deep learning architectures that have gained prominence for their ability to generate realistic synthetic data. GANs consist of two neural networks—the

generator and the discriminator—that compete against each other in a game-theoretic framework.

- **SMOTE (Synthetic Minority Over-sampling Technique):** Creates synthetic samples by interpolating between existing minority samples.
- **ADASYN (Adaptive Synthetic):** The essential idea of ADASYN is to use a weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data is generated for minority class examples that are harder to learn compared to those minority examples that are easier to learn. As a result, the ADASYN approach improves learning with respect to the data distributions in two ways: (1) reducing the bias introduced by the class imbalance, and (2) adaptively shifting the classification decision boundary toward the difficult examples [6].

1.5.3 Hybrid Methods

Some techniques combine undersampling and oversampling:

- **SMOTETomek:** Applies SMOTE oversampling followed by Tomek links undersampling.
- **SMOTEENN:** Combines SMOTE with Edited Nearest Neighbors (ENN) cleaning . GANs (Generative Adversarial Networks)

While these methods can be effective in balancing datasets, they often do not explicitly consider the complex patterns and relationships within the data that may be crucial for maintaining fairness across different groups.

1.6 Existing libraries and their limitations

Several libraries and tools have been developed to address fairness and bias in AI systems. While these tools provide valuable functionality, they also have limitations that our proposed approach aims to address.

1.6.1 AI Fairness 360 (IBM)

The AI Fairness 360 toolkit is an extensible open-source library developed by IBM. It contains a set of metrics to check bias in datasets and ML models, as well as algorithms to mitigate such bias.

Key features:

- Nine different algorithms for bias mitigation
- Interactive online demo with sample datasets
- Guidance on choosing appropriate bias mitigation algorithms
- Comparison of model accuracy before and after mitigation

The toolkit provides more metrics and algorithms than are available in the web demo, allowing developers to address unwanted bias in their models more comprehensively.

1.6.2 Fairlearn (Microsoft)

Fairlearn is a Python package developed by Microsoft to help developers and data scientists assess and mitigate fairness issues in AI systems [4].

Key components:

- Metrics for evaluating the impact of models on different groups
- Algorithms for mitigating unfairness across various AI tasks and fairness definitions
- Focus on fairness in terms of the system's impact on people

1.6.3 FairML

FairML is an end-to-end toolbox for auditing predictive models by quantifying the relative significance of the model's inputs. It uses model compression and four input ranking algorithms to measure a model's relative predictive dependence on its inputs.

1.6.4 FairMaker

Fairmaker is a Python library designed to assist in creating fairer datasets for machine learning applications. It aims to address issues of bias and imbalance in datasets, particularly concerning sensitive features like gender. The library provides tools for generating test datasets that are less biased compared to those created using random sampling methods, which often maintain or even exacerbate existing imbalances.

The library employs constraint programming techniques to compute a test set that is as fair as possible, given the constraints of the original dataset. Additionally, Fairmaker includes a demo web application that allows users to interact with the library's features without needing to write code. This application enables users to visualize dataset specifications, download datasets in CSV format, and adjust parameters to achieve a fairer test set.

1.6.5 What-If Tool (Google)

While not specifically designed for fairness in ML models, the What-If Tool helps non-technical stakeholders understand model performance on a dataset. It allows for interactive exploration of datasets and models, making it useful for data scientists and researchers to learn about machine learning and its implications.

1.6.6 Limitations of existing tools

Despite the valuable contributions of these libraries, several limitations remain:

1. **Lack of pattern-based approaches:** Most existing libraries do not leverage pattern mining techniques to uncover complex relationships in the data that may contribute to bias.
2. **Focus on predefined fairness metrics:** Many approaches optimize for specific fairness metrics, which may not capture all aspects of fairness relevant to a given context.
3. **Lack of data generation:** Some libraries don't offer the possibility to generate data while maintaining some form of fairness, which can be needed by some data scientists in their work.

4. **Limited consideration of data semantics:** Current sampling methods often treat data points as independent entities, ignoring important semantic relationships that could be crucial for maintaining dataset utility while improving fairness.
5. **Scalability issues:** Some sophisticated fairness improvement techniques can be computationally expensive, limiting their applicability to large-scale datasets.
6. **Lack of interpretability:** Many existing bias mitigation techniques operate as "black boxes," making it difficult for users to understand and trust the transformations applied to their data.
7. **Narrow focus:** Some tools and research efforts focus on specific types of bias, such as gender or race-based bias, which may not address the full spectrum of fairness concerns in AI systems.

Our proposed approach aims to address these limitations by introducing a pattern-mining-based framework for dataset balancing. By leveraging frequent itemsets, sequential patterns, and association rules, we can develop more nuanced and context-aware sampling techniques. This approach not only aims to improve fairness but also to maintain the semantic integrity of the dataset, provide better interpretability, and scale effectively to large datasets.

1.6.7 Summary of existing tools and methods

Let's now summarize the existing solutions and methods, their pros and their cons in the following tables.

Method	Description	Pros	Cons
Apriori Algorithm	Bottom-up approach for finding frequent itemsets, extending one item at a time	• Intuitive and easy to implement • Complete pattern discovery • Well-documented	• Multiple database scans • High computational cost • Memory-intensive
FP-Growth	Uses divide-and-conquer strategy with FP-tree structure	• More efficient than Apriori • Only two database scans • No candidate generation	• Complex implementation • Memory-intensive for large datasets • FP-tree construction overhead
PrefixSpan	Pattern-growth approach for sequential pattern mining	• Efficient projection • No candidate generation • Scalable	• Complex for long sequences • Memory constraints • Projection cost
SPADE	Vertical data format approach using equivalence classes	• Efficient support counting • Easy parallelization • Memory efficient	• Complex implementation • Initial conversion overhead • Storage requirements

Table 1.1: Summary of Pattern Mining Techniques for Bias Detection

Method	Description	Pros	Cons
Random Under-sampling	Randomly removes majority class samples	• Simple implementation • Fast execution • Reduces processing time	• Information loss • Potentially removes useful samples • May discard important patterns
Tomek Links	Removes overlapping instances between classes	• Improves class separation • Maintains data structure • Reduces noise	• Computationally expensive • Slow for large datasets • May not fully balance classes
SMOTE	Creates synthetic minority samples through interpolation	• Prevents overfitting • Preserves relationships • Better than simple duplication	• May create unrealistic samples • Sensitive to outliers • Increases class overlap
ADASYN	Generates more synthetic samples for harder-to-learn cases	• Adaptive learning • Focus on difficult cases • Reduces classification bias	• Computationally intensive • Parameter sensitive • Can be unstable

Table 1.2: Summary of Sampling Methods for Dataset Balancing

Tool	Description	Pros	Cons
AI Fairness 360	IBM's comprehensive toolkit for bias detection and mitigation	• Multiple algorithms • Interactive demos • Well-documented	• Steep learning curve • Limited pattern mining • Complex for non-experts
Fairlearn	Microsoft's Python package for fairness assessment	• Human impact focus • Easy integration • Good evaluation metrics	• Limited to predefined metrics • No data generation • Limited semantic consideration
FairML	Toolbox for auditing predictive models	• Input significance ranking • Model compression • Multiple algorithms	• Limited scope • Focus only on input ranking • Less comprehensive metrics
FairMaker	Tool for creating fairer datasets	• Web interface • Constraint programming • Dataset generation	• Limited scalability • Limited customization • Focused on specific biases
What-If Tool	Google's interactive tool for model understanding	• Interactive exploration • Non-technical friendly • Visual analysis	• Not fairness-specific • Limited advanced features • Dataset size limitations

Table 1.3: Summary of Existing Libraries and Tools for Fairness in AI

1.7 Conclusion

Pattern mining techniques offer powerful tools for uncovering hidden patterns, relationships, and biases in large datasets. By applying these techniques to the problem of fairness in AI, we can develop more sophisticated methods for detecting and mitigating bias. Future research should focus on integrating pattern mining approaches with existing fairness frameworks and developing scalable, interpretable solutions that can be applied to real-world AI systems.

Approach and tools

Introduction

In this chapter, we will first describe the tools we used to implement the solution, then explain in details how it works.

2.1 Overview

The system we have designed to tackle the fairness challenges identified in machine learning datasets consists of two core components:

- **Python package:** This package provides the necessary tools to evaluate fairness in machine learning datasets and generate a more balanced test dataset.
- **Demo web application :** The purpose of the demo website is to offer data scientists access to some of the package's functionalities.

2.2 Hardware

To develop the Python library, we utilized a laptop with the following technical specifications:

- **Processor:** Intel Core i5
- **RAM:** 16 GB
- **Storage:** 500 GB SSD
- **Operating System:** Windows 11

In contrast, the demo web application can be accessed easily using a typical laptop, as most processing occurs on the server side.

2.3 Software

The majority of tools used in developing the toolkit are software, as the project centers around creating a Python package for data scientists to utilize in their machine learning projects. The main tools include:

- **Python 3:** Python is a versatile, high-level programming language, known for being dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Often described as a "batteries-included" language due to its vast standard library, Python 3, released in December 2008, addressed many of the limitations found in Python 2. Python was chosen for this project due to its extensive data science and machine learning libraries, ease of use, and strong community support, making it ideal for implementing complex fairness optimization algorithms and data analysis tasks.

Python's vast ecosystem of libraries makes it a go-to language for data science, capable of handling tasks like data preprocessing, visualization, statistical analysis, and the deployment of machine learning and deep learning models. Key libraries often used in data science and ML include:

- **NumPy**
- **Pandas**
- **Scikit-learn**

In this project, Python was utilized to develop the framework that evaluates machine learning datasets and generates a more equitable test set. Additionally, the demo web application was built using Python.

- **Constraint programming** Constraint programming is a declarative programming paradigm used primarily for solving combinatorial problems. It focuses on defining a problem in terms of constraints that specify the relationships between variables, rather than detailing the steps to solve it. Constraint programming was selected because it provides a powerful framework for expressing and solving combinatorial problems, allowing for efficient optimization of fairness constraints in large datasets while maintaining flexibility in defining and adjusting fairness criteria. Here's a brief overview of its key components
 - **Constraints:** These are conditions or restrictions on the values that variables can take. For example, in a scheduling problem, a constraint might specify that two tasks cannot occur at the same time. Those constraints can be :
 - * **Logical:** These constraints allow for the expression of complex relationships between variables that might be impractical or impossible to represent with arithmetic constraints alone.
 - * **Arithmetic:** Defined using standard operators such as $>$, $<$, $=$, etc.
 - * **Extensional:** Enumerated constraints, where a specific set of values satisfies the condition.

- **Variables and domains** : Each variable in a constraint programming model has a domain, which is the set of possible values it can take. The goal is to find values for all variables that satisfy all constraints.
- **Constraint Satisfaction Problems (CSP)** : A CSP involves finding solutions that satisfy a predefined set of constraints. The goal is to find a set of values for variables that meet all imposed conditions. Formally, a CSP can be represented as a triplet $\langle X, D, C \rangle$, where:
 - * $X = X_0, X_1, \dots, X_n$ represents the set of variables.
 - * $D = D_0, D_1, \dots, D_n$ refers to the domains of those variables.
 - * $C = C_0, C_1, \dots, C_n$ is the set of constraints that must be met.
- **Constraint Optimization Problem (COP)**: A COP is essentially a CSP with an added objective function to optimize—either by minimizing or maximizing a particular solution. Common solution techniques include backtracking, constraint propagation, and local search. Depending on the problem, a user may seek:
 - * A solution that satisfies all constraints.
 - * The optimal solution with respect to the objective.
 - * Proof that the best-found solution is indeed optimal.
 - * Proof that no solution exists for the problem.

The core functionality of the developed library leverages constraint programming techniques to generate a more balanced and fair dataset from the original one.

- **FP-Growth** FP-Growth, or Frequent Pattern Growth, is a data mining algorithm used to discover frequent patterns and associations in large datasets. It is particularly effective for market basket analysis and other applications where identifying common itemsets is crucial.
- **Google OR-TOOLS** Google OR-TOOLS is a free and open-source optimization software developed by Google, designed to solve problems in linear programming, constraint programming, and other related optimization domains. OR-TOOLS was extensively used in this project, particularly in calculating the fair test set. Using the integrated solver, CPSAT, we were able to model and solve the constraint programming problem, which resulted in the generation of the balanced test set. Much of the library's functionality relies on OR-TOOLS. OR-Tools was employed for constraint programming and optimization tasks, offering a powerful framework to model and solve complex fairness constraints efficiently.
- **Flask** Flask is a minimalistic web framework for Python that does not depend on external libraries. In this project, Flask was used to build the demo web application that provides access to certain features of the package. Flask was chosen as the web framework due to its lightweight nature, simplicity, and flexibility, making it ideal for creating a user-friendly interface to interact with the fairness optimization system.

Implementation details

Let's break down the core of this approach.

This library implements the FP-growth algorithm for frequent item set mining. Here's a brief overview of how it works:

- We first prepare the data by converting each row in the DataFrame to a list of items (transactions).
- We create a header table containing items that meet the minimum support threshold defined.
- We build an FP-tree using the transactions and the header table.

We then keep a list of the discovered patterns involving the provided sensitive attributes, which we use in the next steps. We choose FP-Growth because it offers us many advantages :

1. FP-Growth is particularly effective for analysing large sets of transactional data. This considerably reduces the number of passes required on the data compared with other algorithms.
2. **Variable creation:** Define binary variables for each sample in the dataset.
3. FP-Growth works well with sparse data sets, which is often the case in many real-life applications, particularly in the analysis of shopping baskets or user behaviour.
4. The FP-Growth algorithm offers a number of optimisation options, such as parallelisation or the use of optimised data structures, which can further improve its performance on very large datasets.

2.4 The metric to quantify the dataset balance

At the heart of the fairness optimization approach lies a carefully crafted metric designed to quantify the balance of sensitive attributes within a dataset. This metric serves as the cornerstone for the optimization process, providing a clear and interpretable measure of fairness, using patterns discovered in the dataset.

