MASTER IN COMPUTER SCIENCE PROFESSIONAL FOCUS IN DATA SCIENCE

Assessing Machine Learning Fairness via Dataset Mutation

HERBAY, Germain

Award date:
2022

Awarding institution:
University of Namur

UNIVERSITÉ DE NAMUR

FACULTÉ D'INFORMATIQUE

Assessing Machine Learning Fairness via Dataset Mutation

Germain Herbay

RUE GRANDGAGNAGE, 21 ● B-5000 NAMUR (BELGIUM)

Abstract

HERBAY, GERMAIN. Assessing Machine Learning Fairness via Dataset Mutation. (Under the direction of Gilles Perrouin)

KEYWORDS: Fairness, Machine Learning, Mutation Analysis

Fairness is becoming a major concern in software engineering. As machine learning (ML) systems are increasingly used in critical systems (e.g., recruitment and lending), it is crucial to ensure that decisions computed by such systems do not exhibit unfair behaviour against certain social groups (e.g., those defined by gender, race, and age). Apart from robustness and safety, fairness is therefore an important property that a well-designed software should have. Previous works have been conducted to ensure this property by exposing, diagnosing and mitigating bias in ML systems. Although bias in data is a well-studied topic, software engineering has not yet fully explored its impact. To this end, we propose an approach relying on mutation testing to inject perturbations in the training data and analyse the impact of these perturbations on conventional fairness metrics. To evaluate our approach, we design data mutation techniques and use three popular datasets (i.e., Adult, COMPAS, Bank). The first evaluation reveals that fairness measures highly differ depending on the nature of the datasets and the perturbations used. This suggests that the ML algorithms are very sensitive to injected perturbations in the datasets. The second evaluation to better understand the impact of data distributions on fairness leads to less conclusive results. In summary, our results suggest that mutation analysis represents a potentially useful approach for a further in-depth understanding of fairness in ML systems, but requires further exploration.

L'équité devient une préoccupation majeure en ingénierie logicielle. Les systèmes d'apprentissage automatique (ML) étant de plus en plus utilisés dans des systèmes critiques (ex: le recrutement et l'octroi de prêts), il est crucial de s'assurer que les décisions calculées par ces systèmes ne présentent pas un comportement injuste envers certains groupes sociaux (ex: ceux définis par le genre, la race et l'âge). Outre la robustesse et la sécurité, l'équité est donc une propriété importante que doit posséder un logiciel bien conçu. Des travaux antérieurs ont été menés pour assurer cette propriété en exposant, diagnostiquant et atténuant les biais dans les systèmes ML. Bien que le biais dans les données soit un sujet bien étudié, l'ingénierie logicielle n'a pas encore pleinement exploré son impact. À cette fin, nous proposons une approche reposant sur le test de mutation pour injecter des perturbations dans les données d'entraînement et analyser l'impact de ces perturbations sur les mesures d'équité conventionnelles. Pour évaluer notre approche, nous concevons des techniques de mutation de données et utilisons trois jeux de données populaires (Adult, COMPAS, Bank). La première évaluation révèle que les mesures d'équité diffèrent fortement en fonction de la nature des jeux de données et des perturbations utilisées. Ceci suggère que les algorithmes ML sont très sensibles aux perturbations injectées dans les jeux de données. Une seconde évaluation visant à mieux comprendre l'impact des distributions sur l'équité conduit à des résultats moins concluants. En résumé, nos résultats suggèrent que l'analyse par mutation représente une approche potentiellement utile pour une meilleure compréhension de l'équité dans les systèmes ML, mais nécessite une exploration plus approfondie.

Acknowledgements

This thesis would not have been possible without the support of some people.

First, I would like to express my gratitude to my primary supervisor, Gilles Perrouin, as well as my co-supervisor, Paul Temple, for their noble guidance, and support with full encouragement and enthusiasm.

My sincere thanks also go to SerVal research group (Michail Papadakis, Ezekiel Soremekun) who gave me the opportunity to perform my internship.

Finally, I would like to thank my friends and family who supported me throughout this thesis.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context

With the general use of artificial intelligence (AI) over the past decades, humans are increasingly being replaced in critical systems, such as finance, hiring and criminal justice. As AI algorithms drive these systems, we might think they are objective and free from human biases.
However, they still exhibit unfair behaviours. Consider the following examples, which illustrate a range of causes and effects in people's lives.
Bias in online recruitment tools. In 2015, online retailer Amazon stopped using a recruiting algorithm after discovering gender bias in evaluating applicants for software developers and other technical positions. Actually, Amazon's recruiting tool penalised any resume containing the word "women's" and downgraded women's resumes who attended women's colleges [9].
Bias in criminal justice algorithms. The COMPAS (*Correctional Offender Management Profiling for Alternative Sanctions*) algorithm, used by U.S. courts to assess the recidivism likelihood of a defendant, was found to be biased against African-Americans. The algorithm assigns African-Americans a significantly higher risk of criminal recidivism compared to Caucasian people, resulting in racial bias [22].

Since these applications directly affect people's lives, researchers and engineers must think about their potentially harmful effects when modelling an algorithm or a system. Numerous works have already been proposed to support engineers in addressing the fairness problem. Based on these researches, bias in data appears to be one of the leading underlying causes of unfairness. This can be illustrated in the example of Amazon's recruiting tool, by an algorithm that learned from resumes submitted to the company over ten years, where males' resumes are dominant.
Even though bias in data is a well-studied topic, the fairness research community has not yet fully explored its impact. In particular, the sensitivity of the algorithms to biased data is not yet a central topic in previous research. Each algorithm has its procedure to learn rules and patterns from data examples, resulting in different behaviours when confronted with data biases.

1.2 Research objectives

Therefore, the main objective of this master thesis is to propose a fairness-driven sensitivity analysis of algorithms when potential changes arise in data. We rely on *mutation testing* to automatically inject targeted biases in data and then measure the sensitivity of various ML algorithms to these biases. This master thesis involved the following activities:

- **Develop mutation techniques for fairness assessment**.
  It refers to the definition of data modification operators, which can mimic potential fairness issues/enhancements in the data, such as ethnic minorities/equal representation of women and men.

- **Develop altered data generation techniques for fairness assessment**.

It includes the design of strategies that apply mutation techniques to automatically generate biased and unbiased sets of modified data to be used for fairness assessment.

- **Perform validation**.
  Because both mutation and generation techniques incorporate degrees of randomness, it is essential to confront our fairness-driven sensitivity evaluation of algorithms on several datasets.

1.3 Research methodology

To inject changes into the datasets, generate a set of altered datasets and evaluate the impact of these alterations on the fairness of the algorithms, we follow two scenarios :

- **Blind mutation**.
  This scenario envisions naively modifying datasets and represents the first manner to see data mutation-induced fairness changes without having to prior-analyse distributions from datasets.

- **Distribution-aware mutation**.
  Based on preliminary distributions analysis, this scenario envisions modifying datasets with targeted mutations towards a specific fairness goal. This scenario represents a more promising technique for fairness analysis and improvement with fewer and smarter mutations.

1.4 Thesis organisation

This thesis is structured as follows: Chapter 2 presents the Background with different theoretical concepts needed to understand the thesis. Chapter 3 presents the approach, about the general process to implement our Fairness-driven Mutation Analysis. Chapter 4 describes the subjects of our experiments and introduces the research questions and experimental procedure. Chapter 5 presents the results and answers the research questions. Chapter 6 identifies different threats to the validity of our experiments and analyses. Finally, Chapter 7 concludes with the contributions and perspectives of this work.

2 Background

Our fairness sensitivity analysis approach bridges the topics/areas of machine learning, fairness and mutation testing. These concepts involve distinct communities in computer science, and their interactions remain challenging due to their different theoretical underpinnings and viewpoints. In this section, we briefly introduce the basic concepts of each domain to get an overview of how we can bridge them.

2.1 Machine learning : basic concepts

Machine learning (ML) is an important component of the growing technical fields, lying at the intersection of computer science and statistics and the core of artificial intelligence and data science [12]. From advancing medicine to optimizing business processes, machine learning is rapidly changing the face of science, business, and everyday life. Herein are a few applications where machine learning approach has been adopted :

- *Finance* : creditworthiness prediction [31].
- *Justice* : recidivism risk assessment [22].
- *Hiring* : resume screening [9].

Machine learning addresses the question of how to build algorithms that improve their performance at some task through experience [13]. Therefore, one major task of machine learning is to construct **models** from **datasets**.

A dataset is a collection of n **instances** which are objects of interest generated by some unknown process to be modelled, and characterised by a set of d **features** x_1, x_2...x_d and a **target** y.

Age	Education	...	Relationship	Income
24	Some-college	...	Unmarried	<=50k
65	HS-grad	...	Husband	>50k
43	HS-grad	...	Wife	<=50k
...	...	...	...	...
34	Bachelors	...	Husband	<=50k
43	Masters	...	Husband	>50k
48	HS-grad	...	Husband	>50k
...	...	...	...	...
59	HS-grad	...	Husband	>50k
18	HS-grad	...	Unmarried	<=50k
55	Masters	...	Other-relative	<=50k
...	...	...	...	...

Table 1: Adult Census Income dataset

Table 1 presents an excerpt of the *Adult Census Income* dataset with 48,842 individuals for whom the income is below or above 50k$/year. Here, *Adult Census Income* dataset consists of 48,842 instances described by 14 features (a.k.a

attributes), including age, education, relationship, and income as target value.

A model may be *predictive* to make predictions from past experience or *descriptive* to gain knowledge from the data. They are characterised by their **parameters** θ_0, $\theta_1 ... \theta_d$ (a.k.a *weights*), which are optimized by a **learning algorithm** in order to get a good and useful approximation to the process which generates data. Unlike model parameters, which are optimized and obtained after the learning procedure, **hyperparameters** are parameters that determine the complexity/capacity/architecture of the model and are chosen before learning. Since hyperparameters drive the learning process and affect the ability of the model to generalize for unseen data examples, choosing the optimal set of hyperparameters is a challenging problem.
To address this challenge and perform machine learning, the dataset is split into three datasets:

- The **training dataset** is the sample of data used to perform the learning algorithm and fit the model's parameters.
- The **validation dataset** is the unseen sample of data used to provide an evaluation of a model fit on the training dataset while tuning model hyperparameters.
- The **test dataset** is the unseen sample of data used to provide an evaluation of a final model fit on the training dataset.

Machine learning can be grouped into two main learning tasks: Unsupervised and Supervised Learning, which differ in the information given to the models for training.

Unsupervised learning uses machine learning algorithms on unlabelled datasets to discover hidden patterns or data groupings without the need for human intervention (see Figure 1).

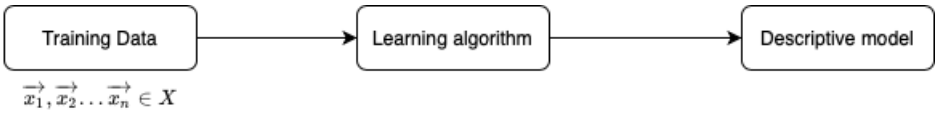


Figure 1: Unsupervised learning

In particular, we can use unsupervised machine learning to :

- Cluster datasets on similarities between features or segment data.
- Understand the relationship between data points such as automated music recommendations.
- Perform initial data analysis.

In **Supervised learning**, labelled datasets are used to train algorithms that classify unseen data into established categories or forecast trends as a predictive model. It is called "supervised" because of the presence of the outcome variable Y (a.k.a *target vector*) to guide the learning process (see Figure 2).