2.4.1 Definition of fairness score

We define the fairness score using the following formula:

$$\text{Fairness Score} = (\text{support}_{\max} - \text{support}_{\min}) \quad (2.1)$$

Where:

- $\text{support}_{\max}$ is the highest support of samples matching any patterns made of sensitive attributes
- $\text{support}_{\min}$ is the lowest support of samples matching any patterns of sensitive attributes

This metric yields a score between 0 and 1, where:

- A score of 0 represents perfect fairness (all attribute combinations are equally represented)
- A score of 1 indicates extreme unfairness (one combination dominates the dataset)

2.4.2 Advantages and justification

This fairness metric offers several key advantages:

1. **Interpretability:** The score is easily understood, with a clear range and meaning.
2. **Multi-attribute consideration:** It accounts for all combinations of sensible attributes, addressing intersectional fairness.
3. **Sensitivity to imbalance:** The metric is responsive to imbalances across all relevant itemsets, not just overall proportions. item **Proof of Reliability:** The metric measures how evenly different demographic groups are represented in the dataset. It identifies the most disparate groups through max-min comparisons, provides a clear direction for improvement (reduce the maximum gap) and enables targeted interventions for the most underrepresented groups.

2.4.3 Example calculation

Consider a dataset with two binary sensitive attributes: Gender (male/female) and Race (white/non-white). The possible combinations are:

Gender	Race
Male	White
Male	Non-white
Female	White
Female	Non-white

Table 2.1: Demographics categories

The table 2.1 shows an example dataset we will use.

Let's say the dataset has the following distribution:

Gender	Race	Percentage
Male	White	40%
Male	Non-white	30%
Female	White	20%
Female	Non-white	10%

Table 2.2: Dataset distribution by gender and ethnicity

This table 2.2 shows the distribution of the dataset used in 2.1

In this case:

- $\text{support}_{\max} = 0.40$ (Male, White)
- $\text{support}_{\min} = 0.10$ (Female, Non-white)

$$\text{Fairness Score} = (0.40 - 0.10) = 0.70$$

This score of 0.70 indicates that while there is some imbalance in the dataset, it's not extremely unfair. However, there is room for improvement to achieve a more balanced representation.

2.5 Constraint programming model: formalizing the optimization problem

To optimize dataset fairness, we formulate the problem as a constraint satisfaction problem and solve it using constraint programming techniques. This approach allows us to efficiently search for a subset of the original dataset that maximizes the fairness metric while satisfying other constraints.

2.5.1 Problem formulation

Let's define the key components of the constraint programming model:

- D : The original dataset with n samples
- x_i for $i \in \{1, \dots, n\}$: Binary variables indicating whether the i th sample is selected (1) or not (0)
- k : The desired size of the fair subset
- C : The set of all possible combinations of sensible attributes
- $\text{count}(c)$ for $c \in C$: The number of selected samples matching combination c

The constraint programming model is formulated as follows:

Minimize: fairness_score

Subject to: $\sum_{i=1}^n x_i = k$

$$\text{count}(c) = \sum_{i: \text{sample}_i \text{ matches } c} x_i \quad \forall c \in C$$

$$\text{fairness_score} = \left(\max_{c \in C} \frac{\text{count}(c)}{k} - \min_{c \in C} \frac{\text{count}(c)}{k} \right)$$

2.5.2 Implementation overview

While we won't delve into specific code, the implementation typically involves the following steps:

1. **Model initialization:** Set up the constraint programming model using a suitable library (in our case, Google OR-Tools).
2. **Variable creation:** Define binary variables for each sample in the dataset.
3. **Constraint addition:** Add constraints for the desired subset size and to count samples for each relevant pattern found.
4. **Objective definition:** Define the fairness score as the objective to maximize.
5. **Solver configuration:** Set up the solver with appropriate parameters (e.g., time limit).

2.6 Dataset adjustment: Implementing the solution found

Once we have solved the constraint programming model, we need to adjust the dataset based on the solution. This process involves either undersampling or oversampling, depending on the desired size of the fair dataset. The fairness optimization process incorporates an undersampling method when the coefficient is less than or equal to 1 and an oversampling method when it is above 1.

2.6.1 Undersampling

Methodology The undersampling process is relatively straightforward but effective:

- **Coefficient evaluation:** The process begins by checking if the provided coefficient is less than or equal to 1, indicating that undersampling is required.
- **Sample selection:** If undersampling is needed, we select a subset of the original data based on the 'selectedindices' provided in the solution. These indices are determined by the constraint solver to optimize fairness. This method is also used in the Fairmaker library we referenced in the first chapter.
- **Data reduction:** The original dataset is reduced to only include the rows corresponding to the selected indices, effectively creating a smaller, more balanced dataset.

Example:

- Original dataset: 1000 samples
- Desired size: 800 samples (80% of original)

Process:

1. Solve the constraint programming model to select 800 samples
2. Create a new dataset containing only the selected samples
3. The resulting dataset will have improved fairness while maintaining 80% of the original size

2.6.2 Synthetic data generation for fairness optimization(oversampling)

In addition to the oversampling technique previously discussed, we implement a hybrid approach that combines elements of both oversampling and perturbation techniques. This method is particularly effective for datasets with sensitive attributes and mixed data types.

Methodology: Consider an unbalanced dataset with three columns and ten rows, where one or more columns represent sensitive attributes. The synthetic data generation process follows these steps:

1. **Sensitive attribute combination selection:** For each new synthetic sample, we randomly select a combination of values for the sensible attributes from the original dataset. This ensures that the distribution of sensitive attributes in the synthetic data mirrors that of the original data.

2. **Sample selection:** We then identify all samples in the original dataset that match the selected combination of sensitive attributes. If matching samples exist, we randomly select one as a base for our synthetic sample. In cases where no matching samples are found, we resort to selecting a random sample from the entire dataset.
3. **Feature-wise synthesis:** For each non-sensitive attribute, we apply a type-specific synthesis method:
 - *Integer attributes:* We generate a new random integer within the range of values observed in the original dataset for that attribute.
 - *Floating-point attributes:* We add a small random perturbation to the original value. This perturbation is drawn from a normal distribution with a mean of zero and a standard deviation of 10% of the attribute's standard deviation in the original dataset.
 - *Categorical and string attributes:* We randomly select a value from the set of unique values observed for that attribute in the original dataset.
 - *Other data types:* For any other data types, we retain the original value without modification.

Advantages This approach offers several advantages:

1. It preserves the distribution of sensitive attributes, ensuring that the synthetic data does not introduce new biases.
2. It maintains the statistical properties of the original dataset while introducing controlled variability.
3. It handles mixed data types effectively, making it versatile for various real-world datasets.

Example: Let's say we need to generate 20 (based on the solver solution) new samples:

1. For each new sample:
 - (a) Randomly select a combination of sensitive attributes
 - (b) Find samples in the original dataset matching this combination
 - (c) If matching samples exist, randomly select one as the base sample; otherwise, select a random sample from the entire dataset
 - (d) For each non-sensitive attribute of the base sample:
 - For integer columns: Generate a new random integer within the range of the original data
 - For float columns: Add a small random perturbation (normal distribution with 0 mean and 10)
 - For categorical or string columns: Randomly choose a value from the unique values in the original data
 - For other types: Keep the original value
2. Repeat this process until 20 new samples are generated.

We therefore make sure the data generated have the same type as the original data. This approach ensures that the new samples have similar characteristics to the existing samples within each pattern, maintaining data integrity while improving fairness.

2.7 Example scenario

Let's consider a small dataset with 8 samples and one sensitive attribute (Gender). We'll demonstrate both undersampling and oversampling cases.

2.7.1 Dataset

This dataset contains 4 features: The Id, the Gender of the concerned individual, the Income and the Purchase, a boolean indicating whether or not a purchase has been made.

ID	Gender	Income	Purchase
1	Male	High	Yes
2	Male	Medium	Yes
3	Male	High	Yes
4	Male	Low	No
5	Male	Medium	Yes
6	Female	Low	No
7	Female	Medium	Yes
8	Male	High	Yes

Table 2.3: Initial Dataset

2.7.2 Undersampling Case (coefficient = 0.75)

2.7.2.1 Step 1: Mark sensitive attributes and get the coefficient

We mark 'Gender' as the sensitive attribute and set the coefficient to 0.75.

2.7.2.2 Step 2: Transform into transaction collection and apply FP-Growth

FP-Growth (with a threshold of 0.1) gives us:

1. Frequent 1-item sets: {Male}: 6, {Female}: 2, {High}: 3, {Medium}: 3, {Low}: 2, {Yes}: 6, {No}: 2
2. Frequent 2-item sets: {Male, High}: 3, {Male, Medium}: 2, {Male, Yes}: 5, {High, Yes}: 3, {Medium, Yes}: 3
3. Frequent 3-item sets: {Male, High, Yes}: 3, {Male, Medium, Yes}: 2

2.7.2.3 Step 3: Generate combinations of sensitive attributes

Generated combinations: [('Gender', 'Male'), ('Gender', 'Female')]

Calculated supports:

- ('Gender', 'Male'): $6/8 = 0.75$
- ('Gender', 'Female'): $2/8 = 0.25$

Initial fairness score: $1 - (0.75 - 0.25) = 0.5$

The initial fairness is 0.5, indicating a significant imbalance.

2.7.2.4 Step 4: ConstraintGenerator creates a CP-SAT model

Generated constraints include:

1. Decision variables: $x[i]$ for i from 0 to 7 (8 boolean variables, 1: selected, 0: not selected)
2. Size constraint: $\sum x = 6$ (because $8 * 0.75 = 6$)
3. Count variables:
 - $\text{count}[(\text{'Gender'}, \text{'Male'})] = x[0] + x[1] + x[2] + x[3] + x[4] + x[7]$ (count number of males)
 - $\text{count}[(\text{'Gender'}, \text{'Female'})] = x[5] + x[6]$ (count number of females)
4. $\text{model.Maximize}(\text{fairness_score}) = \text{fairness_score} * 75 == 1000000 * 75 - (\text{max_count} - \text{min_count}) * 1000000$

2.7.2.5 Step 5: ConstraintSolver solves the CP-SAT model

Suppose it finds the optimal solution: $x = [1, 1, 1, 0, 0, 1, 1, 1]$ This solution selects indices [0, 1, 2, 5, 6, 7], corresponding to rows 1, 2, 3, 6, 7, 8 of the original dataset.

Resulting dataset:

ID	Gender	Income	Purchase
1	Male	High	Yes
2	Male	Medium	Yes
3	Male	High	Yes
6	Female	Low	No
7	Female	Medium	Yes
8	Male	High	Yes

Table 2.4: Resulting Dataset after Undersampling

The table 2.4 shows the result obtained after the process. The lines 4 and 5 have been removed.

2.7.2.6 Step 6: Evaluation of final fairness

The FairnessEvaluator calculates the fairness of the adjusted dataset:

1. Supports for each combination:
 - $(\text{'Gender'}, \text{'Male'})$: $4/6 \approx 0.67$
 - $(\text{'Gender'}, \text{'Female'})$: $2/6 \approx 0.33$

Final fairness score: $1 - (0.67 - 0.33) \approx 0.66$

The final fairness is approximately 0.66, which represents a slight improvement from the initial fairness of 0.5.

2.7.3 Oversampling case (coefficient = 1.25)

For the oversampling case, we'll use the same initial dataset but with a coefficient of 1.25, aiming to increase the dataset size to 10 samples ($8 * 1.25 = 10$).

2.7.3.1 Steps 1-4

These steps remain the same as in the undersampling case, except the size constraint in Step 4 becomes: $\sum x = 10$

2.7.3.2 Step 5: ConstraintSolver solves the CP-SAT model

The solution found keeps all original samples and indicates we need to generate 2 new samples.

2.7.3.3 Step 6: Synthetic Data Generation

Following the oversampling methodology:

1. We randomly select combinations of sensitive attributes:
 - New sample 1: ('Gender', 'Female')
 - New sample 2: ('Gender', 'Female')
2. For each new sample, we select a base sample matching the chosen combination (or a random sample if no match exists), then apply feature-wise synthesis:

New sample 1 (based on row 7):

ID	Gender	Income	Purchase
9	Female	Low	Yes

New sample 2 (based on row 6):

ID	Gender	Income	Purchase
10	Female	Medium	No

2.7.3.4 Step 7: Evaluation of final fairness

The FairnessEvaluator calculates the fairness of the adjusted dataset:

1. Supports for each combination:
 - ('Gender', 'Male'): $6/10 = 0.6$
 - ('Gender', 'Female'): $4/10 = 0.4$

Final fairness score: $1 - (0.6 - 0.4) = 0.8$

The final fairness is 0.8, which represents a significant improvement from the initial fairness of 0.5.

This example demonstrates how the fairness optimization tool can improve dataset balance through both undersampling and oversampling techniques, depending on the specified coefficient.

2.8 Web application

The Web application is based on the framework, and the following process is going :

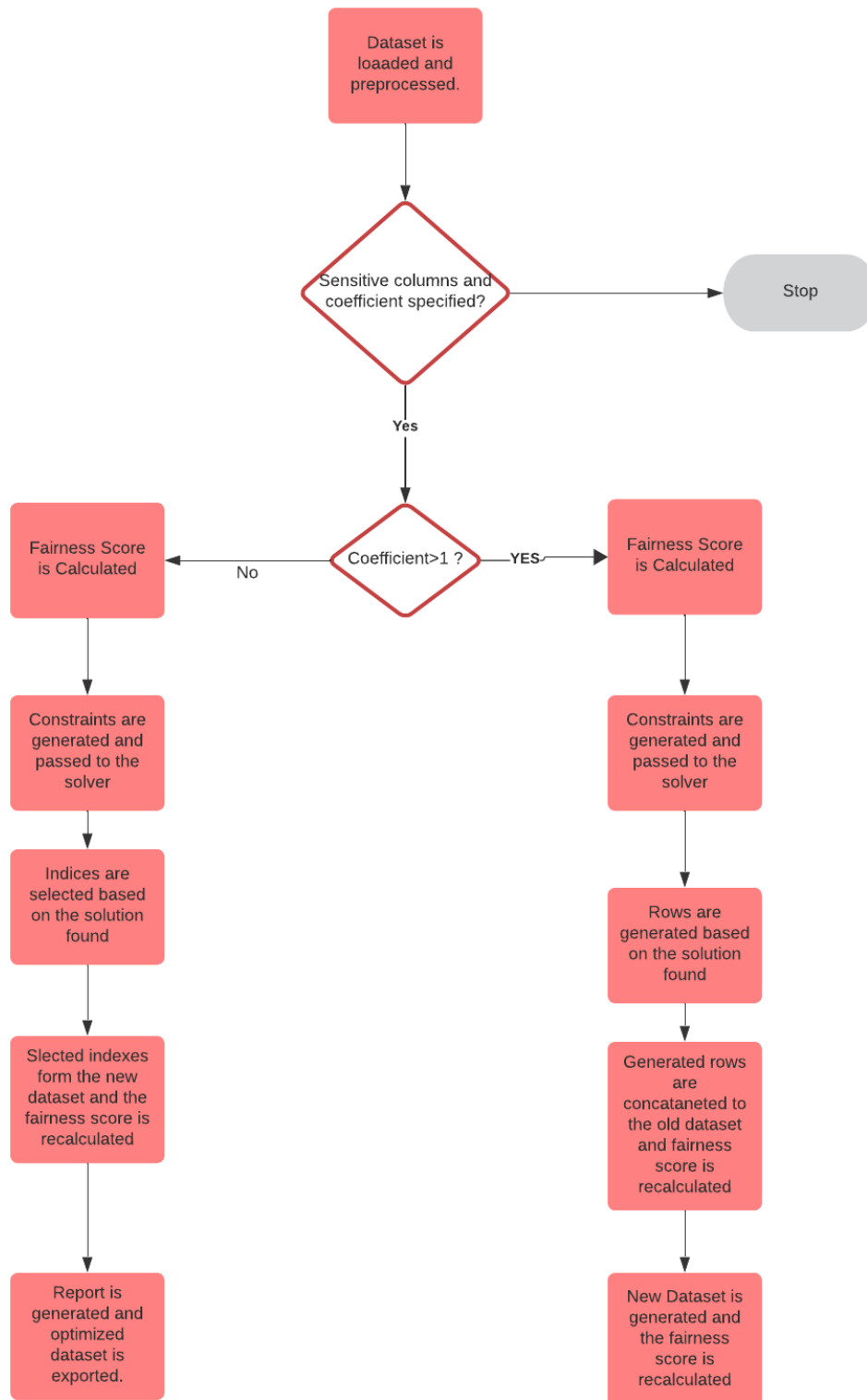


Figure 2.1: FairnessOptimizer Web Application user process

The 2.1 shows the user process on the web app, allowing him to generate, when the solution is

found, a fairer dataset in addition to an optimization report. What we mainly do is :

- Load and Preprocess the dataset
- Generate, depending on the coefficient and the sensible attributes provided, either a smaller or a bigger version of the dataset, while improving the fairness score.
- Generate a downloadable report of the optimization process, clarifying the improvement done.

2.9 UML diagrams

We will indicate how the system works through UML diagrams.

2.9.1 Sequence Diagram

With the next figure, we will illustrate the FairnesOptimizer process through a sequence diagram.

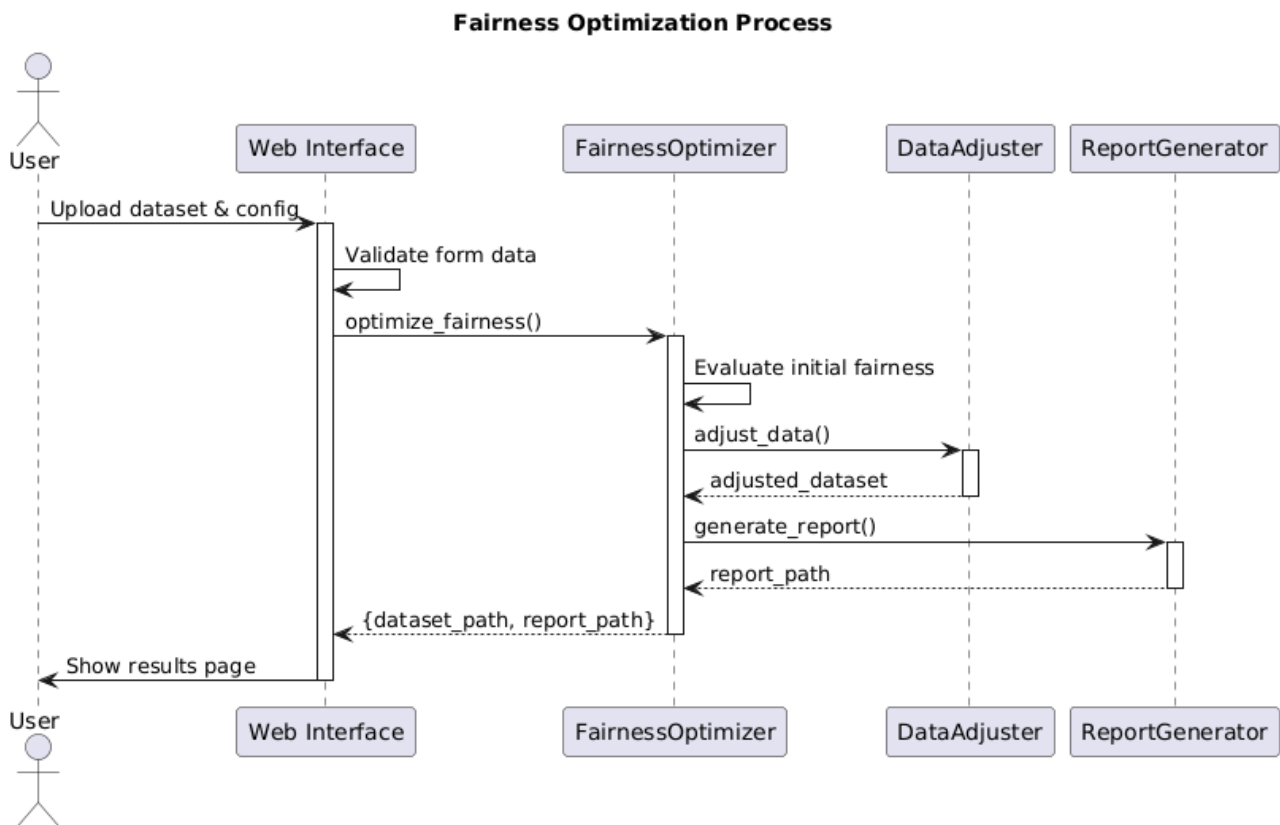


Figure 2.2: FairnessOptimizer Sequence Diagram

The figure 2.2 shows the inner process of the Fairness Optimizer and its components during the treatment.

2.9.2 Use Case Diagram

With the next figure, we will illustrate the use cases offered to the data scientist using the solution.

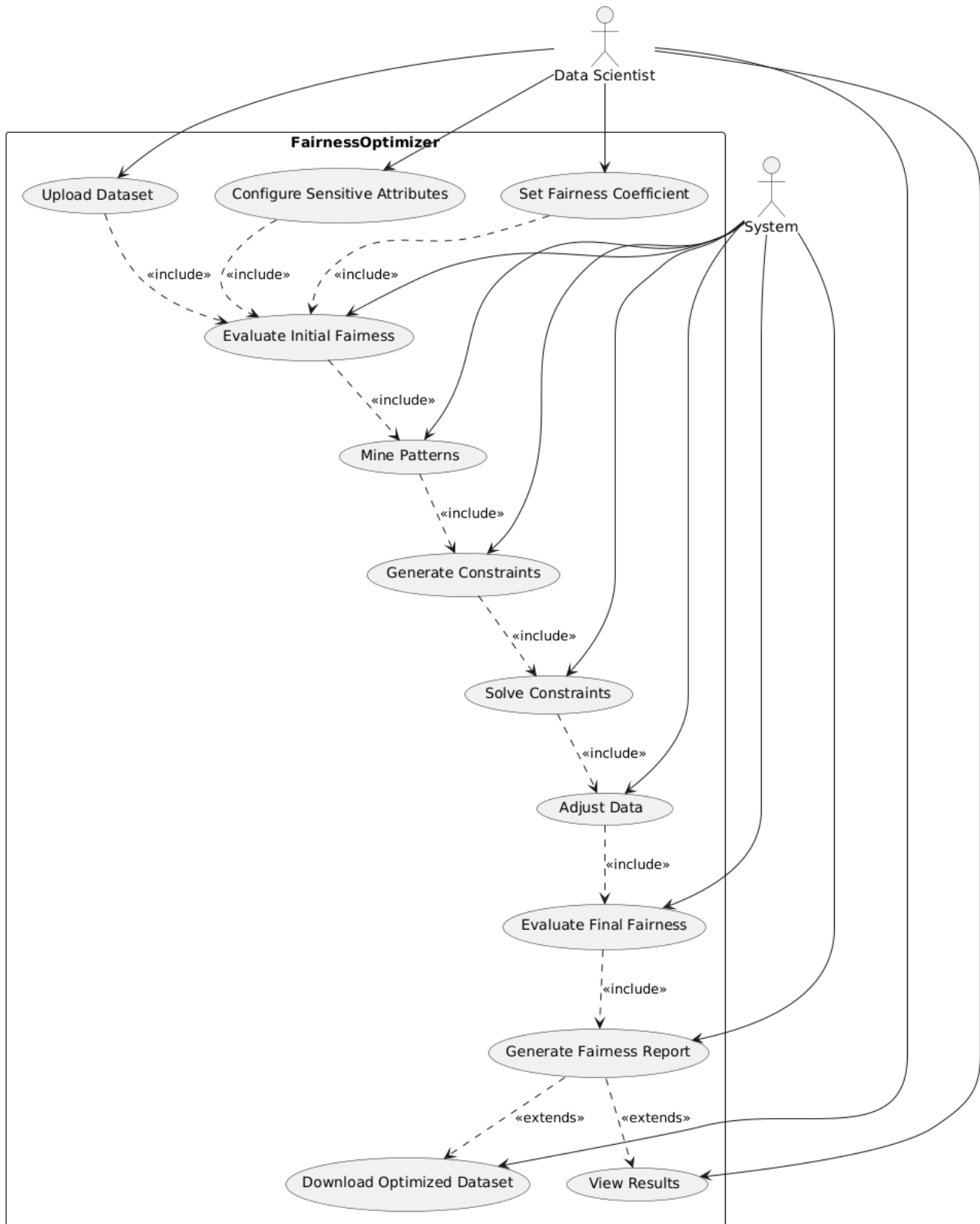


Figure 2.3: FairnessOptimizer Use Case Diagram

The figure 2.3 shows the different use cases offered by the system.

2.9.3 Class Diagram

With the next figure, we will illustrate the class diagram.

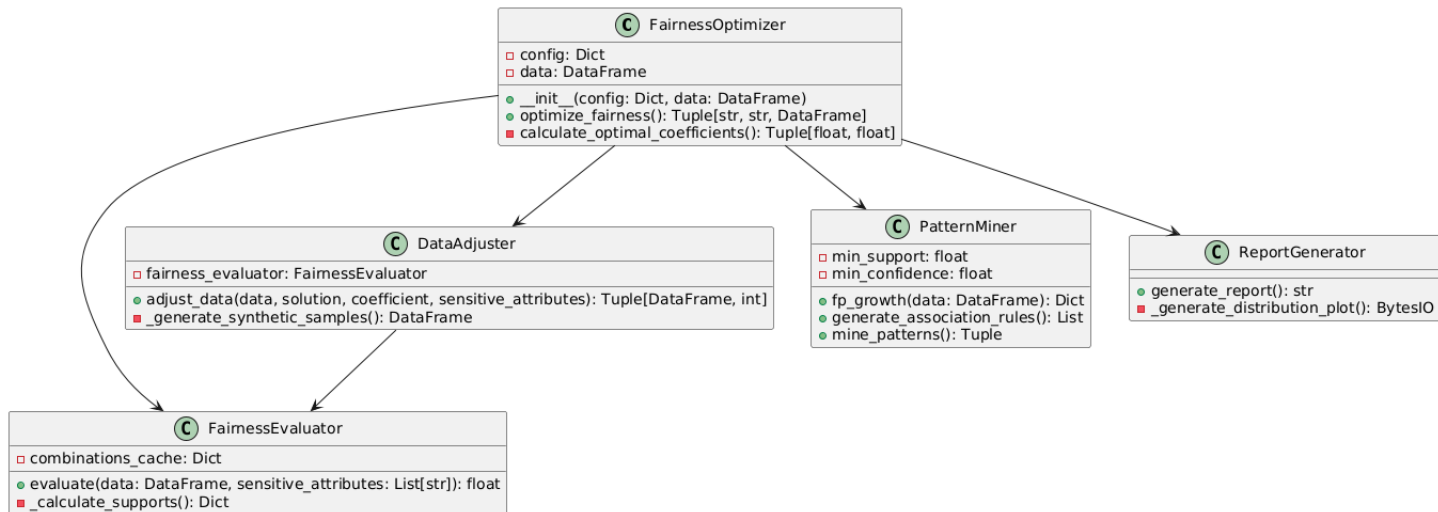


Figure 2.4: FairnessOptimizer Class Diagram

The figure 2.4 shows the different classes that make the system and how they interact with one another.

Conclusion

In conclusion, this chapter has addressed a critical aspect of fairness in machine learning systems through the development of a comprehensive fairness optimization tool. As artificial intelligence continues to permeate various sectors of society, the need for trustworthy and unbiased AI systems becomes increasingly paramount. This research has culminated in the creation of a Python-based package that empowers data scientists to assess and mitigate bias in machine learning models, thereby contributing to the broader goal of ethical AI development. The implemented FairnessOptimizer system offers a robust framework for evaluating and improving the fairness of datasets with respect to sensitive attributes. By incorporating pattern mining, constraint generation, and data adjustment techniques, the tool provides a holistic approach to fairness optimization. Furthermore, the development of a user-friendly web interface makes the tool accessible to a wider audience of data scientists and researchers. This interface allows for easy upload of datasets and configuration of optimization parameters, streamlining the process of fairness assessment and mitigation. The reporting functionality provides comprehensive insights into the optimization process. By generating detailed reports that include fairness scores, configuration details, and distribution comparisons of sensitive attributes, the tool offers transparency and interpretability, which are crucial aspects of trustworthy AI systems.

Experimental results and discussion

In this chapter, we will prove the efficiency of the method, based on comparison of the distribution of data in datasets obtained after using different approaches, including the fairness optimizer.

3.1 FairnessOptimizer vs FairMaker vs Random Split

3.1.1 Dataset Study

For the needs of this study, we generated a random dataset with the fellow features. We will also consider "Previouscompany", "Religion" and "Region" as sensitive attributes. The following image shows an overview of the used dataset.