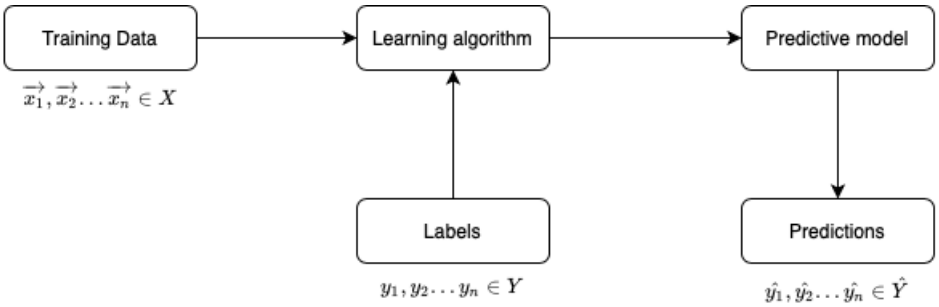


Figure 2: Supervised learning

In supervised learning, there are two categories of tasks: regression and classification.
A **regression** task in machine learning is when a model is used to identify patterns and relationships within a labelled training dataset to predict continuous outcomes. Herein are a few examples of regression tasks:

- Artificial pancreas controller (157.3/.../164.6 (mg/dL blood glucose))
- House price estimation (230,000/.../374,200 ($))
- Global cancer incidence analysis (334.9/.../356 (people per 100,000))

A **classification** task in machine learning is when a model is used to classify whether a given sample belongs to a known group or class. While a regression task seeks to predict a continuous output, classification tasks predict a discrete output. Herein are a few examples of classification tasks:

- Spam filtering (spam/non-spam)
- Automated diagnosis (healthy/ill patient)
- Emotion recognition (neutral/happy/sad/fear)

In this thesis, we focus on classification problems to study the performance and fairness of machine learning algorithms. Before presenting fairness assessment in machine learning, we first need to understand how performance evaluation works.
Performance defines the ability of a model to provide accurate and trustworthy predictions on unseen data. There are several performance metrics for classification problems. In this thesis, we look at common performance metrics which we can compute from the confusion matrix.

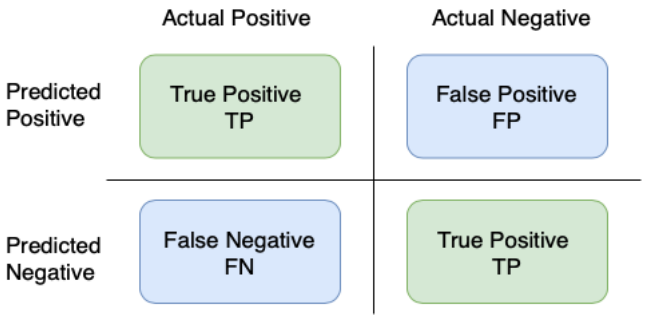


Figure 3: Confusion matrix

A **confusion matrix** is used to define a classification algorithm's performance. Based on the true labels (Y), it gives a summary of the correct and incorrect classifications ($\hat{Y}$) of each class (see Figure 3). For a binary classifier, predicted and actual classes have two values: positive and negative. For instance, positive and negative classes in the Adult Census Income dataset correspond to income over and under 50k$/year respectively.

- **True Positive (TP)**: a case when the predicted and actual outcome are both in the positive class.
  Example: $\hat{y}$ is $>50k\$/year$ and y is $>50k\$/year$
- **False Positive (FP)**: a case predicted to be in the positive class when the actual outcome belongs to the negative class.
  Example: $\hat{y}$ is $>50k\$/year$ and y is $<=50k\$/year$
- **False Negative (FN)**: a case predicted to be in the negative class when the actual outcome belongs to the positive class.
  Example: $\hat{y}$ is $<=50k\$/year$ and y is $>50k\$/year$
- **True Negative (TN)**: a case when the predicted and actual outcome are both in the positive class.
  Example: $\hat{y}$ is $<=50k\$/year$ and y is $<=50k\$/year$

From this matrix, we can derive the following performance metrics :

- **Recall**: $\frac{TP}{(TP+FN)}$
- **Accuracy**: $\frac{TP+TN}{TP+FP+TN+FN}$
- **Precision**: $\frac{TP}{TP+FP}$

2.2 Fairness and bias

This section introduces the concepts of fairness and bias and how they manifest themselves in data.

Machine learning bias occurs when an ML algorithm produces systematically unfair results due to wrong assumptions in the machine learning process. From finance and criminal justice to hiring, biases may arise in many critical systems. Depending on how machine learning systems are used, biases can lead to unfair or possibly illegal actions to even potentially harmful conditions. Such bias issues in these applications are certainly undesirable. As a result, it leads machine learning researchers to consider: how do biases arise in machine learning models?
A central idea of machine learning is that the ML algorithm learns the rules and patterns by itself from data examples. Therefore, its decision-making is really sensitive to the data used for learning. If the data shows some kind of biases, the algorithm may reproduce or amplify the same kind of biases by making decisions on what it's learned. Consequently, biases arise in the model through the data.

Since biases in data can exist in many shapes and forms [20], we focus on two of them: Historical bias and Sample bias.

Historical bias *refers to the already existing bias and socio-technical issues in the world* [5]. It can seep into the data generation process even if the data is perfectly measured and sampled. For instance, consider the COMPAS algorithm. Suppose African-Americans are more likely to be arrested or incarcerated due to historical racism or disparities in policing practices. In that case, these realities will be mirrored in the training data used by the COMPAS algorithm to predict whether a defendant will be a recidivist or not. As a result, the prejudices of the US criminal system will lead the model to make the same wrong decisions.
Further, historical bias can be reinforced by the machine learning model itself. For example, if a predictive recidivism model determines an area of the city to be high-risk, more police officers will be deployed to that neighbourhood. Then, the officers will prioritise this area, increasing the chance of arrest. The risk of this area will therefore increase in a vicious feedback loop. The model will be validated and retrained on data containing even more arrests in this area. As a result, instead of becoming less detrimental, the model will be stuck in a negative feedback loop and become increasingly biased over time. Such a negative feedback loop can lead to over-/under-representation of a population and reinforce the following source of bias, sampling bias.

Sampling bias *occurs when data examples are not representative of the population due to non-random sampling of subgroups* [20]. Some members of a population are systematically more likely to be selected than others, leading to imbalanced data. As a result, the estimated trends for one population may not generalise to data collected from a new population. For example, consider an elaborated dataset of photographs of humans of all ethnicities, without bias towards any particular ethnicity. Suppose a specific face recognition system is

mainly trained on pictures of white men. In that case, it won't perform well when identifying women and people of diverse ethnicities, even if the collected data was not initially biased.

Many fairness definitions have been proposed to address such sources of algorithmic bias. Before explaining some of the definitions used for fairness in algorithmic classification problems, we need to understand the theoretical concepts of fairness in the context of machine learning.
Machine learning fairness is defined in terms of **protected attributes** and **privileged/unprivileged groups**. Protected attributes refer to the sensitive characteristics/features that need to be protected against unfairness. For instance, age, sex, race, religion, country of origin, marital status, etc., are typical protected attributes. Protected attributes divide the population of a dataset into privileged and unprivileged groups, where the privileged population group would be more likely to receive favourable treatment than the unprivileged population group. For instance, the sex attribute in the Adult Census Income dataset is a protected attribute that splits the population into male/female as privileged/unprivileged groups. Indeed, it is assumed that the Adult Census Income predictive classification model will favour the male group over the female group in its income prediction (i.e., predicting a higher income for males than females).

Based on these concepts, different definitions of fairness have been proposed in the literature. They are two main types of fairness definitions: group fairness and individual fairness.
A model has **group fairness** *with regard to an input characteristic when the privileged and unprivileged groups are treated equally (e.g., the distribution of prediction outcomes for each group is similar)*[10]. In this thesis, we focus on two popular definitions for group fairness: SPD [16][4][19] and EOD[17]. We define them via the predicted outcome of the classifier $\hat{Y}$, where $\hat{Y}$=1 is the desired predicted outcome, and the protected attribute A, where A=1 is the privileged group and A=0 is the unprivileged group.

Definition 2.1 (Stastical Parity Difference)

- ***SPD*** $= |P(\hat{Y}=1|A=0)|$ - $|P(\hat{Y}=1|A=1)|$

Statistical Parity Difference is the difference in probability of being assigned to the positive predicted class for the privileged and unprivileged groups. In the Adult Census Income algorithm, the classifier would be fair regarding the gender attribute if men and women would have the same probability of being labelled as making more than 50k$ a year (i.e, SPD = 0).

Definition 2.2 (Equal Opportunity Difference)

- ***EOD*** $= |P(\hat{Y}=1|A=0,Y=1)|$ - $|P(\hat{Y}=1|A=1,Y=1)|$

Equal Opportunity Difference is the difference of probability for a person in a positive class being assigned to the positive predicted class for the privileged and unprivileged group. In other words, the equal opportunity definition is the difference of true positive rates for the protected and unprotected groups.

However, group fairness may fail to observe some discrimination by ignoring all attributes of the classified subject except the sensitive attribute. In the Adult Census Income algorithm, male and female individuals could have the same probability of being assigned a positive outcome, but some individuals who differ only in gender could get a different outcome. Even if the classifier satisfies demographic parity, it may still produce gender discrimination.
Individual fairness addresses this problem by not marginalising insensitive attributes and ensuring that any two individuals who only differ in the protected attributes should be classified similarly [28].
However, if group and individual fairness seem desirable in a system, they cannot be satisfied simultaneously. Therefore, we decide to only focus on the previous definitions of group fairness since they are widely adopted in the literature [8][11][23]. We can also automatically derive them from the confusion matrix.

2.3 Bias mitigation methods

Many attempts have been proposed to avoid model bias and ensure model fairness. These methods fall under three categories:

- **Pre-processing** methods aim at modifying the training data to reduce bias in the data.
- **In-processing** methods aim to change the learning procedure to reduce training bias.
- **Post-processing** methods aim at changing the prediction outcomes to mitigate the bias of a learned model.

Like pre-processing algorithms, our fairness testing approach aims to modify the training data to evaluate fairness in machine learning algorithms. In this section, we present the main methods in pre-processing algorithms.
Reweighing (RW) applies weights to different groups in the training data to make a discrimination-free training dataset without having to change any of the labels [4].
Optimised pre-processing (OP) learns a probabilistic transformation that edits the features and labels in the data with discrimination control to limit the dependence of the modified labels on sensitive attributes, distortion control to avoid some important changes (e.g., a very low credit score is mapped to a very high credit score in a loan approval decision algorithm), and utility preservation to ensure that the distributions of the modified features and labels are close to the original distributions of features and labels [3].
Learning fair representations (LFR) encodes data into an intermediate representation to lose any information that can identify whether an individual belongs to a protected subgroup while preserving as much information as possible about the individual's attributes[24].
Disparate impact remover (DIR) edits values of non-protected attributes to make the distributions for the unprivileged and privileged subgroups close to each other and increase fairness [16].
As illustrated in Figure 4, **Fair-SMOTE** rebalances internal distributions such that they are equal, based on class and sensitive attributes, to improve fairness [7].