	Age	Previouscompany	Religion	NumberOfKids	Region	InterviewScore
0	23	Sobebra	Other	8	Alibori	10
1	62	Moov	Buddhist	9	Littoral	>10
2	41	Moov	Other	2	Collines	>10
3	45	Moov	Buddhist	0	Littoral	10
4	21	Sobebra	Other	8	Couffo	<10
...						
	Age	Previouscompany	Religion	NumberOfKids	Region	InterviewScore
495	27	Sobebra	Buddhist	5	Collines	<10
496	23	MTN	Buddhist	7	Plateau	<10
497	40	Moov	Christian	9	Littoral	>10
498	35	Moov	Buddhist	6	Atacora	<10
499	43	Moov	Muslim	5	Zou	>10

Figure 3.1: The study dataset

The figure 3.1 shows a snippet of the first example dataset we will use in this study. The dataset is made of 500 rows and 6 columns. For the needs of this study, we generated a random dataset with the fellow features. For the needs of this study, we will consider "Previouscompany", "Religion" and "Region" as sensitive attributes. Let's compare the initial distributions on those sensitive attributes.

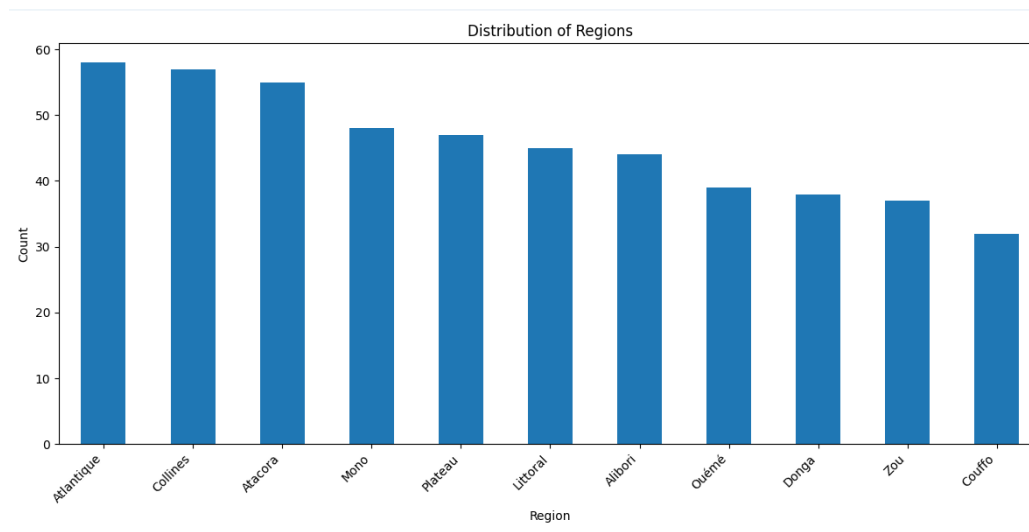


Figure 3.2: "Region" Feature initial Distribution

The figure 3.2 shows the distribution of the "Region" feature in the original dataset. As we can see, the "Region" feature contain several possible dataset observations. The "Atlantique" and "Collines" Observation are the most represented. The initial fairness Score regarding this column is 0.052000000000000005 according to the fairness optimizer library's report. Let's vizualize it with the next figure.

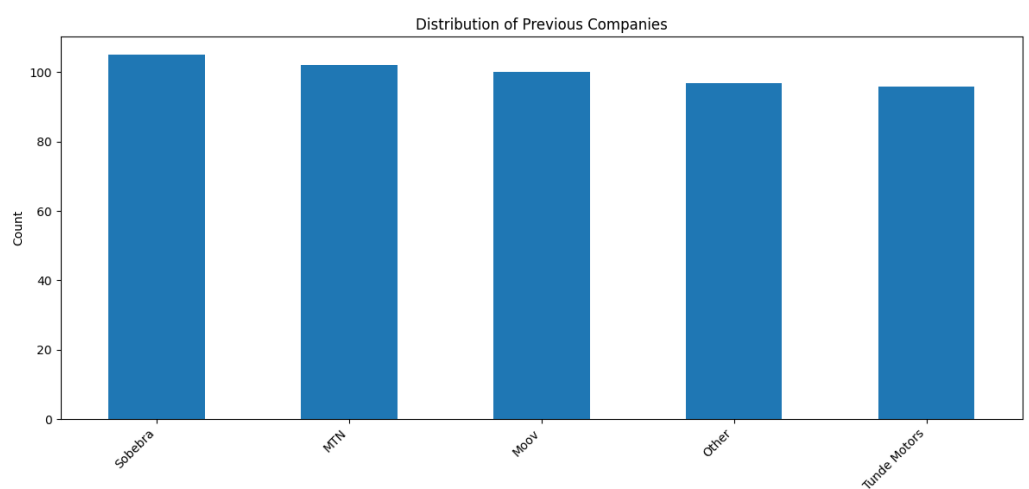


Figure 3.3: "PreviousCompanies" Feature initial Distribution

The figure 3.3 shows the distribution of the "PreviousCompanies" feature in the original dataset. This feature has 5 possible dataset observations. The distribution is more balanced compared to the previous. The initial fairness Score regarding this column is 0.0180 according to the library's report.

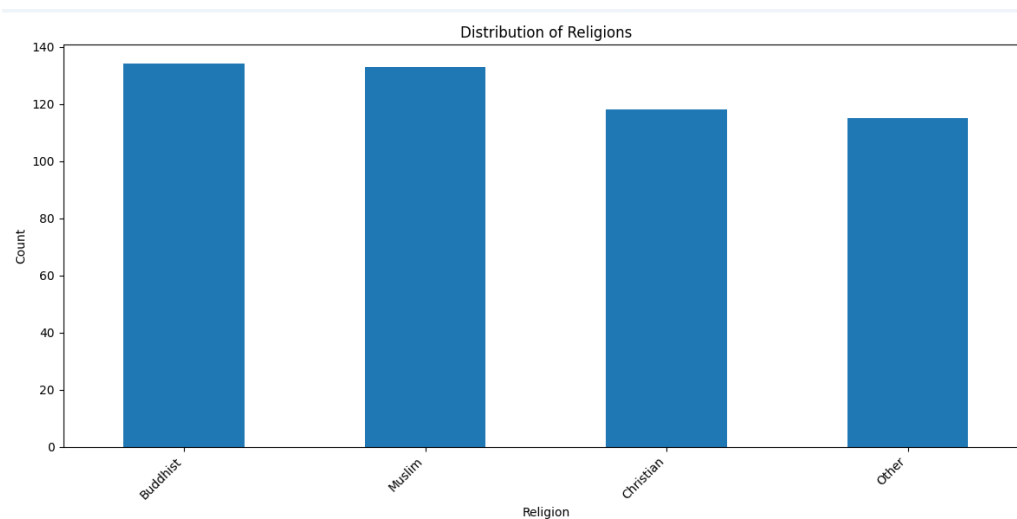


Figure 3.4: "Religion" Feature initial Distribution

The figure 3.4 shows the distribution of the "Religion" feature in the original dataset. This feature has 4 possible dataset observations. The "Buddhist" and "Muslim" distributions are a bit more represented than the other ones. The initial fairness Score regarding this column is 0.038000000000000006 according to the library's report. Let's now analyze a second dataset.

	Name	Gender	Antecedents	Religion	LoanGranted
0	Corey Cortez	Male	Bad	Christian	True
1	Paul Fisher	Female	Bad	Christian	False
2	Kenneth Smith	Female	Good	Christian	True
3	Jason Jones	Male	Bad	Muslim	True
4	Carol Taylor	Male	Good	Christian	True
...					
	Name	Gender	Antecedents	Religion	LoanGranted
4995	Craig Raymond	Female	Good	Muslim	False
4996	Maria Henry	Male	Good	Christian	True
4997	Megan Carlson	Male	Good	Christian	True
4998	Casey Wilkins	Female	Bad	Muslim	False
4999	Logan Newton	Male	Bad	Christian	True

Figure 3.5: 2nd Example Dataset

The figure 3.5 shows a snippet of the second example dataset we will use in this study. The dataset is made of 5000 rows and 5 columns. For the needs of this study, we generated a random dataset with the fellow features. Here, we will consider the "Gender" Column as a sensitive attribute. The initial distribution is as follows:

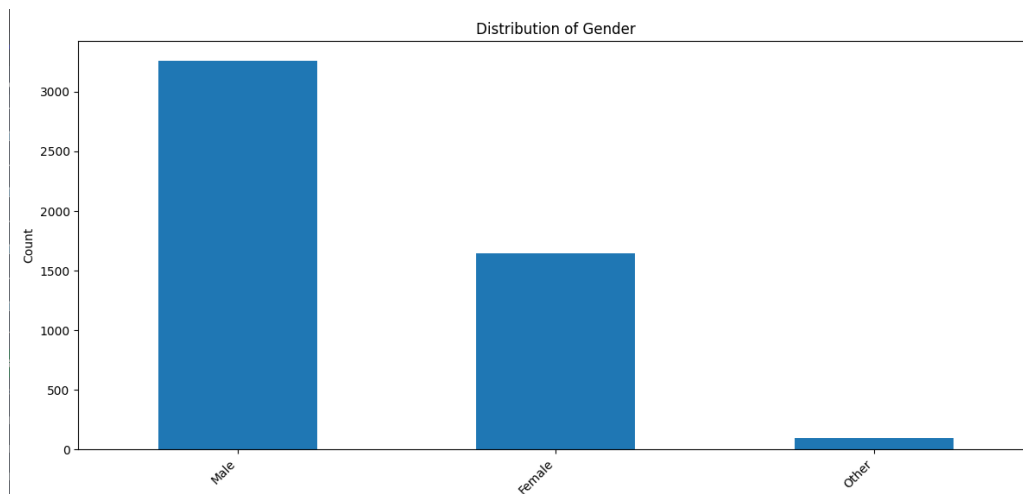


Figure 3.6: "Gender" Feature Initial Distribution

The figure 3.6 shows the distribution of the feature in the second dataset. We can notice a predominance of the "Male" subgroup. Initial fairness is 0.6328.

3.1.2 Results comparison

After applying the random split to a 0.8 ratio, we obtain the following results on the dataset represented by 3.1.

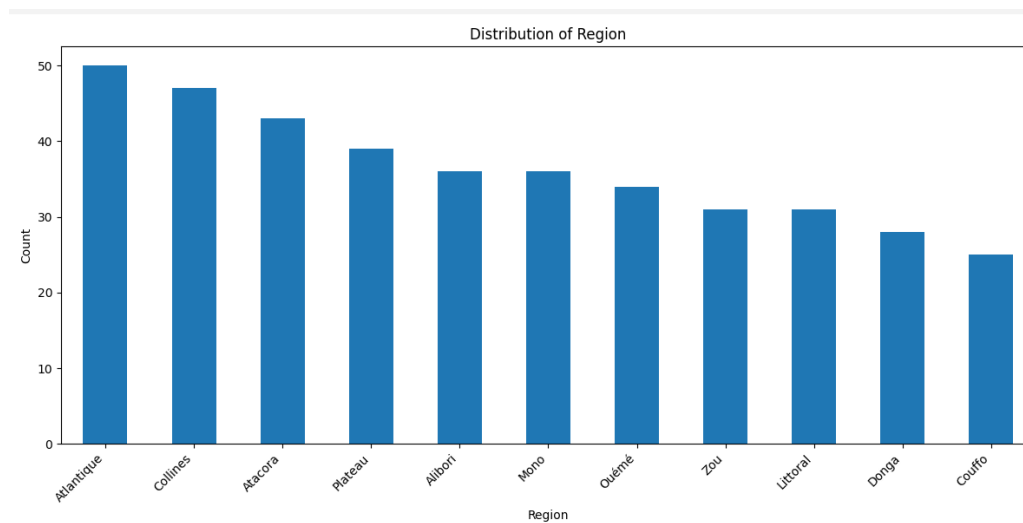


Figure 3.7: "Region" Feature Random split Distribution

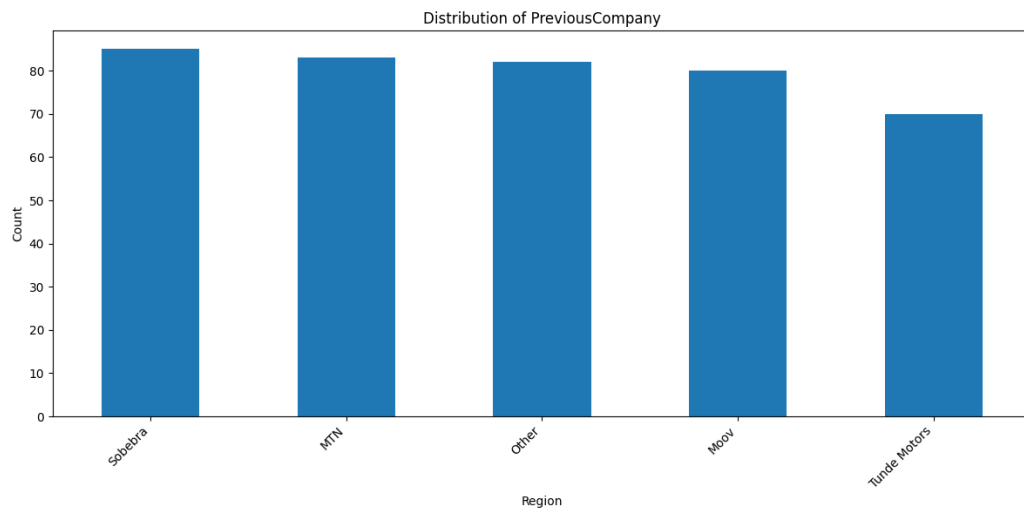


Figure 3.8: "Religion" Feature Random split Distribution

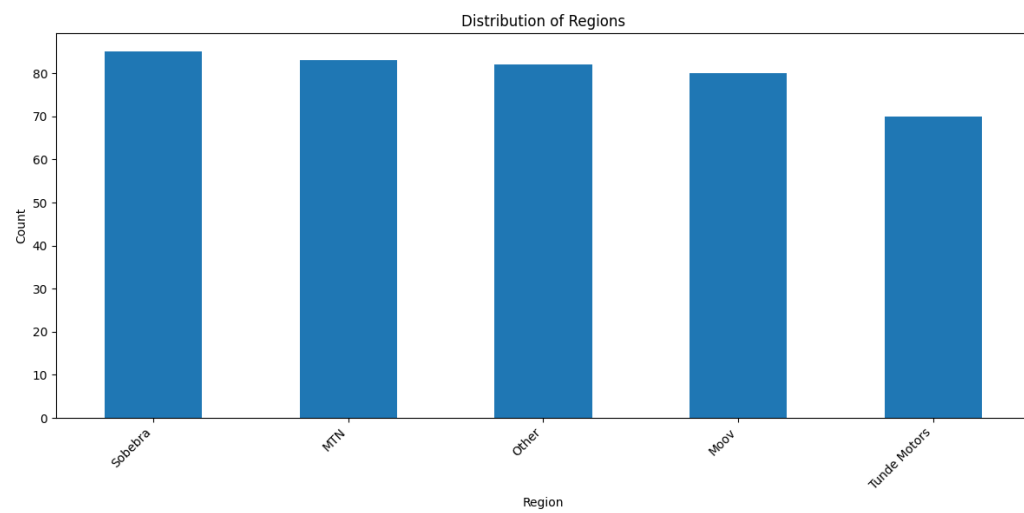


Figure 3.9: "Previous Companies" Feature Random split Distribution

Globally, the Random Split leaves similar proportions compared to the original dataset. Let's now try the FairnessOptimizer Split, with the same 0.8 split coefficient.

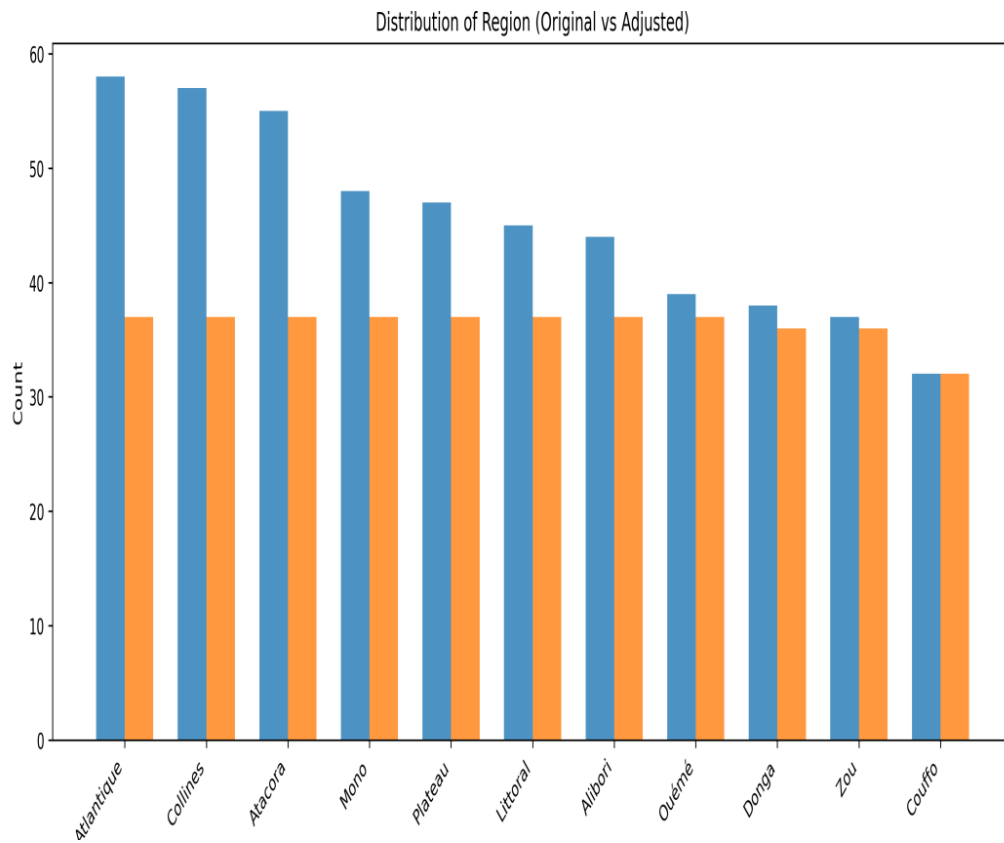


Figure 3.10: "Region" Feature FairnessOptimizer split Distribution

The score obtained is 0.0125, from an initial score of 0.052000000000000005. The fairmaker library returns a 0.02 score for the generated dataset. The "Couffo" region is still the less represented but the overall distribution is more balanced.

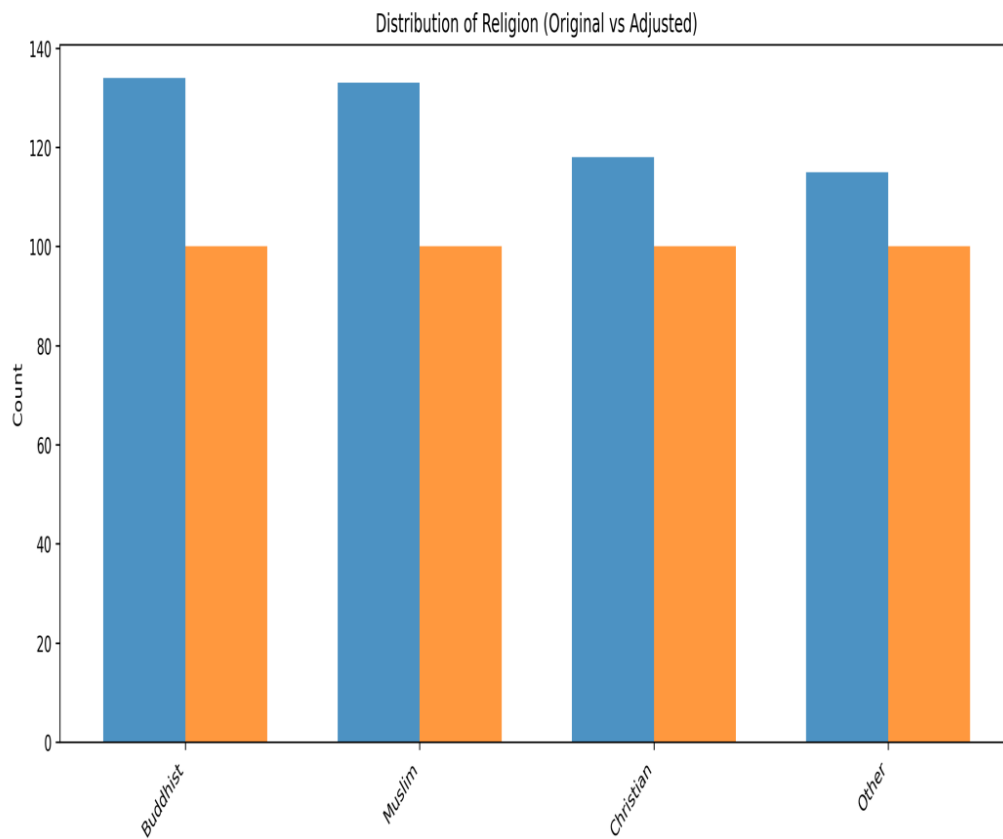


Figure 3.11: "Religion" Feature FairnessOptimizer split Distribution

The score obtained is 0.000, from an initial score of 0.038000000000000006, showing perfect fair distribution. The Fairmaker library returns a 0.00 score for the generated dataset.

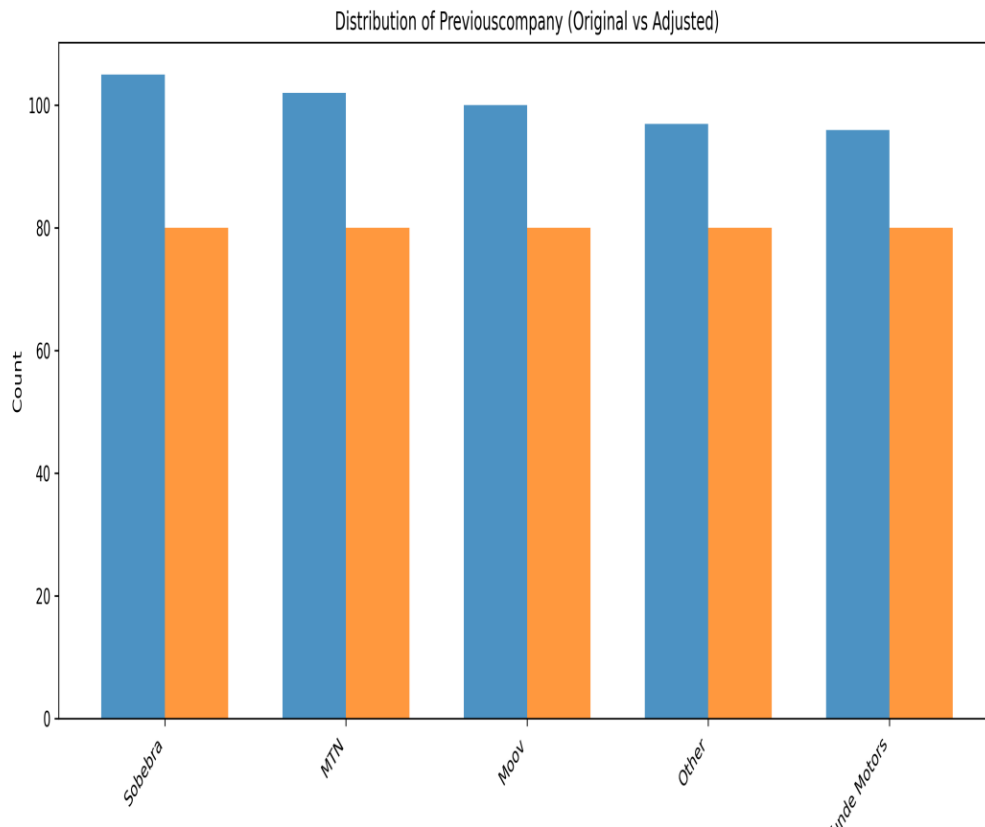


Figure 3.12: "Previouscompany" Feature FairnessOptimizer split Distribution

The score obtained is 0.00, also showing perfect fair distribution, from an initial score of 0.0180. Same score is returned by the fairmaker library for the generated dataset. Let's now try the Fairmaker approach, with the same split coefficient.

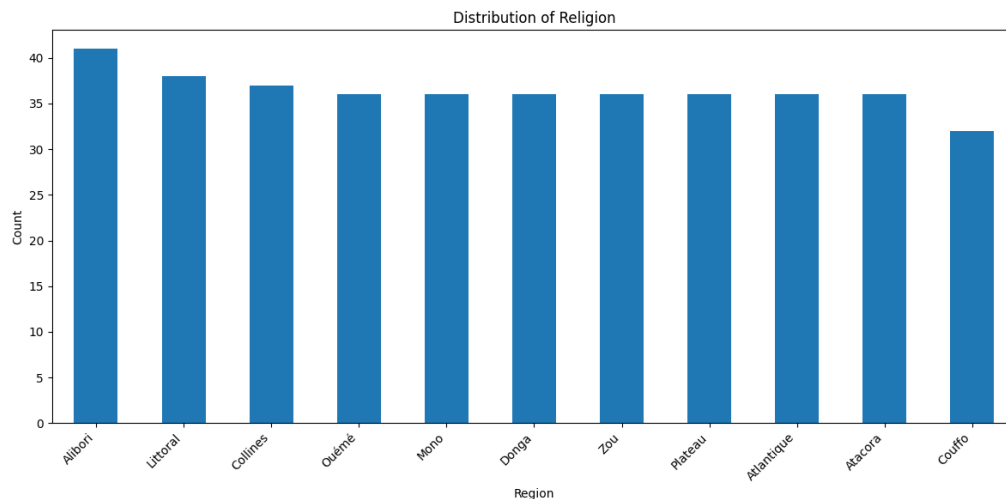


Figure 3.13: "Region" Feature FairMaker split Distribution

The figure 3.13 shows the split distribution after applying Fairmaker. The score obtained is 0.02, from an initial score of 0.15. Similar to the FairnessOptimizer approach, but not with a perfect result.

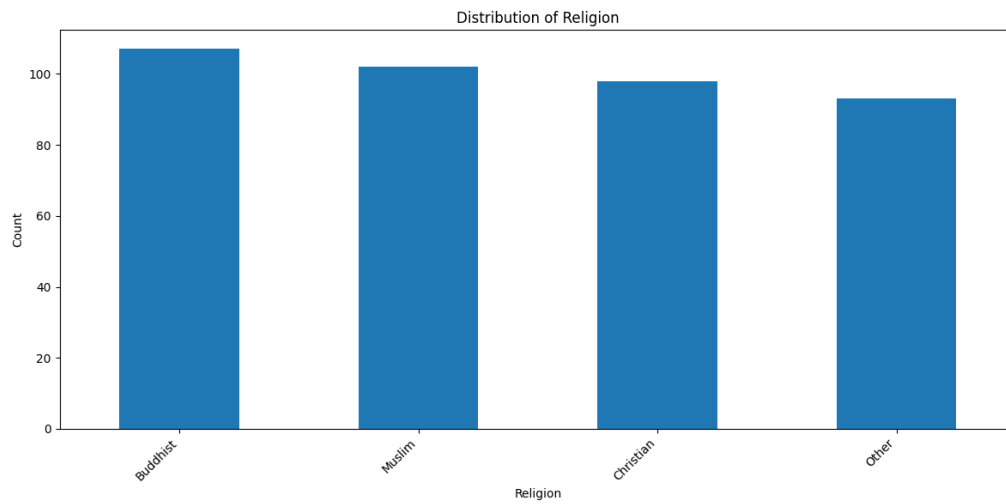


Figure 3.14: "Religion" Feature FairMaker split Distribution

The score obtained is 0.04, from an initial score of 0.07. Less unbalanced representation.