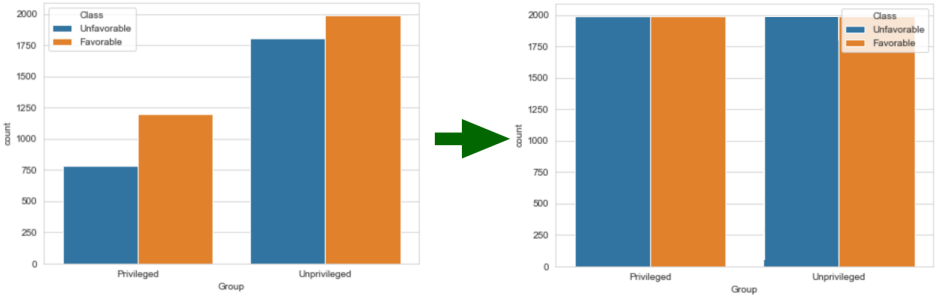


Figure 4: Fair-SMOTE

Based on the previous techniques, we adapt the training data mutation perspective to evaluate fairness in ML algorithms.

2.4 Fairness testing

Even though developing a fairer machine learning model is a start, it cannot be the entire solution. Apart from robustness and safety, fairness is also an important property that a well-designed software should have. *Good design and proper algorithms are important, but so are quality control via, e.g., testing and formal verification* [32]. Therefore, several techniques have been proposed to conduct fairness testing.

Themis is an automated test suite generator that randomly samples the value of all attributes from the corresponding domain to determine whether a given model prejudices individuals [25][2]. As Themis generates tests in a full random way, it suffers from an absence of any systematic test case generation technique.

Unlike Themis, **Aequitas** can be used as a directed test generation module to uncover discriminatory inputs. Aequitas improves Themis by adopting a two-phase generation framework combining a global and local search. In the first step, it randomly samples the input space to discover the presence of discriminatory inputs. In a second step, Aequitas searches in the neighbourhood of the resulting discriminatory inputs to generate further inputs with similar characteristics [27].

Even though Themis and Aequitas apply to any black-box system, they miss many combinations of non-protected attribute values for the individual discrimination that may exist. **Symbolic Generation** (SG) attempts to solve this problem by generating a local explanation decision tree for approximating the machine learning model and then performing symbolic execution based on the decision tree to generate test cases automatically [1].

Since existing approaches are developed mostly for traditional machine learning models, **Adversarial Discrimination Finder** (ADF) is proposed as a scalable gradient-based approach for searching individual discriminatory instances of Deep Neural Network (DNN) [21]. It generates individual discriminatory instances through adversarial sampling with a two-phase generation framework. ADF aims to identify individual discriminatory instances during global generation by iteratively perturbing the seed inputs towards the decision boundary

from the original dataset. During local generation, ADF generates as many individual discriminatory instances as possible by following the intuition that instances nearby the seed data are likely to be individual discriminatory instances.

Unlike previous fairness testing techniques, **Fairea** is proposed as a first mutation analysis approach for fairness evaluation targeting machine learning software [18]. It applies mutation on the labels to compose a baseline for evaluating bias mitigation methods.

Like Fairea, we want to adapt the mutation testing approach to fairness evaluation of machine learning models with a data perspective. However, instead of mutating the labels, we want to mutate all dataset to assess how data mutation can impact the fairness of models.

2.5 Mutation testing

Mutation testing is a fault-based testing technique where software program variations are subjected to the test set. Since software programmed by developers could be a major source of defect introduction, mutation testing introduces small faults (e.g., replacing the '+' operator in the program to the '-' operator) into the source code to check whether the defined test cases can detect the resulting errors when running the software. The key process of general mutation testing is illustrated in Figure 5.

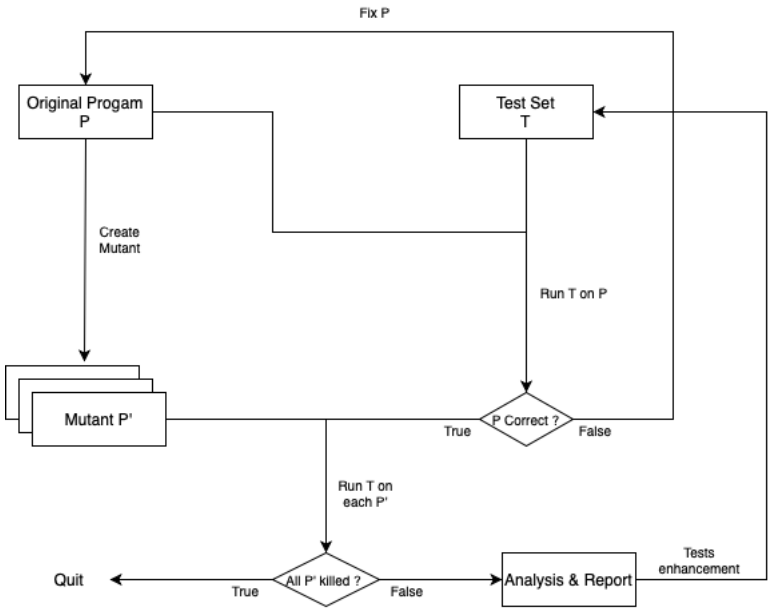


Figure 5: Classical mutation testing approach

Based on an original program P, a set of faulty programs $P'_1, P'_2, \ldots, P'_n \in P'$ (**mutants**) are created based on predefined rules (**mutation operators**), each of which slightly modifies the syntax of the program under analysis P

[15]. In this case, mutants P' represent potential mistakes that programmers could make, and mutation operators represent transformation rules designed to modify variables and expressions by replacement, insertion or deletion. Before starting the mutation analysis, a step of pre-processing is used to check the correctness of the original program P for the test case. If P is incorrect, it has to be fixed before running other mutants; otherwise, each mutant $P'_i \in P'$ will then be run against this test suite $T_1, T_2, \ldots, T_n \in T$. Afterwards, the complete test set T is run against each mutant $P'_i \in P'$. If the result of running $P'_i \in P'$ is different from the result of P for any test case $T_i \in T$, then the mutant P'_i is said to be "killed", otherwise it is said to have "survived". When all the mutants P' have been tested against T, a **mutation score** is calculated as the ratio of the number of mutants killed over the total number of mutants. In essence, the mutation score indicates the quality of the input test set by its degree of achievement in fulfilling the test objectives (design test cases that kill all the mutants). After all test cases have been executed, "surviving" mutants may remain. The developer can further enhance the quality of the test set by adding more tests to kill these surviving mutants.
The general purpose of mutation testing is to raise the mutation score to evaluate the quality of a test set, provide feedback and guide the test enhancement.

2.5.1 Mutation testing for machine learning

As mutation testing has empirically proven to be an effective software testing method [26], it has recently been extended to machine learning systems. Ma et al. [15] consider that a higher quality test dataset would provide more comprehensive feedback and guidance for further in-depth understanding of machine learning systems. Therefore, they propose a mutation testing framework to measure the quality of the test dataset. They adapt mutation testing techniques to deep learning software development in two approaches: source-level and model-level mutation testing (see Figures 6 and 7).

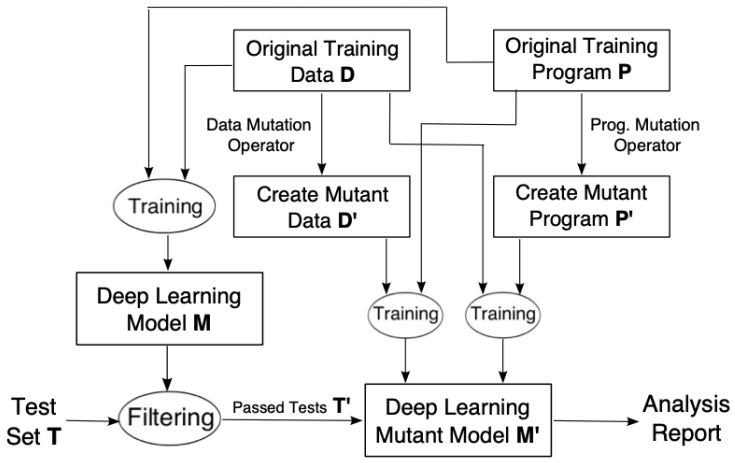


Figure 6: Source-level mutation testing workflow of DL systems

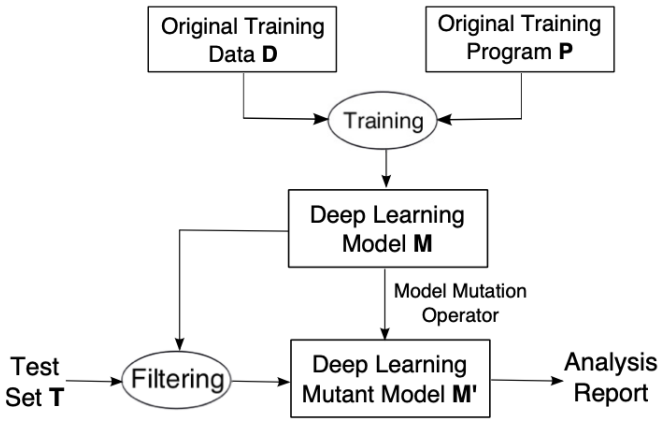


Figure 7: Model-level mutation testing workflow of DL systems

In **source-level mutation testing** approaches, one targets training data or training program to inject faults (e.g., add random perturbations to some pixels of an image) that could be potentially introduced during the learning process. After mutating sources of a deep learning system, which are the training dataset and the learning algorithm, they obtain a mutated model M'. Then, they execute and analyse M' against the filtered test set T' to evaluate the quality of the test dataset. Note that T' are the test data points in T that the original deep learning model M correctly processes. In **model-level mutation testing** approach, they follow the same workflow. However, instead of mutating the sources of the deep learning systems, they directly inject faults (e.g., switching two neurons within a layer to exchange their roles and influences for the next layers) into the deep learning algorithm.

After obtaining a set of mutant deep learning models M', each test data point t' $\in$ T' that the original deep learning model M correctly handles is evaluated on the set of mutant models M'. Afterwards, they say that test data T' kill mutated model m' if there exists a test input t' $\in$ T' that m' does not correctly handle.

In the next section, we want to adapt the source-level mutation testing workflow proposed by Ma et al. [15] to build a mutation analysis approach for fairness evaluation targeting machine learning software.

3 Approach

In this section, we introduce the main steps of our approach to evaluate the fairness sensitivity of machine learning algorithms.

Based on the first assumption that fairness issues do not reside in the ML code but are strongly linked to problems in the data, we want to adapt the perspective of the mutation testing approach to assess fairness in machine learning algorithms from a data perspective. In such a vision, we consider modifying the original dataset, training the ML algorithms on them and analysing the difference with the originally trained algorithm. This vision allows us to evaluate the sensitivity of ML algorithms to several kinds of injected fairness issues. The overall process is shown in Figure 8.

There are three primary steps in our approach to evaluating ML algorithms.

Step 1: Baseline creation
First, we create a baseline by training an ML algorithm on an original training set (1), yielding an ML model and a set of performance and fairness scores (2).

Step 2: Mutant creation
Second, we apply data mutation operators to inject perturbations in the original training dataset and create a set of mutated training datasets (3). Next, we train the same ML algorithm (4) (with the same hyper-parameters as the original) on each modified training dataset yielding a set of ML model variants and a set of performance and fairness scores (5). Note that the ML model variants and the original ML model are tested against the same test dataset.

Step 3: Sensitivity analysis
Third, we quantify the sensitivity of the ML algorithm by analysing dispersion between the scores of the original ML model and ML model variants (6).

Through this approach, we aim to provide a tool for analysing the sensitivity of ML algorithms to data mutation and determine whether algorithms are more sensitive than others. The following section explains how these steps are implemented to achieve our research objectives.

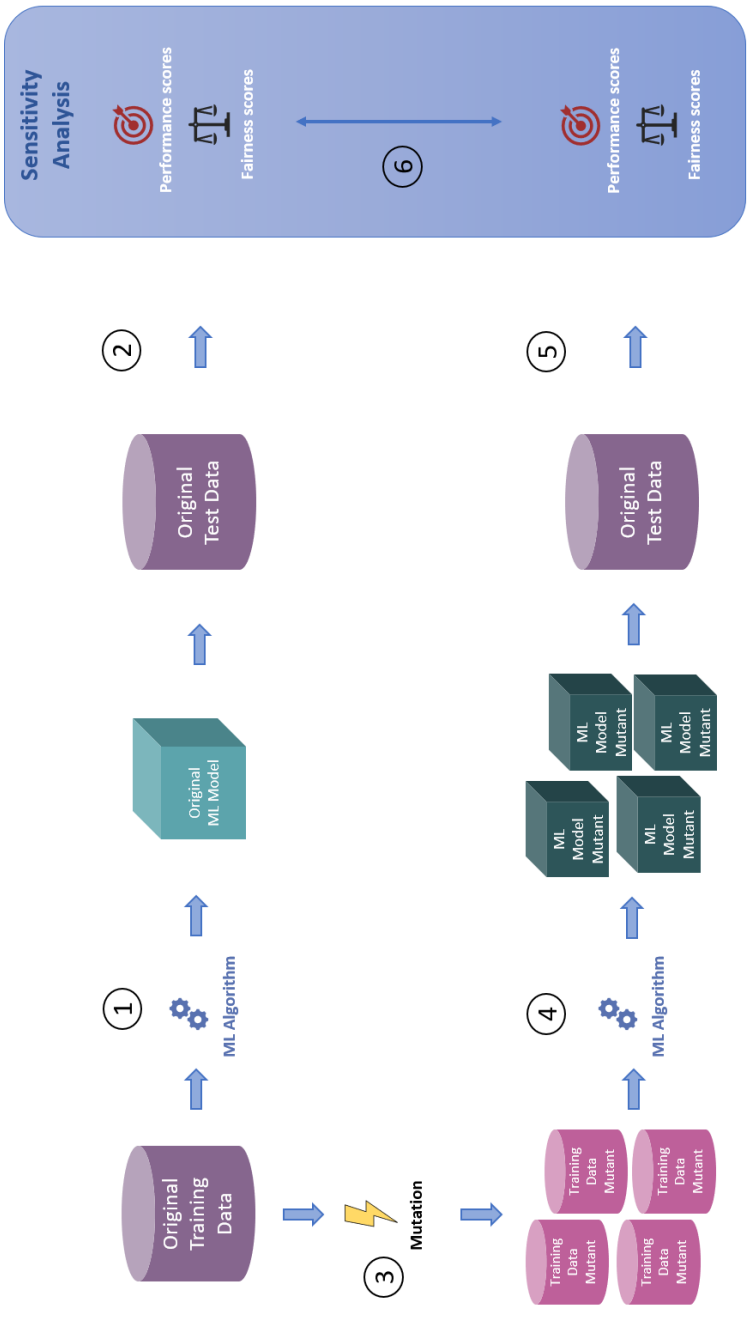


Figure 8: Fairness-driven Mutation Analysis Process

4 Experimental setup

In this section, we describe our experiment design and setup. We first describe the subjects of our experiments and then introduce our research questions and experimental procedure.

4.1 Experimental subjects

4.1.1 Candidate algorithms

We select four machine learning classifiers to explore the sensitivity of ML algorithm to data mutation:

- Logistic Regression (LR)
- Random Forest (RF)
- Support-Vector Machines (SVC)
- Multi-Layer Perceptron (MLPC)

As in previous work [7], we use the standard implementation found in Scikit-learn[1] machine learning library with the default configuration for each classifier.

4.1.2 Datasets

The experiments are conducted on three popular datasets, which are commonly used in the literature of machine learning fairness research:

- Adult Census [30]
- COMPAS [22]
- Bank Marketing [14]

To be able to compare the performance and fairness of our candidate algorithms in each dataset, we use the same data preprocessing strategy: the missing and invalid values are removed, and categorical attributes are encoded by using *LabelEncoder*[2] transformer and each attribute is scaled between zero and one by using *MinMaxScaler*[3]. We specify the protected attributes, the privileged and unprivileged group and the favourable predicted label for fairness analysis. Finally, each dataset uses the same train-test splitting (80% - 20%)[4].

[1] `https://scikit-learn.org/`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

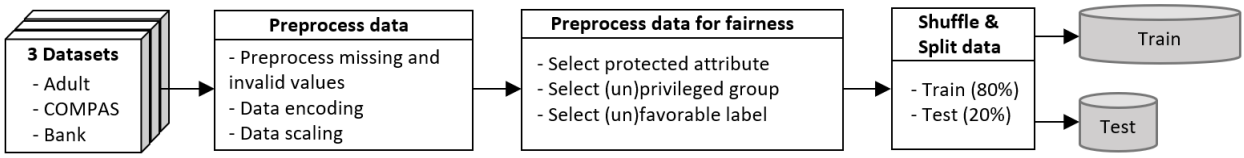


Figure 9: Data preprocessing

ADULT CENSUS INCOME:

The **Adult Census Income** dataset contains financial and demographic information about individuals from the 1994 U.S. Census. Based on census data, the classification task is to predict whether a given adult makes over $50k a year.

Dataset characteristics: The dataset consists of 48,842 instances described via 15 attributes (6 are numerical, 7 are categorical and 2 are binary). We provide an overview of attribute characteristics in Table 2. As suggested in previous work [29], we exclude the attribute *fnlwgt* from our experiments.

Missing values are present in 3,620 instances. As they can bias the results of the machine learning models, we remove them to obtain a clean dataset with 45,222 instances.

Attributes	Type	Values	#Missing values	Description
age	Numerical	[17 - 90]	0	The age of the individual
workclass	Categorical	7	2,799	The career status
fnlwgt	Numerical	[13,492 - 1,490,400]	0	The final weigth
education	Categorical	16	0	The education level
education-num	Numerical	[1 - 16]	0	The education level in numerical form
marital-status	Categorical	7	0	The marital status
occupation	Categorical	14	2,809	The general type of occupation
relationship	Categorical	6	0	The relationship with others
race	Categorical	5	0	Race
sex	Binary	{Male, Female}	9	The gender
capital-gain	Numerical	[0 - 99,999]	0	The capital gains
captal-loss	Numerical	[0 - 4,356]	0	The capital loss for an individual
hours-per-week	Numerical	[1 - 99]	0	The hours an individual has reported to work
native-country	Categorical	41	857	The country of origin for an individual
income	Binary	{<=50K, >50K}	0	Whether or not an individual makes more than 50K$ a year

Table 2: Adult: attributes characteristics

Protected attributes:

- **Sex**, where *male* is the privileged group and *female* is the unprivileged group.
  The ratio of male and female is **67.5 - 32.5**(%).

Class attribute: The class attribute is *income* $\in$ {<=50K, >50K}indicating whether the individual makes more than 50K$ a year, where *>50K* is the desired predicted outcome.
The ratio of <=50K and >50K is **75.2 - 24.8**(%).

COMPAS:

The **COMPAS** (Correctional Offender Management Profiling for Alternative Sanctions) contains criminal history, jail, prison time and demographic information about defendants in Broward County, Florida. The classification task is to predict the likelihood of a defendant to become a recidivist.

Dataset characteristics: The dataset consists of 7,214 instances described via 51 attributes, as shown in Table 11 .

Missing values are present in 6,395 instances. Based on this work [6], we clean the dataset by removing missing data, such as the violent recidivism attribute that contains only *NaN* values or charge date cases that did not occur within 30 days of a COMPAS assessment. Finally, we consider a subset of the COMPAS attributes previously used for fairness experiments [19], which comprises five features (2 are categorical, 2 are binary and 1 is numerical). We provide an overview of its attribute characteristics in Table 3.

Attributes	Type	Values	#Missing values	Description
age_cat	Categorical	3	0	Age in a categorical form
race	Categorical	6	0	Race
priors_count	Numerical	[0 - 38]	0	The prior offenses count
c_charge_degree	Binary	{F, M}	0	Charge degree of original crime
two_year_recid	Binary	{0, 1}	0	Whether the defendant is rearrested within two years

Table 3: Subset of COMPAS: attributes characteristics

Protected attributes:

- **Race**, where *caucassian* is the privileged group and *non-caucassian* is the unprivileged group.
  The ratio of caucassian and non-caucassian is **34.1 - 65.9**(%).

Class attribute: The class attribute is *two_year_recid* $\in \{0, 1\}$ indicating whether the defendant re-offends within two years, where *0* is the desired predicted outcome.
The ratio of 0 and 1 is **54.5 - 45.5**(%).

BANK MARKETING:

The **bank** dataset contains data from a bank's marketing campaign to Portuguese individuals. The classification task is to predict whether a given client will subscribe to a term deposit.

Dataset characteristics: The dataset consists of 11,162 instances described via 17 attributes, as presented in Table 4.

Attributes	Type	Values	#Missing values	Description
age	Numerical	[18 - 95]	0	The age of the client
job	Categorical	12	0	The type of job
marital	Categorical	3	0	The marital status
education	Categorical	4	0	The education level
default	Binary	{Yes, No}	0	Whether or not the client has credit default
balance	Numerical	[-8,019 - 102,127]	0	The balance of the client account
housing	Binary	{Yes, No}	0	Whether or not the client has a housing loan
loan	Binary	{Yes, No}	0	Whether or not the client has a personal loan
contact	Categorical	3	0	The contact communication type
day	Numerical	[1 - 31]	0	The last contact day of the week
month	Categorical	12	0	The last contact month of the year
duration	Numerical	[0 - 4,912]	0	The last contact duration (in seconds)
campaign	Numerical	[1 - 63]	0	The number of contacts performed during this campaign and for the client
pdays	Numerical	[1 - 871]	0	The number of days that have passed since the last contact with the client
previous	Numerical	[0 - 275]	0	The number of contacts performed before this campaign and for the client
poutcome	Categorical	4	0	The outcome of the previous marketing campaign
y	Binary	{Yes, No}	0	Whether or not the client has subscribed a term deposit

Table 4: Bank: attributes characteristics

Protected attributes:

- **Age**, where people, who are between the age of 25 to 60 years old, are in the privileged group and people, who are less than 25 or more than 60 years old, are in the unprivileged group.
  The ratio of 0 and 1 is **54.5 - 45.5**(%).

Class attribute: The class attribute is $y \in$ {Yes, No} presenting whether a customer will subscribe to a term deposit or not, where *Yes* is the desired predicted outcome.
The ratio of Yes and No is **54.5 - 45.5**(%).

4.1.3 Data Imbalance

Chakraborty et al.[7] postulate that the historical and sampling biases underlying these datasets, which result in imbalanced data, could be the cause of unfairness.
As presented in Figure 10, these datasets have a **class imbalance** (i.e, the target class has an uneven distribution of observations). Further, the number of observations per group is not equally distributed.