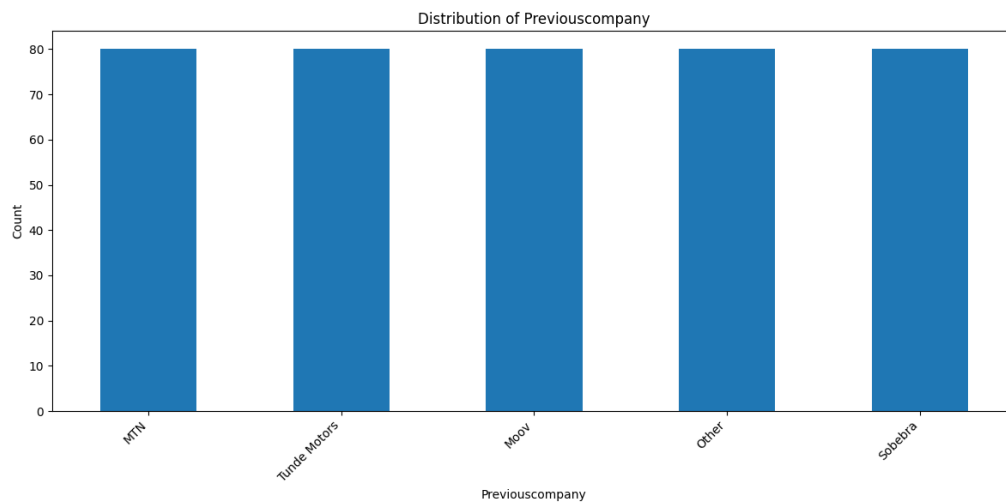


Figure 3.15: "Previouscompany" Feature FairMaker split Distribution

The score obtained is 0.00, also showing perfect fair distribution, from an initial score of 0.03. Let's now compare the results of fairmaker and fairnessoptimizer for the second dataset, for a 0.1 split coefficient.

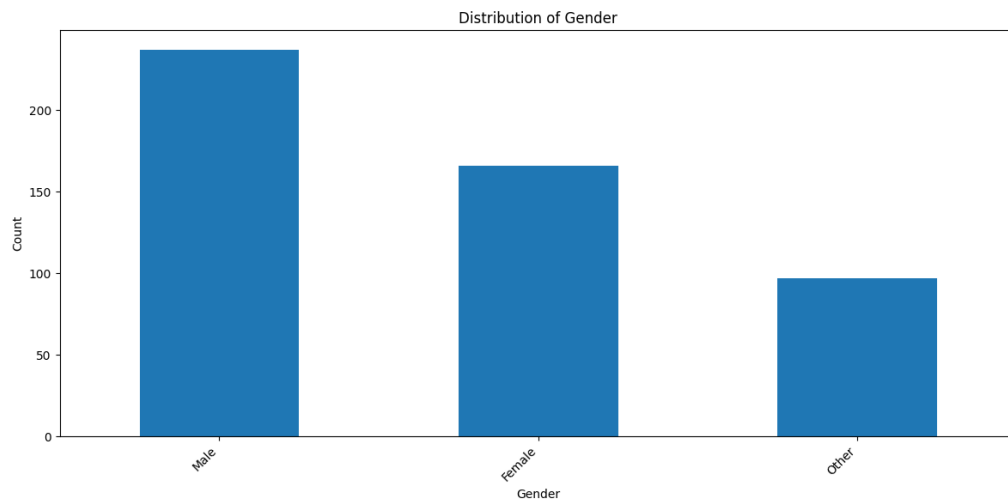


Figure 3.16: "Gender" Feature Fairmakersplit Distribution Dataset 2

The coefficient obtained with the Fairmaker approach is 0.28. With the FairnessOptimizer it is 0.21. Let's check the FairnessOptimizer results.

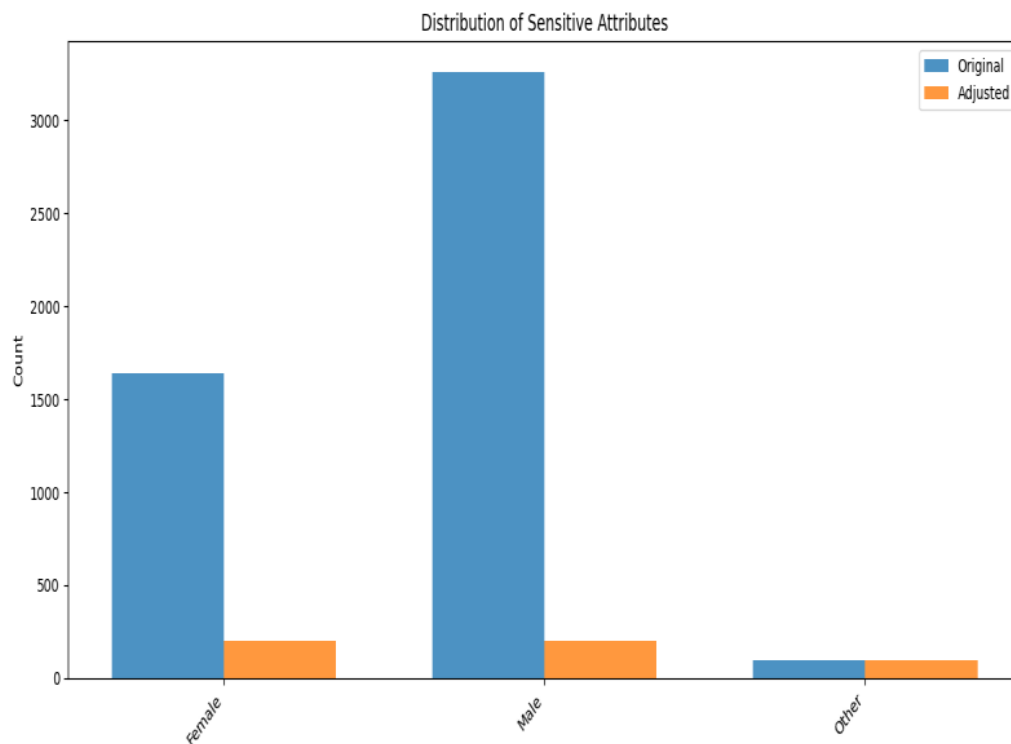


Figure 3.17: "Gender" Feature FairnessOptimizer split Distribution Dataset 2

The figure 3.17 shows the distribution of the "Gender" feature in the optimized dataset generated by the FairnessOptimizer vs the one in the original dataset. . The distribution seems similar to the one generated by the Fairmaker.

3.2 FairnessOptimizer vs Random Oversampling

We chose to implement GANs (Generative Adversarial Networks) we already mentioned in the first chapter to generate random data, but data that is close to the original one for each feature.

	Age	Previouscompany	Religion	NumberOfKids	Region	InterviewScore
0	21	MTN	Muslim	8	Atacora	>10
1	24	MTN	Muslim	10	Alibori	10
2	37	Sobebra	Muslim	3	Donga	>10
3	27	MTN	Christian	0	Littoral	>10
4	30	Sobebra	Muslim	4	Plateau	>10
.						
	Age	Previouscompany	Religion	NumberOfKids	Region	InterviewScore
1395	30	MTN	Other	2	Collines	<10
1396	30	MTN	Other	2	Alibori	>10
1397	30	Sobebra	Buddhist	3	Ouémé	<10
1398	30	Sobebra	Buddhist	3	Ouémé	>10
1399	30	MTN	Christian	2	Alibori	>10

Figure 3.18: Oversampled GAN Approach dataset shape

	Age	Previouscompany	Religion	NumberOfKids	Region	InterviewScore
0	23	Sobebra	Other	8	Alibori	10
1	62	Moov	Buddhist	9	Littoral	>10
2	41	Moov	Other	2	Collines	>10
3	45	Moov	Buddhist	0	Littoral	10
4	21	Sobebra	Other	8	Couffo	<10
.						
	Age	Previouscompany	Religion	NumberOfKids	Region	InterviewScore
1395	30	MTN	Other	2	Collines	<10
1396	30	MTN	Other	2	Alibori	>10
1397	30	Sobebra	Buddhist	3	Ouémé	<10
1398	30	Sobebra	Buddhist	3	Ouémé	>10
1399	30	MTN	Christian	2	Alibori	>10

Figure 3.19: Oversampled FairnessOptimizer Approach dataset shape

Let's compare the results. We first consider the random approach.

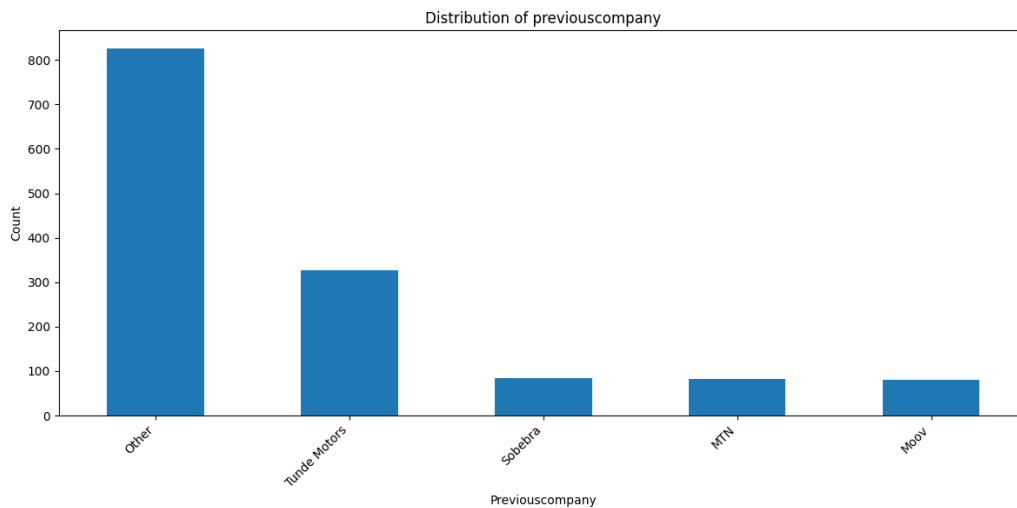


Figure 3.20: "Previouscompany" Feature Random Oversample Distribution

The figure 3.20 shows the distribution of "Previouscompany" feature on the random split. We can notice a huge disparity in the distributions, with a heavy tendency for the "Other" observation. The score obtained is 0.533 from a 0.018 in the original dataset.

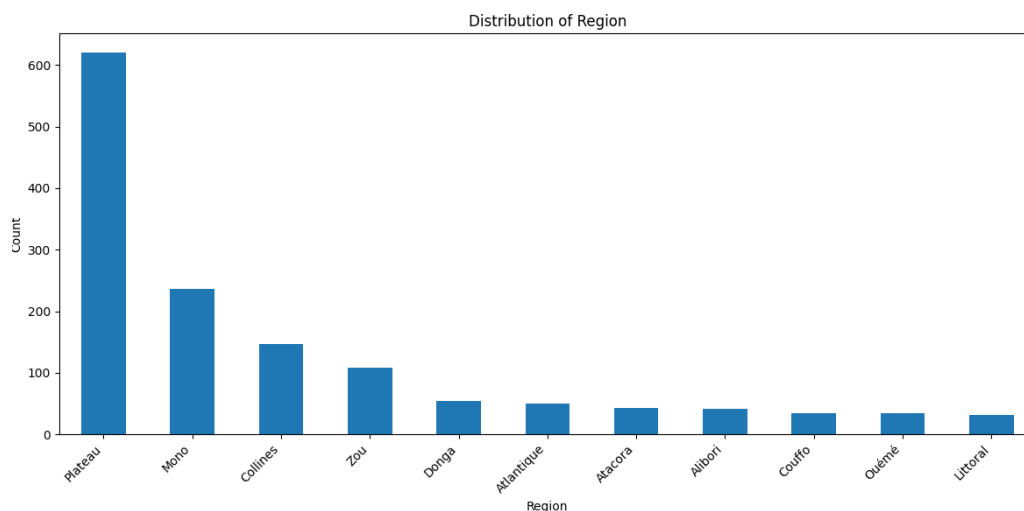


Figure 3.21: "Region" Feature Random Oversample Distribution

This figure shows the distribution of the "Region" feature. We can again notice a huge disparity in the distributions, with a heavy tendency for the "Plateau" observation. The score obtained is 0.421 from 0.038 in the original dataset.

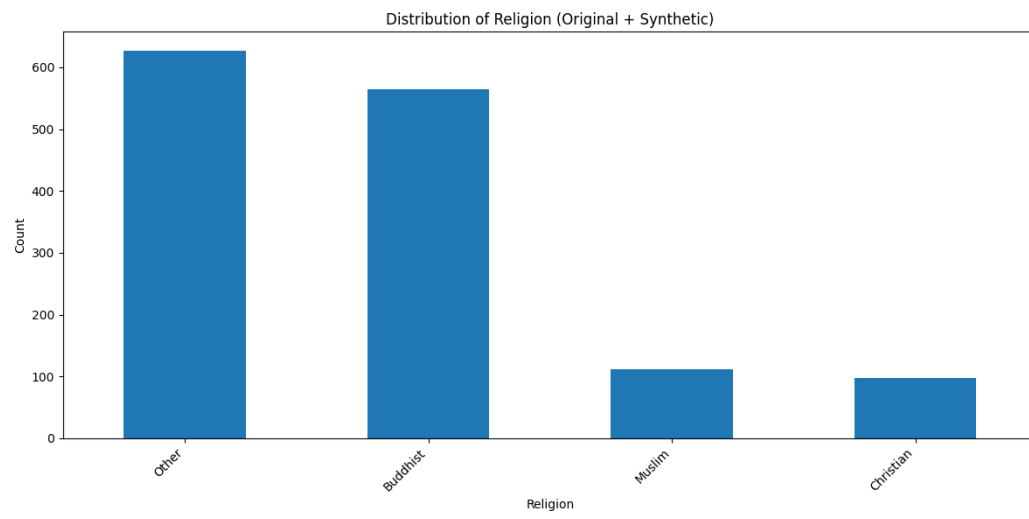


Figure 3.22: "Religion" Feature Random Oversample Distribution

We once can again notice a huge unfairness in the distributions, with a heavy tendency for the "Other" and "Buddhist" observations. The score obtained is 0.378 from a 0.038 original score. Overall, the random approach generates very unbalanced results. Let's now compare it to the FairnessOptimizer approach.