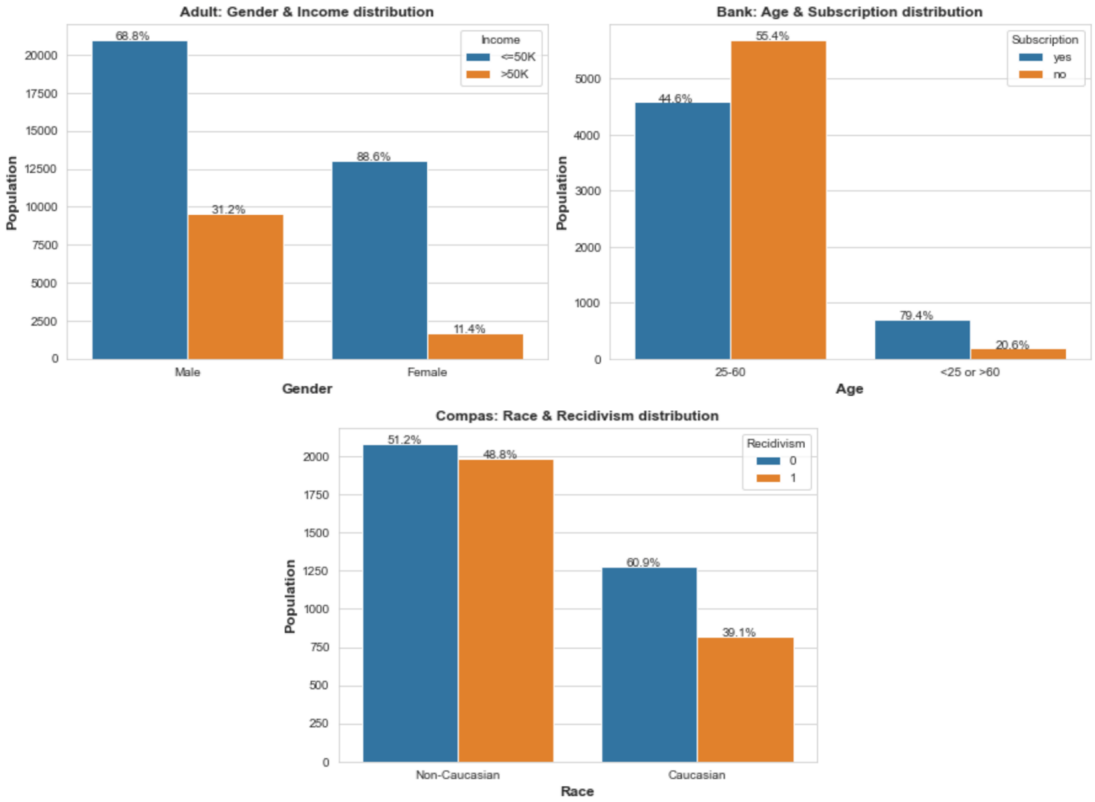


Figure 10: Datasets distributions

For instance, the Adult dataset is highly dominated by people earning less than 50K$ per year but also by white males (see Figure 10). Such imbalance can be critical as classification models trained on Adult may have difficulty predicting the minority class or group. We need to consider datasets' imbalances during our data mutation process to critically assess ML algorithms' sensitivity.

4.2 Research questions

The evaluation answers the following research questions.

RQ1: How sensitive are machine learning algorithms to mutations in the training data regarding fairness?
We believe that fairness issues do not reside in the ML code but are strongly linked to issues in the training data. Therefore, our purpose is to investigate whether the fairness of an ML model is sensitive to mutations in the training data, which can mimic certain biases such as the inadequate representation of women or ethnic minorities. If so, it means ML algorithms acquire bias from the training data, and we can already remove that bias by modifying the training data before model training.

RQ1.1: What is the impact of a few injected perturbations in the training data on the fairness scores of a machine learning model?
To answer this first research sub-question, we first ask whether modifying the training data (via mutations) can already lead to differences in the fairness scores of an ML model. That is an important question because if confirmed, then the chances of bias in the training data affecting the final decision are high.

RQ1.2: Which machine learning algorithms are the most sensitive (or insensitive) to mutations in the training data regarding fairness?
We also ask whether some ML algorithms are more prone to data changes. Each ML algorithm has its procedure to learn rules and patterns from data examples. As a result, they might behave differently when bias occurs in the training data.

RQ2: What targeted mutations in training data can improve the fairness of a machine learning model?
Our assumption is that we can remove bias by directly modifying the training data. Therefore, RQ2 investigates some targeted data-mutation techniques to improve the fairness of ML models.

RQ2.1: Can having the same proportions for each group (privileged/unprivileged) in each class reduce the bias?
In section 2, we introduced Fair-SMOTE. This tool has already shown fairness improvement. Fair-Smote makes proportions equal based on class and protected attributes by rebalancing internal distributions. Similarly, we ask whether modifying the training data without creating new instances can significantly reduce bias.

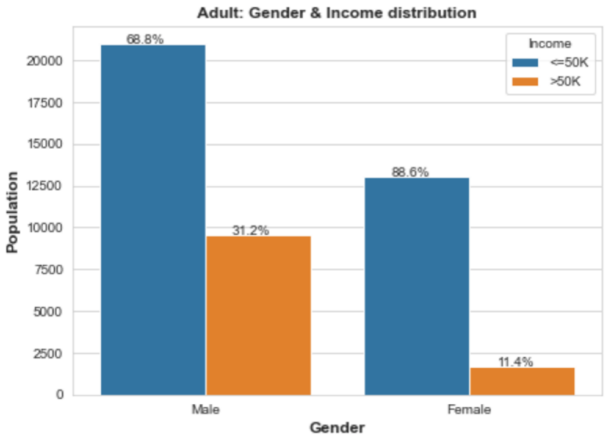


Figure 11: Adult: Gender and Income distributions

The proportion of males in the high-income class is three times higher than females showing an under-representation of highly paid women (see Figure 11). In such a vision, we want to investigate whether having the same proportion of females and males in each *Income* class can also improve fairness.

RQ2.2: Can making proportions equal based on class, protected attributes and non-protected attributes reduce bias?

Although modifying the training data on the protected attribute can improve fairness, bias may remain in other parts of the training data. Fair-SMOTE only modifies distributions of the protected attributes. However, the initial over-representation is also found in other attributes that affect the final decision. For instance, it can be seen with the distribution for each group of the "hours-per-week" attributes in the Adult dataset (see Figure 12).

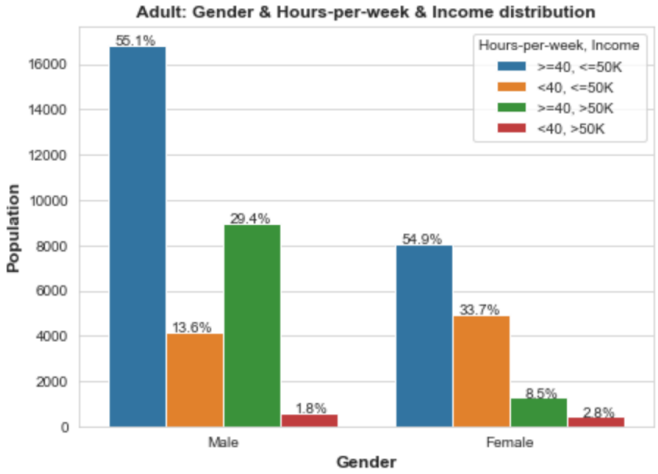


Figure 12: Adult: Gender, Hours-per-week and Income distributions

Highly paid people are more likely to work at least 40h/week. However, we note that the proportion of males working a lot in the high-income class is three times higher than that of females showing an under-representation of highly paid, high-working women (see Figure 12). In such a vision, we want to investigate whether having the same proportion of females and males in each pair *Hours-per-week-Income* can further improve fairness than the last targeted mutation strategy.

4.3 Experimentation protocol

We answer the research questions through our fairness-driven mutation analysis approach (see Section 3). Depending on the research question, we use a specific mutation scenario. Therefore, our research procedure is divided into two scenarios:

1. Blind mutation;
2. Distribution-aware mutation.

In the first scenario, we aim to answer our first research questions by naively modifying the original training datasets. This approach represents the first

way to see data mutation-induced fairness changes without analysing distributions from the dataset regarding fairness. However, one problem that prevents blind mutation from becoming a practical mutation analysis technique is the enormous number of possible mutated training datasets to train and analyse. Furthermore, some mutations may be useless or harmful since they may modify a feature that does not need to be changed.

Due to the previous drawbacks, we also offer a more effective mutation scenario to answer our second research question. We propose a distribution-aware approach for more targeted mutation towards a specific fairness goal. Unlike the blind mutation approach, we need to analyse and learn from the original dataset distributions to generate the modified training datasets. Although this approach requires a preliminary dataset analysis, we postulate that it represents a more promising technique for fairness analysis and improvement with fewer and smarter mutations.

In this section, we explain the steps involved in the baseline creation and altered datasets creation for each scenario. The results and sensitivity analysis step for each scenario are presented in Section 5 to answer the research questions.

4.4 Blind mutation

4.4.1 Baseline

We first create a baseline by computing the performance and fairness scores of the trained algorithms on the original training datasets. Because of the nature of the learning algorithms, we may obtain different results by running the same algorithm on the same training data. We train our candidate algorithms ten times to get a baseline and average the results.
The original scores are shown in Table 5.

Dataset-Prot.Attr.	Learning Algorithm	Recall	Accuracy	Precision	EOD	SPD
Adult - Sex	Logistic Regression	0.47	0.82	0.71	0.3	0.19
	Random Forest	0.61	0.84	0.69	0.02	0.17
	SVC	0.56	0.84	0.75	0.16	0.18
	MLPC	0.61	0.84	0.72	0.13	0.19
COMPAS - Race	Logistic Regression	0.82	0.68	0.67	0.11	0.18
	Random Forest	0.74	0.67	0.68	0.15	0.25
	SVC	0.73	0.66	0.68	0.08	0.17
	MLPC	0.72	0.68	0.71	0.15	0.24
Bank - Age	Logistic Regression	0.82	0.82	0.79	0.19	0.5
	Random Forest	0.86	0.84	0.81	0.11	0.42
	SVC	0.82	0.82	0.8	0.19	0.5
	MLPC	0.85	0.83	0.8	0.14	0.44

Table 5: Baseline - Blind mutation: original scores

4.4.2 Mutation

Second, we create our mutated training datasets to evaluate the score of the algorithms on modified data. To generate these mutants, we design **mutation operators** and a **selection strategy**.
Both are described on the Adult dataset.

Mutation Operators. Similar to mutation testing for traditional software, our mutation operators represent predefined rules, each of which modifies the original training data to create mutated training datasets. As we drive our mutation on the data, we design the following operators to introduce potential fairness changes into the source of the ML algorithms: the training data.

Column Change (CC) operator changes a data point (i, c) at index i and column c with another value in the range of possible values of the column. Figure 13 illustrates CC operator in the Adult dataset by changing the value 'Husband' at index 10,000 in the 'Relationship' column to a new random value:' Unmarried'.

	Age	Education	...	Relationship	Income
0	24	Some-college	...	Unmarried	<=50k
1	65	HS-grad	...	Husband	>50k
2	43	HS-grad	...	Wife	<=50k
...	...	...	...	...	...
10,000	34	Bachelors	...	Husband	<=50k
10,001	43	Masters	...	Husband	>50k
10,002	48	HS-grad	...	Husband	>50k
...	...	...	...	...	...
30,000	59	HS-grad	...	Husband	>50k
30,001	18	HS-grad	...	Unmarried	<=50k
30,002	55	Masters	...	Other-relative	<=50k
...	...	...	...	...	...

{Not-in-Family, Husband, Unmarried, Own-child, Wife, Other-relative} => Unmarried

Figure 13: Column Change(CC) operator

Column Shuffle (CS) operator shuffles the values at indexes I within a column c into different orders. Figure 14 illustrates CS operator in the Adult dataset by shuffling the values at indexes 10,000, 30,001, 30,002 within the 'Relationship' column into different orders.