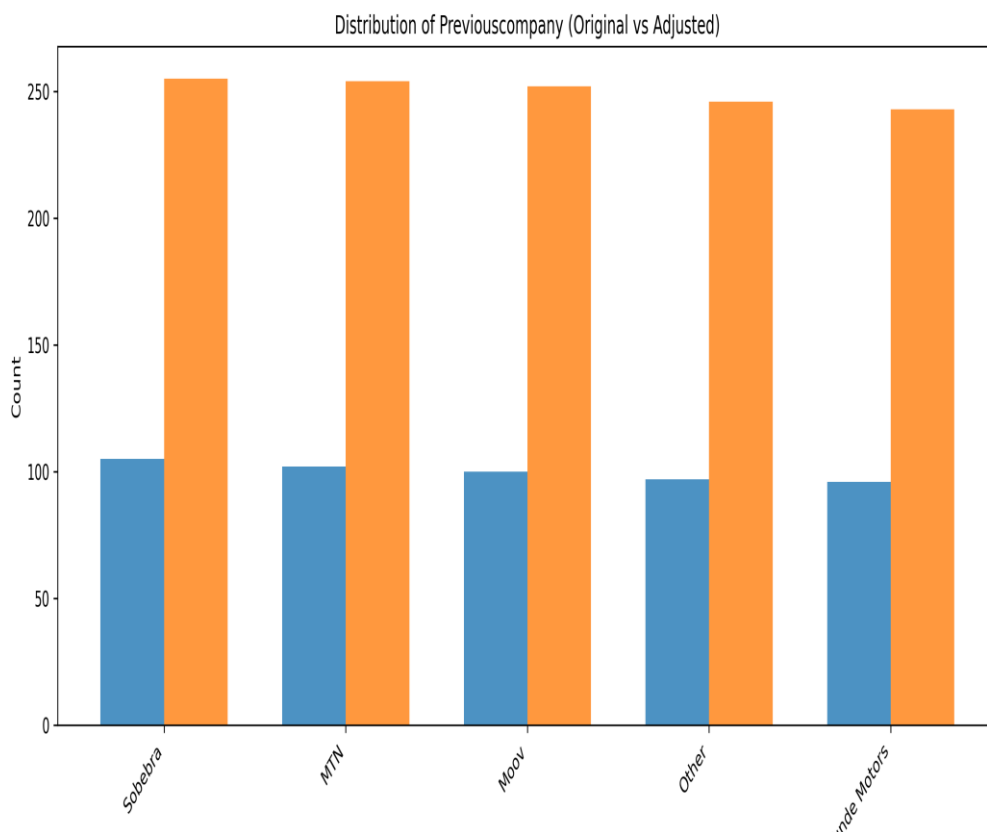


Figure 3.23: "Previouscompany" Feature Fairness Optimizer Oversample Distribution

The figure 3.23 shows a comparison between the distribution of the original dataset and the optimized dataset made by FairOp, relatively to the Previouscompany feature. We can notice a little improvement in the distribution, which is still dominated by the "Buddhist" observation but a lot less. The score is 0.014 from a 0.018 in the original dataset.

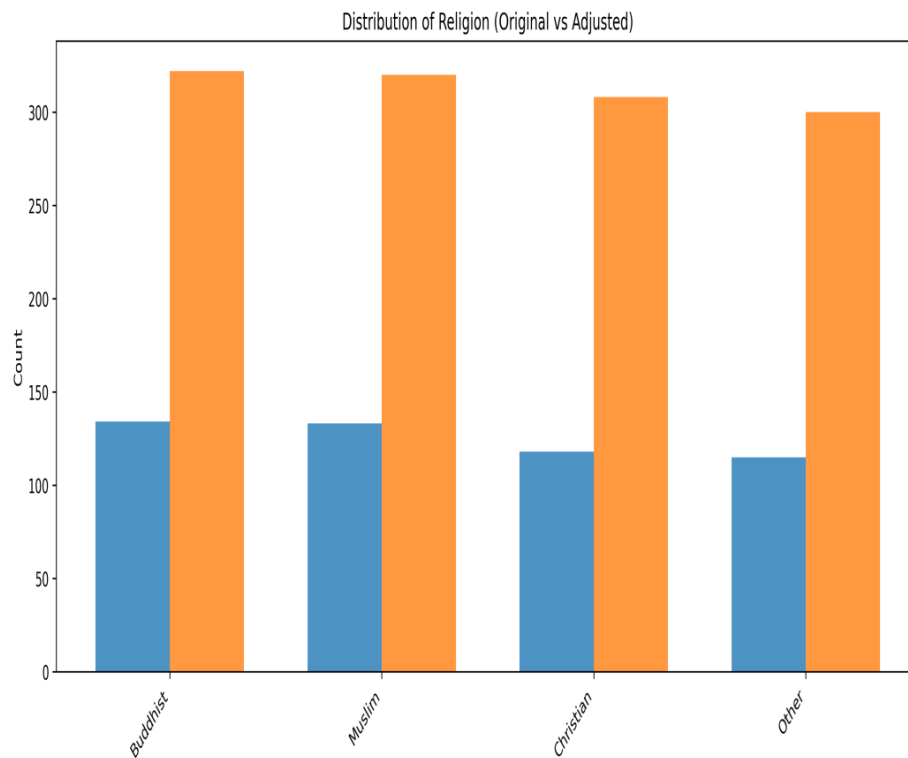


Figure 3.24: "Religion" Feature FairnessOptimizer Oversample Distribution

The figure 3.24 shows a comparison between the distribution of the original dataset and the optimized dataset made by FairOp, relatively to the Religion feature. We can yet again notice a little improvement in the distribution, which is less dominated by the "Christian" and "Other" observations. The score is 0.0184 from a 0.038 in the original dataset.

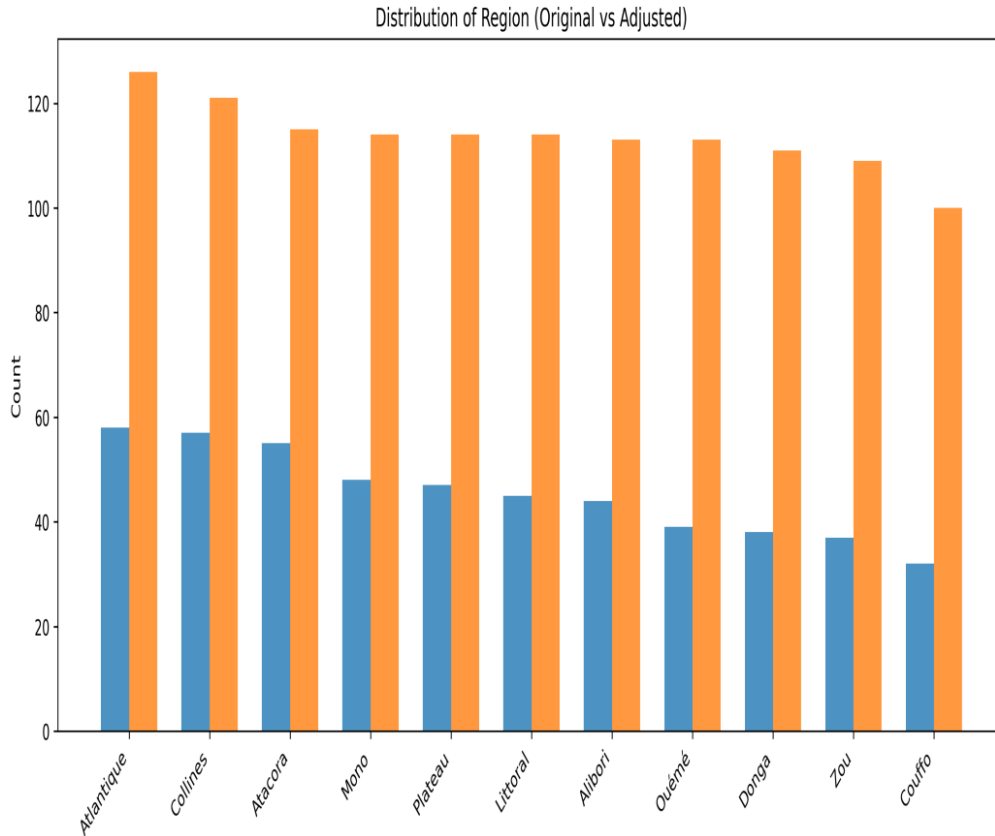


Figure 3.25: "Region" Feature FairnessOptimizer Oversample Distribution

The figure 3.25 shows a comparison between the distribution of the original dataset and the optimized dataset made by FairOp, relatively to the Region feature. Again, we notice a little improvement in the distribution, which is still dominated by the "Atlantique" observation. The score is 0.038 from a 0.052 in the original dataset.

In conclusion of this comparison section, we can clearly see that the FairnessOptimizer approach provides much better results than random approaches, both in the cases of undersampling and oversampling. We can also notice the FairMaker gives similar results to the FairnessOptimizer approach, due to the proximity of their processes, but it's important to raise the fact that the overall solution finding time is slightly faster in the case of the FairnessOptimizer process (generally 3 seconds vs 1 seconds in the case of the first dataset studied, and 13 seconds vs 5 seconds in the case of the second). The speed difference is likely due to the use of the FP-Growth algorithm for the FairnessOptimizer. The random approach generally increases the level of disproportion in the representation of data, while the fairness optimizer approach always provide a better score, if it is possible. In both cases, and especially in oversampling case, getting perfect representation highly depends on the coefficient applied. According to the observations, especially in cases of huge disparities, the bigger, the better, given that enough data has to be generated for the minority groups to reach majority groups.

3.3 FairnessOptimizer vs ADASYN Oversampling

We also applied the ADASYN algorithm we mentioned earlier on the dataset used before. ADASYN helps us generate data while solving imbalance problem. Let's see the results, with another dataset

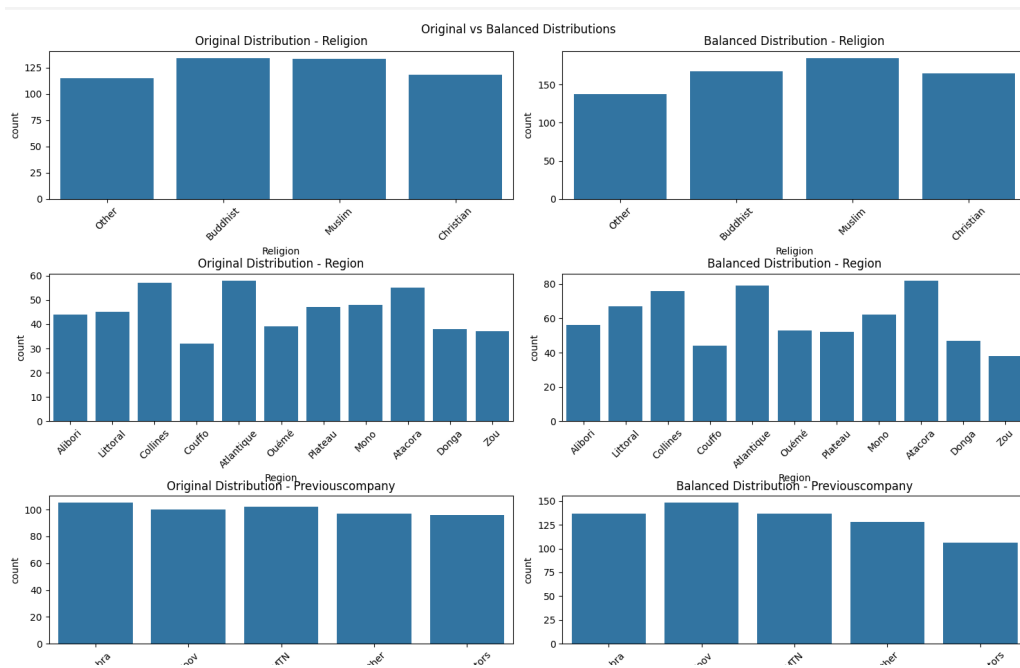


Figure 3.26: Oversampling with ADASYN comparison on different features

From an initial fairness score (based on the fairnessOptimizer metric) of 0.732, we obtain a general score of 0.7, which is a small improvement. We applied ADASYN with the "minority" sampling strategy, which allows to generate data by considering balancing the distribution.

3.4 Web Application

We built a small web application using Flask for the needs of this work.

The screenshot shows the 'FairnessOptimizer' web application interface. It includes a file upload section with a 'Choose File' button and a text input showing 'thesis 2.csv'. Below this is a 'Configuration JSON' section with a text area containing the following JSON:

```
{
  "sensitive_attributes": ["Region", "Religion"],
  "coefficient": 2.5
}
```

An 'Optimize' button is located below the JSON configuration. At the bottom of the interface, there is a loading indicator and the text 'Optimizing... This may take a few minutes.'

Figure 3.27: "FairnessOptimizer Web Application page Step 1

After specifying the sensitive attributes and the coefficient the process can begin.

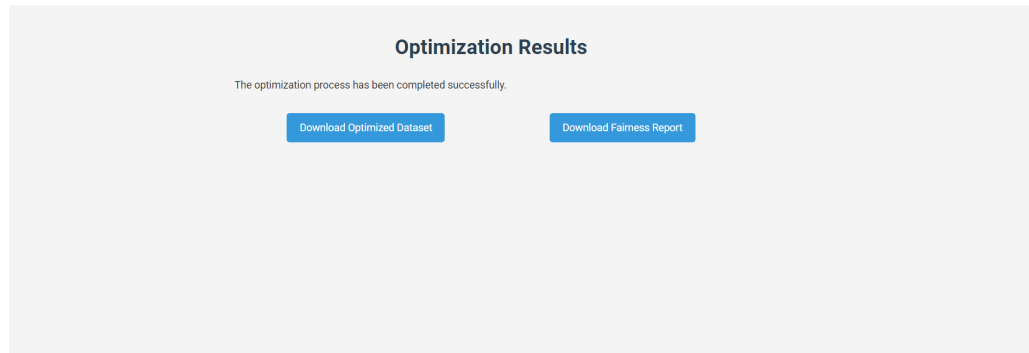


Figure 3.28: "FairnessOptimizer Web Application page Step 2

When the user downloads the report, he has access to a PDF File he can download.