	Age	Education	...	Relationship	Income
0	24	Some-college	...	Unmarried	<=50k
1	65	HS-grad	...	Husband	>50k
2	43	HS-grad	...	Wife	<=50k
...	...	...	...	...	...
10,000	34	Bachelors	...	Husband	<=50k
10,001	43	Masters	...	Husband	>50k
10,002	48	HS-grad	...	Husband	>50k
...	...	...	...	...	...
30,000	59	HS-grad	...	Husband	>50k
30,001	18	HS-grad	...	Unmarried	<=50k
30,002	55	Masters	...	Other-relative	<=50k
...	...	...	...	...	...

I : {10,000 , 30,001 , 30,002}

Figure 14: Column Shuffle(CC) operator

Selection strategy Once we have our mutation operators, we have to determine on which part of the training data to apply them. Therefore, we use a splitting strategy allowing us to separate the training data into two parts :

- One to be mutated
- One left unchanged

As we are aware of the data imbalances within the datasets, we decide to use a semi-blind splitting strategy: *StratifiedShuffleSplit*[5]. It allows us to remain random in choosing which indexes to mutate while ensuring that each class is mutated.

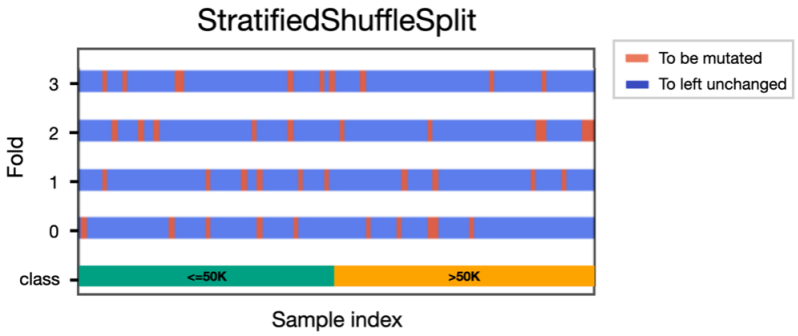


Figure 15: Selection strategy: stratifieShuffleSplit

For the Adult dataset, *StratifiedShuffleSplit* (see Figure 15) randomly selects the instances to be mutated and ensures that these instances preserve the same percentage of individuals from the population earning more than \$50K/year and from the population earning less than \$50K/year.

Mutant generation Based on our selection strategy, we use four channels to generate our mutated training datasets: **5%-10%-15%-20%** training data to be mutated. Each channel represents the percentage of mutated instances in the training dataset. As we want to evaluate the impact of small mutations in the training dataset on the fairness scores of a machine learning model, we do not modify beyond 20% the training dataset.

For each channel, we have five index folds where we apply mutation operators. We apply one mutation operator for each index fold on only one attribute at a time.
This leads us to create :

- 480 training data mutants for the Adult dataset
- 200 training data mutants for the COMPAS dataset
- 640 training data mutants for the Bank dataset

Then, we train the candidate learning algorithms on each mutated training dataset and compute the performance and fairness scores of the resulting model variants.

[5]`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html`

4.5 Distribution-aware mutation

Before starting the distribution-aware scenario, we slightly adapt the preprocessing of the datasets to ease our mutations. Directing our mutations to specific parts of the dataset, we discretise some numerical attributes and also transform some categorical attributes to conduct our experiments. For Adult, we change *relationship* and *hours-per-week* as follows : *relationship*={Married, Other}, *hours-per-week*={>=40, <40}. For COMPAS, we transform *priors-count* into *priors-count*={0, 1-5, >5}

4.5.1 Baseline

As our original training datasets are different from the last mutation scenario, we create a new baseline shown in Table 6.

Dataset-Prot.Attr.	Learning Algorithm	Recall	Accuracy	Precision	EOD	SPD
Adult - Sex	Logistic Regression	0.56	0.84	0.72	0.23	0.21
	Random Forest	0.59	0.83	0.7	0.07	0.18
	SVC	0.56	0.85	0.76	0.13	0.18
	MLPC	0.59	0.85	0.75	0.1	0.18
COMPAS - Race	Logistic Regression	0.76	0.64	0.64	0.03	0.1
	Random Forest	0.81	0.66	0.64	0.05	0.11
	SVC	0.83	0.65	0.63	0.07	0.13
	MLPC	0.8	0.66	0.64	0.06	0.12
Bank - Age	Logistic Regression	0.82	0.82	0.79	0.19	0.5
	Random Forest	0.86	0.84	0.81	0.11	0.42
	SVC	0.82	0.82	0.8	0.19	0.5
	MLPC	0.85	0.83	0.8	0.14	0.44

Table 6: Baseline - Distribution-aware mutation: original scores

4.5.2 Mutation

Second, we create our mutated training datasets by targeted mutation towards a specific fairness goal. To generate these mutants, we use *Column Change (CC)* operator and design **targeted redistribution strategies** to explore fairness improvement approaches.
Our redistribution strategies are described below in the Adult dataset.

- **Protected attribute redistributing**
  As described in **RQ2.1**, we first want to investigate whether having the same proportions for each group (privileged/unprivileged) in each class improves fairness. We follow the next steps to achieve the same proportions. The overall process is illustrated in Figure 16.

  1. **Count the initial proportions**.
     We count the number of individuals in each subgroup of (*Protected attribute* x *Class*) in the original training dataset.
     For Adult:
     - 10,437 females are earning 50K$ or less per year.
     - 16,797 males are earning 50K$ or less per year.
     - 1,342 females are earning more than 50K$ a year.
     - 7,601 males are earning more than 50K$ a year.

2. **Count differences**.
   We count the differences between each group in each class to determine the number of instances to be mutated.
   Based on the previous numbers:
   - Among those earning 50K$ or less per year, the difference between males and females is 6,360. As a result, 3,180 males have to be transformed into females to obtain the same proportion of females and males in the negative class.
   - Among those earning more than 50K$ per year, the difference between males and females is 6,259. As a result, 3,130 males have to be transformed into females to obtain the same proportion of females and males in the positive class.
3. **Change value**.
   Based on the last differences, we randomly select indexes from the group with the most instances for each class and change their value in the sensitive attribute to the other group to obtain the same proportions.

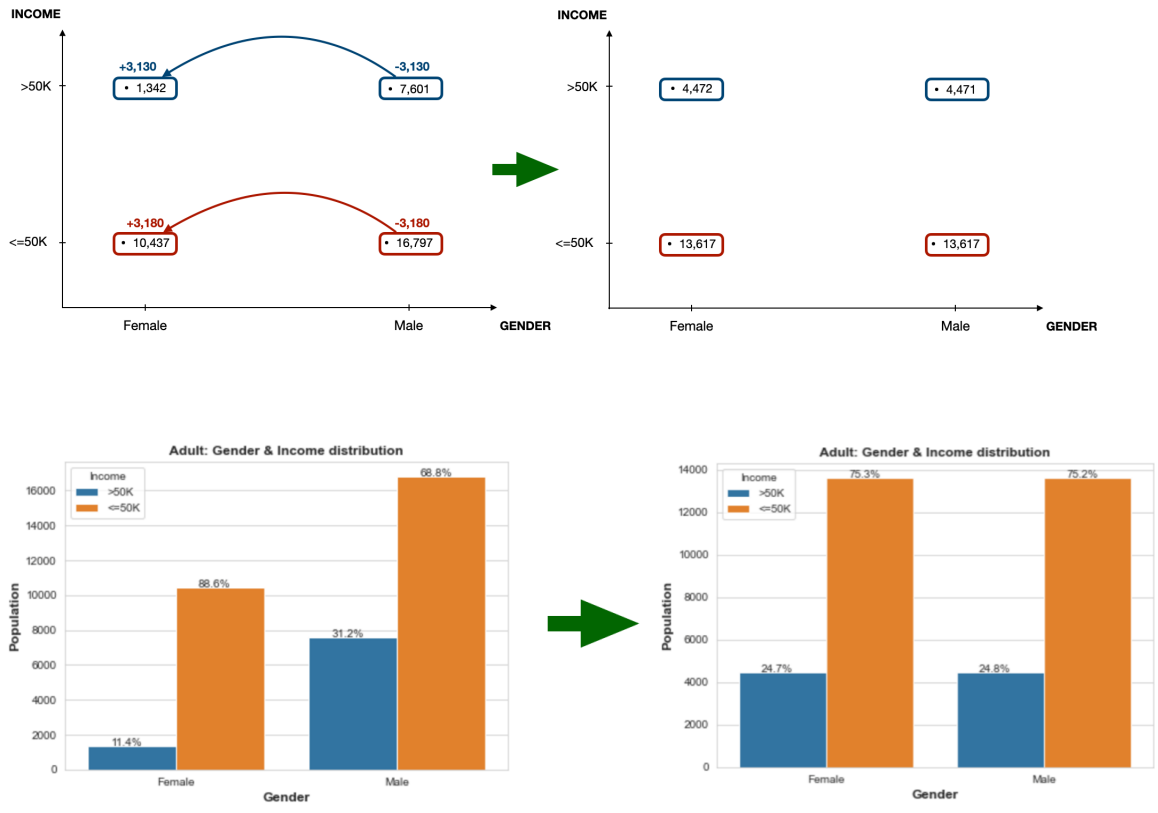


Figure 16: Protected attribute redistribution strategy

Through this strategy, we ensure for each class that the number of instances from the unprivileged group is equal to the number of instances from the privileged group.

- **Protected attribute and non-protected attributes redistributing**
  As described in **RQ2.2**, we want to go beyond making proportions equal based on class and protected attribute. We want to investigate whether making proportions equal based on class, protected attributes and non-protected attributes improve more fairness than the last redistribution strategy. We follow the next steps to achieve the same proportions. The overall process is illustrated in Figure 18.

  1. **Identify other attributes to be rebalanced**.
     When we make the proportions equal based on class and protected attribute in the previous redistribution strategy, we decrease the correlation between the protected attribute and the target y to around 0. Figure 17 shows the correlation matrix between the different attributes of the Adult training set. After rebalancing the instances regarding the "sex" attribute (highlighted columns), correlation between the protected attribute and the target y ≈ 0.

     As stated in **RQ2.2**, we believe that the initial over-representation is also present in other attributes that affect the final decision. Unfortunately, simply making proportions equal based on protected attribute and class does not recover the initial over-representation in other attributes.

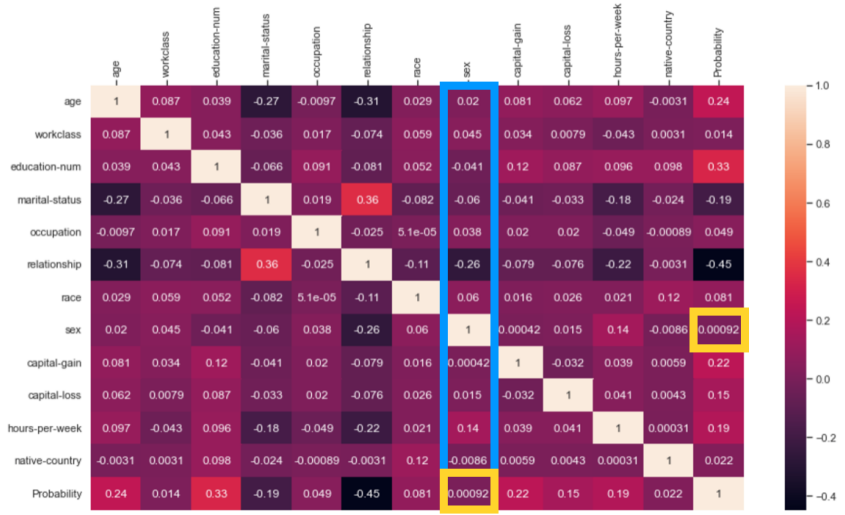


Figure 17: Correlation matrix after the protected attribute redistribution strategy

Based on the previous correlation matrix, we can see that *relationship* and *hours-per-week* are attributes where the different groups

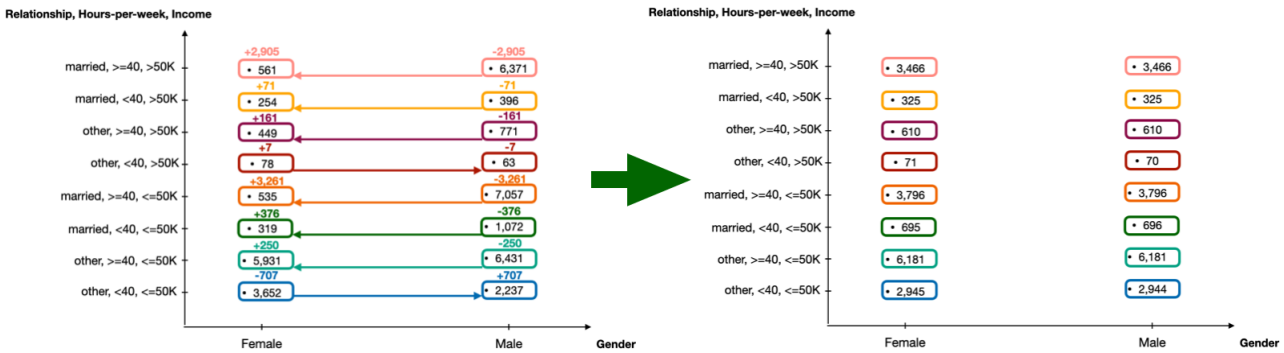
(privileged and unprivileged) still have different distributions. Therefore, we propose an approach to select attributes for which the groups still have a very different distribution even after making proportions equal based on class and protected attribute. We consider the attributes satisfying the following condition:

isSelected(att) = $|corr(\text{A}, \text{att})| > 0.1$,
where A is the protected attribute and att is the studied attribute.

Accordingly, we select the following attributes to be considered initially in the redistribution process :

- *relationship* and *hours-per-week* for Adult.
- no attribute for COMPAS and Bank since each attribute has a correlation with the protected attribute around 0 after being rebalanced on the class and the protected attribute.

2. **Count the initial proportions**.
   We count the number of individuals in each subgroup of (*Protected attribute* x *Selected attributes* x *Class*) in the original training dataset.
3. **Count differences**.
   We count the differences between each group in each subgroup of (*Selected attributes* x *Class*) to determine the number of instances to be mutated.
4. **Change value**.
   Based on the differences, we randomly select indexes from the group with the most instances for each subgroup of (*Selected attributes* x *Class*) and change their label in the sensitive attribute to the other group to obtain the same proportions.



Through this strategy, we ensure each subgroup of (*Selected attributes* x *Class*) that the number of instances from the unprivileged group is equal to the number of instances from the privileged group. As a result, the different groups (privileged and non-privileged) no longer have different

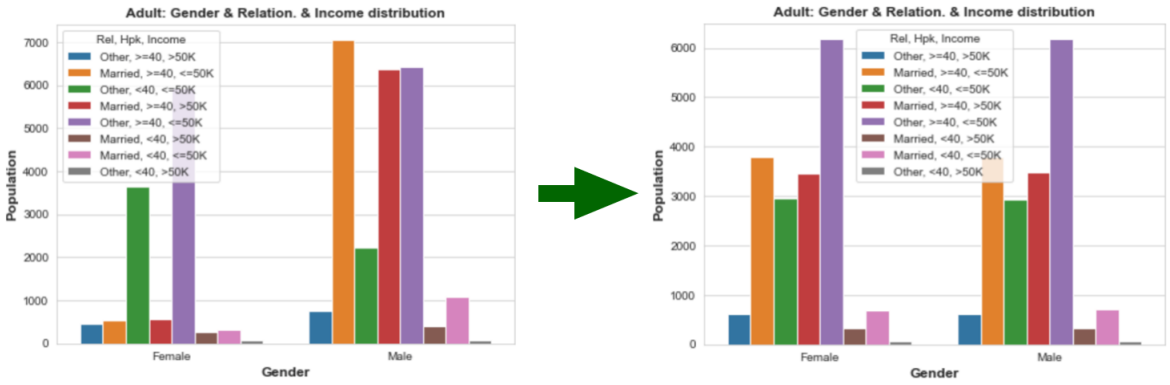


Figure 18: Protected and non-protected attributes redistribution strategy

distributions in the selected attributes. Figure 19 shows the correlation matrix between the different attributes of the Adult training set after being rebalanced with the Protected and non-protected attributes redistribution strategy.

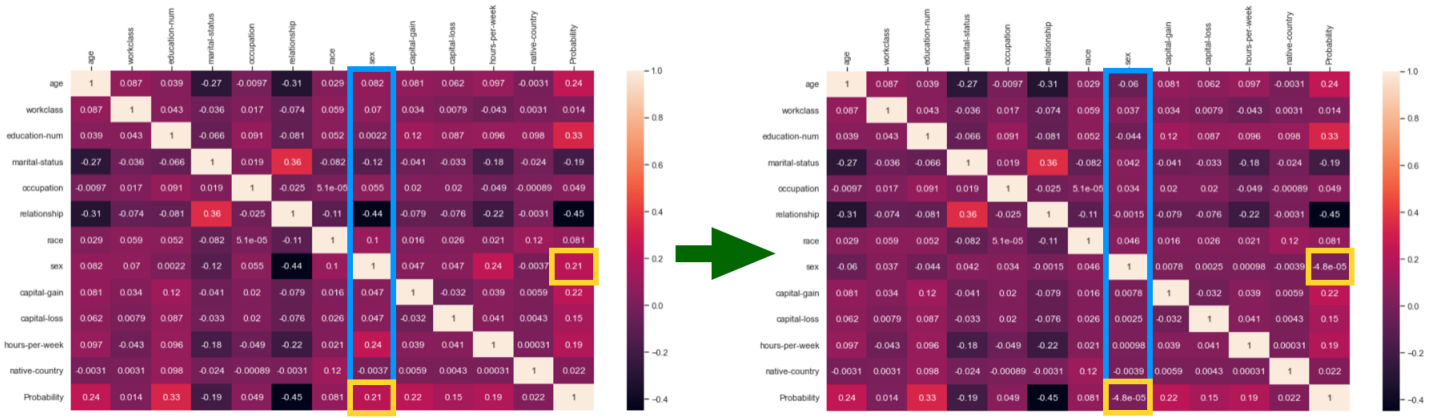


Figure 19: Correlation matrix after the protected and non-protected attribute selection strategy

Based on Figure 19, we can see that the protected attribute no longer correlates greater than 0.1 with any other attribute. This means that the distributions for the privileged and non-privileged subgroups are close for each attribute.

- **Mutant generation**
  Each redistribution strategy is illustrated with a scenario where we fully recover the distributions to make proportion equal based on protected attributes, class or more. For each redistribution strategy, we use ten channels to generate our mutated training datasets:
  10%-20%-30%-40%-50%-60%-70%-80%-90%-100% of distributions recovered. Each channel represents the percentage of distributions recov-

ered from the training dataset to fully apply the selection strategy. For each channel, we have five index folds. Each redistribution strategy leads us to create 50 training data mutants for each dataset.

5 Results and Discussion

We conduct a sensitivity analysis for each mutation scenario using three performance metrics (*recall, accuracy, precision*) and two fairness metrics (*EOD, SPD*) introduced in Section 2. To analyse the sensitivity of the algorithms when potential changes arise in data, we use 1,320 altered training datasets for the blind mutation scenario and 200 altered training datasets for the distribution-aware mutation scenario. To illustrate how these modifications impact the algorithms' fairness, we compare the performance and fairness scores of the resulting altered training datasets with our baselines presented in Table 5 and Table 6.

5.1 Blind mutation - Sensitivity analysis

First, we evaluate the sensitivity of the candidate learning algorithms through the mean, standard deviation, maximum and minimum of the resulting fairness scores of the altered datasets. Table 7 shows the results of the following boxplots.

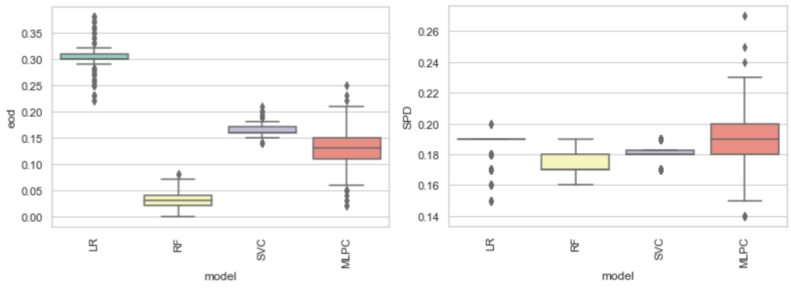


Figure 20: Blind mutation: Adult fairness scores

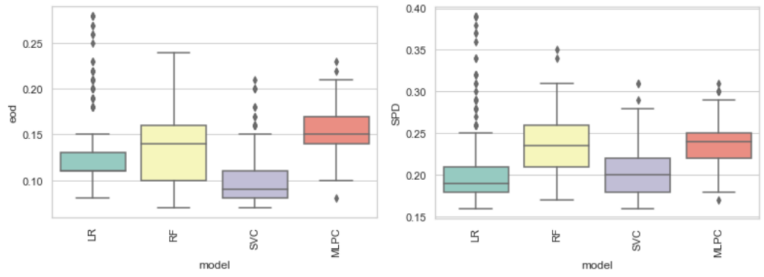


Figure 21: Blind mutation: COMPAS fairness scores

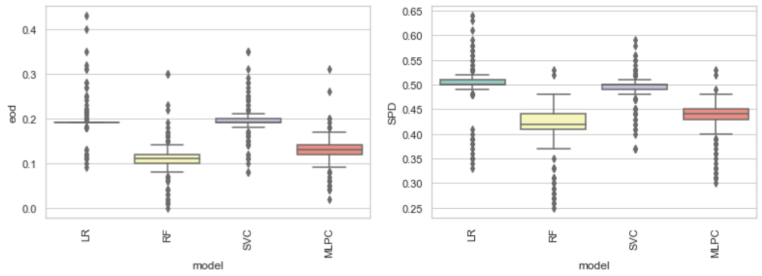


Figure 22: Blind mutation: Bank fairness scores

Dataset-Prot.Attr.	Learning Algorithm	Fairness Metric	Mean	Std	Min	Max
Adult - Sex	Logistic Regression	EOD	0.304	0.023	0.22	0.38
		SPD	0.186	0.008	0.15	0.2
	Random Forest	EOD	0.033	0.013	0	0.08
		SPD	0.174	0.005	0.16	0.19
	SVC	EOD	0.165	0.011	0.14	0.21
		SPD	0.182	0.005	0.17	0.19
	MLPC	EOD	0.127	0.036	0.02	0.25
		SPD	0.188	0.018	0.14	0.27
COMPAS - Race	Logistic Regression	EOD	0.132	0.044	0.08	0.28
		SPD	0.21	0.05	0.16	0.39
	Random Forest	EOD	0.135	0.036	0.07	0.24
		SPD	0.234	0.033	0.17	0.35
	SVC	EOD	0.101	0.028	0.07	0.21
		SPD	0.202	0.03	0.16	0.31
	MLPC	EOD	0.154	0.022	0.08	0.23
		SPD	0.237	0.025	0.17	0.31
Bank - Age	Logistic Regression	EOD	0.193	0.023	0.09	0.43
		SPD	0.502	0.025	0.33	0.64
	Random Forest	EOD	0.111	0.026	0	0.3
		SPD	0.422	0.028	0.25	0.53
	SVC	EOD	0.194	0.02	0.08	0.35
		SPD	0.497	0.018	0.37	0.59
	MLPC	EOD	0.128	0.023	0.02	0.31
		SPD	0.434	0.024	0.3	0.53

Table 7: Blind mutation: Boxplots scores

In addition, we analyse these results in the light of the baseline by examining how much the fairness scores can decrease with minor changes in the training dataset. As ensuring fairness often comes at the cost of performance, we also compare the impact of fairness enhancement on performance scores. Table 8 shows the scores of one of the resulting models that obtained the minimum for EOD or SPD. It also compares these scores with the baseline by calculating the gains or losses regarding fairness and performance.