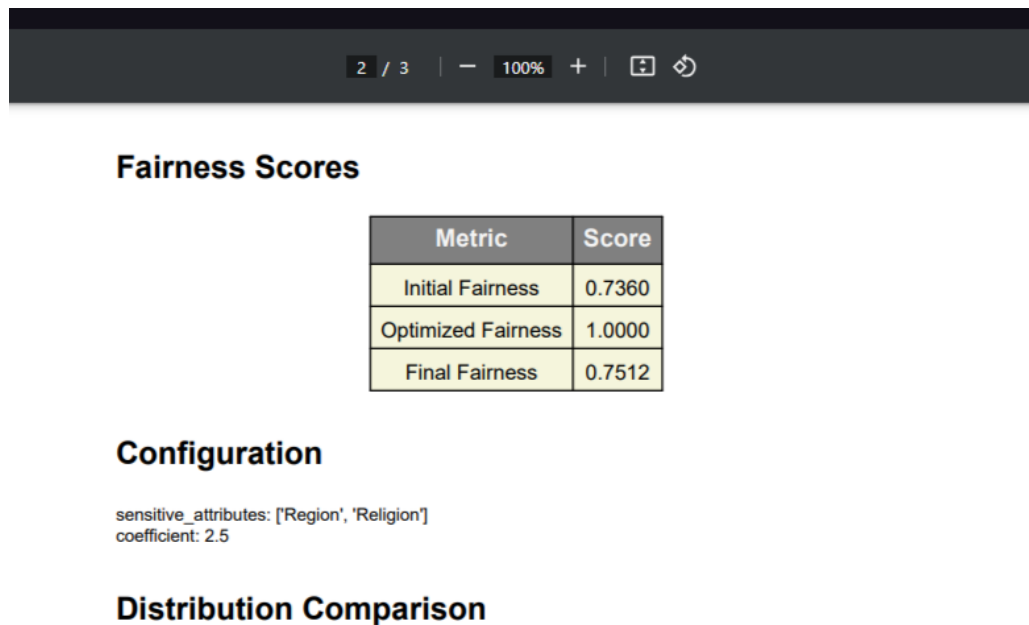


Figure 3.29: "FairnessOptimizer Web Application page

This figure, 3.29, shows the first page of the optimization report automatically generated, displaying the improvement made.

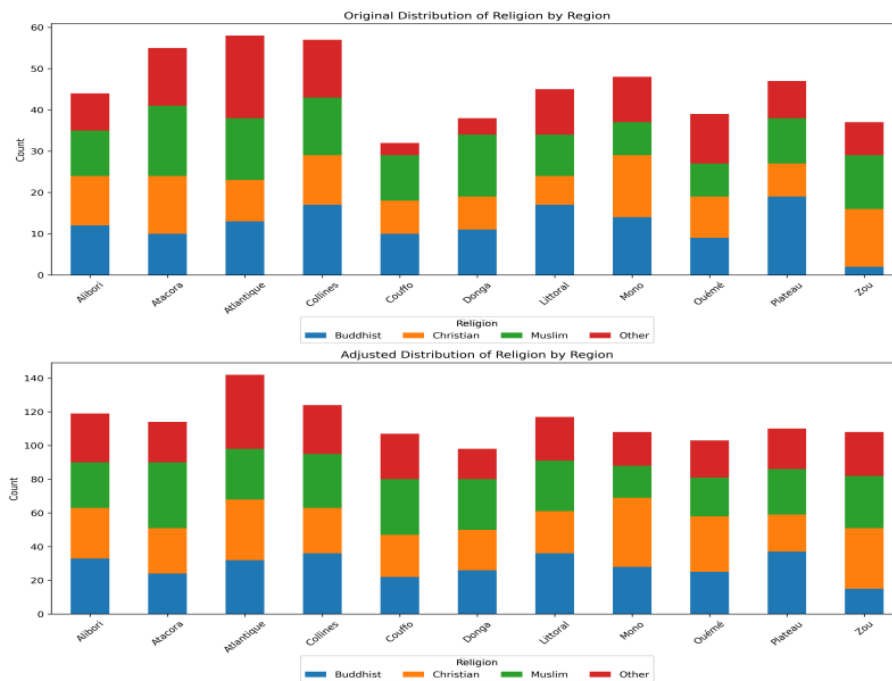


Figure 3.30: "FairnessOptimizer Web Application page

The figure 3.30 shows the 2nd page of the report generated, showing a diagram with the distributions of the new dataset. Two sensitive attributes had been considered, thus explaining the two parameters in the diagrams.

Conclusion

In this chapter, we presented the results of the fairness optimizer approach. We showed that the FairnessOptimizer has significantly improved results, compared to the random approaches commonly used. The performance of the framework highly depends on the number of columns selected and on the coefficient, relevant to the needs of the data scientist.

Conclusion

In this work, we developed a FairnessOptimizer system that evaluates and optimizes the fairness of datasets by adjusting data based on sensitive attributes and generating comprehensive reports. The system uses various components, including pattern mining, constraint solving, and data adjustment, to ensure that fairness scores are improved, with the results presented in a detailed report. In conclusion, this work represents a significant step towards addressing the critical issue of fairness in AI systems. By providing data scientists with a powerful and accessible tool for fairness optimization, we contribute to the ongoing efforts to build more trustworthy and ethical AI systems. But despite the promising results and the satisfaction of our initial expectations, there is room for improvement and future work. One potential avenue for enhancement is the optimization of the tool's performance, particularly in terms of computation time for larger datasets. This could be achieved by incorporating more advanced data mining techniques or parallel processing methods. Additionally, extending the fairness metrics and integrating them with existing open-source solutions, such as AIF360, could further enhance the tool's capabilities and interoperability with other fairness-focused libraries. This integration would contribute to the standardization of fairness assessment practices in the machine learning community. Lastly, future work could focus on expanding the tool's functionality to directly assess model performance in addition to dataset fairness. This would provide a more comprehensive evaluation of machine learning systems, encompassing both fairness and accuracy metrics. As machine learning continues to evolve and impact various aspects of society, tools like ours will play an important role in ensuring that these systems are developed and deployed responsibly, with fairness and equity at their core.

Bibliography

- [1] Clustering with unsupervised learning and key concepts of it — DS/ML course.
- [2] Kaggle Titanic Dataset : Cleaning & Split data into train, validation, and test set.
- [3] C. Atten. The Ultimate Beginner Guide of Supervised Learning, July 2024.
- [4] S. Bird, M. Dudík, R. Edgar, B. Horn, R. Lutz, V. Milan, M. Sameki, H. Wallach, K. Walker, and A. Design. Fairlearn: A toolkit for assessing and improving fairness in AI.
- [5] A. Castelnovo, R. Crupi, G. Greco, D. Regoli, I. G. Penco, and A. C. Cosentini. A clarification of the nuances in the fairness metrics landscape. *Scientific Reports*, 12(1):4209, Mar. 2022. Publisher: Nature Publishing Group.
- [6] H. He, Y. Bai, E. A. Garcia, and S. Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328, June 2008. ISSN: 2161-4407.
- [7] S. Kangralkar. Types of Biases in Data, July 2023.
- [8] S. Kangralkar. Types of Biases in Data, July 2023.
- [9] M. Z. Kurdi. *Natural Language Processing and Computational Linguistics: Speech, Morphology and Syntax*. John Wiley & Sons, Aug. 2016.
- [10] K. Ramamohanarao and J. Bailey. Emerging patterns: mining and applications. In *Proceedings of International Conference on Intelligent Sensing and Information Processing, 2004*, pages 409–414, Jan. 2004.
- [11] A. A. Sulaiman. Basic Introduction to Reinforcement Learning, July 2023.
- [12] A. Tyler. *Expert Systems Research Trends*. Nova Science Publishers, 2007.
- [13] P. Whig. *Sustainable Development through Machine Learning, AI and IoT*. Springer Nature. Google-Books-ID: WOkjEQAAQBAJ.

Webography

- **Fairness in ML:** <https://www.labelia.org/fr/blog/fairness-dans-le-machine-learning> – last accessed: Nov 23, 2024
- **Machine Learning:** https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=1248577140 – last accessed: Oct 1, 2024
- **The Ultimate Beginner Guide of Supervised Learning:** <https://medium.datadriveninvestor.com/exploring-the-basics-of-supervised-learning-bfc18fe5ce72> – last accessed: Oct 1, 2024
- **Clustering with Unsupervised Learning and Key Concepts of It:** https://machine-learning-and-data-science-with-python.readthedocs.io/en/latest/assignment1_unsup_ml.html – last accessed: Oct 2, 2024
- **Basic Introduction to Reinforcement Learning:** <https://medium.com/@Khalboy10/basic-introduction-to-reinforcement-learning-c37c12924dd7> – last accessed: Oct 2, 2024
- **Types of Biases in Data:** <https://towardsdatascience.com/types-of-biases-in-data-cafc4f2634fb> – last accessed: Oct 2, 2024
- **Fairness Metrics:** <https://mlr3fairness.mlr-org.com/articles/measures-vignette.html> – last accessed: Oct 2, 2024
- **FP Growth Algorithm in Data Mining:** <https://www.javatpoint.com/fp-growth-algorithm-in-data-mining> – last accessed: Oct 2, 2024
- **Data Types in Machine Learning – What Data Can ML Use:** <https://trainingdata.pro/data-types-in-machine-learning> – last accessed: Oct 2, 2024
- **Bias in AI: What it is, Types, Examples & 6 Ways to Fix It:** <https://research.aimultiple.com/ai-bias/> – last accessed: Oct 2, 2024

Appendix

Thesis Synthesis

Summary of Main Contributions

This thesis presents a novel approach to improve fairness in datasets using pattern mining techniques coupled with innovative sampling methods. The main contributions of this work are as follows:

- Development of a library named FairnessOptimizer that uses pattern mining and constraint programming techniques to identify and mitigate biases in datasets.
- Implementation of a constraint-based intelligent sampling method to rebalance datasets while preserving their utility.
- Integration of synthetic data generation techniques, particularly Generative Adversarial Networks (GANs), to increase dataset diversity while maintaining fairness.
- Creation of a web user interface to facilitate the use of FairnessOptimizer by machine learning practitioners.

Experimental Results and Future Perspectives

The experiments conducted in this thesis demonstrated the effectiveness of FairnessOptimizer compared to traditional methods of random data splitting and oversampling. The main results include:

- Significant improvement in fairness scores for sensitive attributes.
- More balanced distribution of sensitive features in datasets processed by FairnessOptimizer compared to random splitting methods.
- Better preservation of data structure and utility compared to random oversampling techniques based on GANs.

These results open up several perspectives for future work:

- Extending FairnessOptimizer to account for other types of biases and fairness metrics.
- Exploring transfer learning techniques to apply knowledge gained from one dataset to other similar datasets.

- Integrating interpretability methods to better understand the decisions made by the algorithm during fairness optimization.
- Studying the long-term impact of using FairnessOptimizer on the performance and fairness of machine learning models in real-world applications.

In conclusion, this thesis significantly contributes to the field of responsible AI by providing a powerful and flexible tool for detecting and mitigating biases in datasets. FairnessOptimizer offers a more nuanced and contextual approach to balancing datasets, paving the way for fairer and more equitable AI models in various applications.

Contents

Dedication	ii
Acknowledgements	iii
Résumé	iv
.....	iv
Abstract	v
.....	v
List of figures	vi
List of tables	viii
Introduction	1
1 State of the art and background	4
1.1 Artificial intelligence	4
1.2 What is Machine Learning, and how does it work ?	5
1.2.1 Data input	5
1.2.2 Algorithm selection	6
1.2.3 Model training	7
1.2.4 Evaluation and testing	8
1.2.5 Deployment and improvement	8
1.3 Dataset bias and fairness in AI	8
1.3.1 Types of biases	8
1.3.2 Existing fairness metrics	9
1.4 Pattern mining techniques	10
1.4.1 Frequent itemset mining	10
1.4.1.1 Apriori algorithm	10
1.4.1.2 FP-Growth algorithm	10
1.4.2 Sequential pattern mining	11
1.4.2.1 PrefixSpan algorithm	11
1.4.2.2 SPADE algorithm	11
1.4.3 Association rule mining	11
1.4.3.1 Apriori-based algorithms	12
1.4.3.2 FP-Growth-based algorithms	12
1.4.4 Emerging pattern mining	12
1.4.5 Applying pattern mining to fairness and bias detection	12

1.5	Sampling methods for dataset balancing	13
1.5.1	Undersampling techniques	13
1.5.2	Oversampling techniques	13
1.5.3	Hybrid Methods	14
1.6	Existing libraries and their limitations	14
1.6.1	AI Fairness 360 (IBM)	14
1.6.2	Fairlearn (Microsoft)	15
1.6.3	FairML	15
1.6.4	FairMaker	15
1.6.5	What-If Tool (Google)	15
1.6.6	Limitations of existing tools	15
1.6.7	Summary of existing tools and methods	16
1.7	Conclusion	20
2	Approach and tools	21
2.1	Overview	21
2.2	Hardware	21
2.3	Software	22
2.4	The metric to quantify the dataset balance	24
2.4.1	Definition of fairness score	24
2.4.2	Advantages and justification	25
2.4.3	Example calculation	25
2.5	Constraint programming model: formalizing the optimization problem	26
2.5.1	Problem formulation	26
2.5.2	Implementation overview	26
2.6	Dataset adjustment: Implementing the solution found	27
2.6.1	Undersampling	27
2.6.2	Synthetic data generation for fairness optimization(oversampling)	27
2.7	Example scenario	29
2.7.1	Dataset	29
2.7.2	Undersampling Case (coefficient = 0.75)	29
2.7.2.1	Step 1: Mark sensitive attributes and get the coefficient	29
2.7.2.2	Step 2: Transform into transaction collection and apply FP-Growth	29
2.7.2.3	Step 3: Generate combinations of sensitive attributes	29
2.7.2.4	Step 4: ConstraintGenerator creates a CP-SAT model	30
2.7.2.5	Step 5: ConstraintSolver solves the CP-SAT model	30
2.7.2.6	Step 6: Evaluation of final fairness	30
2.7.3	Oversampling case (coefficient = 1.25)	30
2.7.3.1	Steps 1-4	31
2.7.3.2	Step 5: ConstraintSolver solves the CP-SAT model	31
2.7.3.3	Step 6: Synthetic Data Generation	31
2.7.3.4	Step 7: Evaluation of final fairness	31
2.8	Web application	32
2.9	UML diagrams	33
2.9.1	Sequence Diagram	33

2.9.2	Use Case Diagram	33
2.9.3	Class Diagram	34
3	Experimental results and discussion	36
	Introduction	36
3.1	FairnessOptimizer vs FairMaker vs Random Split	36
3.1.1	Dataset Study	36
3.1.2	Results comparison	39
3.2	FairnessOptimizer vs Random Oversampling	46
3.3	FairnessOptimizer vs ADASYN Oversampling	50
3.4	Web Application	51
	Conclusion	54
	Bibliography	55
	Webography	56
	Appendix	57
	Contents	59