Dataset-Prot.Attr.	Learning Algorithm	Minimum	Recall	Accuracy	Precision	EOD	SPD
Adult - Sex	Logistic Regression	EOD	-9%	-1%	+0%	-8%	-4%
		SPD	-9%	-1%	+0%	-8%	-4%
	Random Forest	EOD	-1%	-1%	+0%	-2%	+0%
		SPD	+0%	+0%	+0%	+0%	-1%
	SVC	EOD	+0%	+0%	-1%	-2%	+0%
		SPD	-3%	+0%	+1%	+0%	-1%
	MLPC	EOD	-6%	+0%	-3%	-11%	-5%
		SPD	-6%	+0%	-3%	-11%	-5%
COMPAS - Race	Logistic Regression	EOD	-1%	+0%	+0%	-3%	-2%
		SPD	-1%	+0%	+0%	-3%	-2%
	Random Forest	EOD	-2%	+0%	+0%	-8%	-8%
		SPD	-2%	+0%	+0%	-8%	-8%
	SVC	EOD	+0%	+0%	+0%	-1%	+0%
		SPD	+0%	+0%	+0%	-1%	-1%
	MLPC	EOD	+6%	+1%	-2%	-7%	-7%
		SPD	+6%	+1%	-2%	-7%	-7%
Bank - Age	Logistic Regression	EOD	-1%	+0%	+1%	-10%	-17%
		SPD	-1%	+0%	+1%	-10%	-17%
	Random Forest	EOD	-1%	+0%	+0%	-11%	-14%
		SPD	-4%	-1%	+1%	-7%	-17%
	SVC	EOD	+0%	+0%	-1%	-11%	-13%
		SPD	+0%	+0%	-1%	-11%	-13%
	MLPC	EOD	+1%	+1%	+0%	-12%	-14%
		SPD	+1%	+1%	+0%	-12%	-14%

Table 8: Blind mutation: sensitivity analysis

RQ1.1. What is the impact of a few injected perturbations in the training data on the fairness scores of a machine learning model?

From what we can see, the fairness of machine learning algorithms is highly sensitive to mutations in the training data. As shown in Table 8, a few alterations in the original training data can already lead to significant fairness enhancement:

- For Adult, MLPC trained on a mutated training dataset obtains gains up to 11% in EOD and 5% in SPD.
- For COMPAS, Random Forest trained on a mutated training dataset obtains gains up to 8% in EOD and 8% in SPD.
- For Bank, MLPC trained on a mutated training dataset obtains gains up to 12% in EOD, and Logistic Regression obtains gains up to 17% in SPD.

The results show that the nature of the training dataset is a crucial factor in the fairness of an ML model. This reinforces our first assumption that the fairness of an ML model is very sensitive to problems in the training data and that we should study the constitution of the training data to eliminate the bias.
Furthermore, fairness does not necessarily come at the cost of performance. As presented in Table 8, only recall appears to be impacted. This represents an additional motivation to explore the mutation of training data to enhance the fairness of an ML model while maintaining its performance.

RQ1.2. Which machine learning algorithms are the most sensitive (or inflexible) to mutations in the training data regarding fairness?

Depending on the nature of the mutated training dataset, the variation in the fairness of an algorithm differs. Consider the following ranking based on the algorithm with the biggest fairness improvement.

Dataset-Prot.Attr.	Metric	1st	2nd	3rd	4th
Adult - Sex	EOD	MLPC*(-11%)*	LR*(-8%)*	RF and SVC*(-2%)*	/
	SPD	MLPC*(-5%)*	LR*(-4%)*	RF and SVC*(-1%)*	/
COMPAS - Race	EOD	RF*(-8%)*	MLPC*(-7%)*	LR*(-3%)*	SVC*(-1%)*
	SPD	RF*(-8%)*	MLPC*(-7%)*	LR*(-2%)*	SVC*(-1%)*
Bank - Age	EOD	MLPC*(-12%)*	SVC and RF*(-11%)*	LR*(-10%)*	/
	SPD	LR and RF*(-17%)*	MLPC*(-14%)*	SVC*(-13%)*	/

Table 9: Fairness enhancement ranking

Based on Table 9, no algorithm gets the best fairness improvement on all datasets.
Furthermore, there is no algorithm where the fairness scores are more spread out for each dataset. Regarding Figures 20,21, 22, we could say at first glance that Random Forest (RF) or MLPC are the most sensitive algorithms since their interquartile range is more spread than the others. However, the previous boxplots also show a large number of outliers within the Logistic Regression fairness results. Since Logistic Regression shows many fairness scores that do not follow the same pattern as others, its fairness appears unpredictable and highly variable. It can also be interpreted as a lack of robustness to data mutations.
Therefore, previous results do not allow us to say that one algorithm is more prone to data alterations than others. The sensitivity of an algorithm to modifications in the training data is specific to the nature of the original dataset on which the mutations are performed.

RQ1. How sensitive are machine learning algorithms to mutations in the training data regarding fairness?

In summary, the nature of the training dataset is a crucial factor in the fairness of an ML model. A few injected mutations in the training dataset can lead to large variations in the fairness of an ML model (i.e, high sensitivity of the learning algorithm to a few injected perturbations in the training dataset). This confirms our original assumption that fairness issues of an ML model are strongly linked to problems in the data. Therefore, the mutation approach represents a pertinent method to explore the fairness of an ML model and track fairness improvement achieved by mutation techniques on the original training dataset.
Nevertheless, the sensitivity of algorithms to alterations in the training data is specific to the nature of the original dataset on which the mutations are performed. The sensitivity of an algorithm regarding fairness differs from one dataset to another. Therefore, the fairness-driven sensitivity interpretation of an algorithm through the mutation approach must remain in the dataset's context.

5.2 Distribution-aware mutation - Sensitivity analysis

We analyse the sensitivity of the candidate learning algorithms for each redistribution strategy by analysing the evolution of their fairness and performance scores as we recover the original training dataset distributions.

- **Protected attribute redistributing**

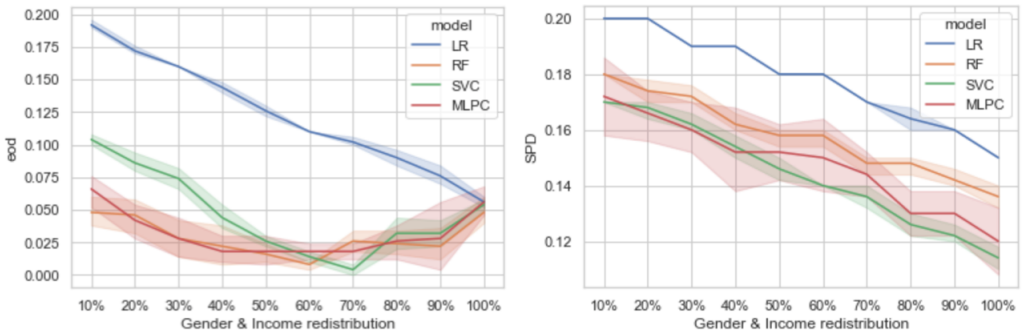


Figure 23: Protected attribute redistributing: Adult fairness scores

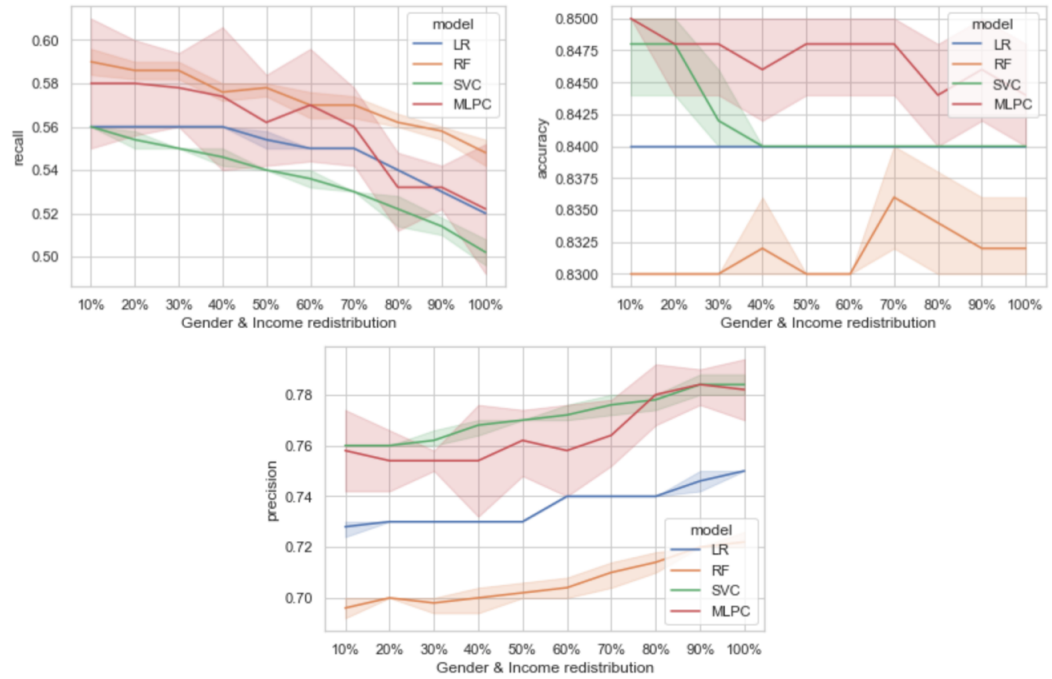


Figure 24: Protected attribute redistributing: Adult performance scores

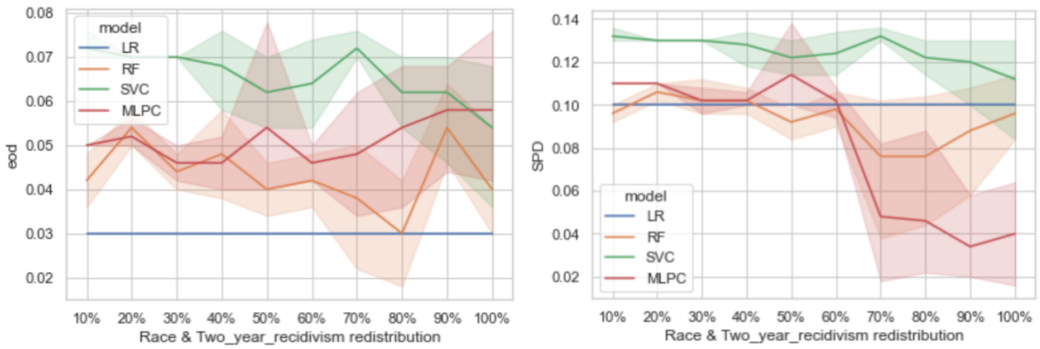


Figure 25: Protected attribute redistributing: COMPAS fairness scores

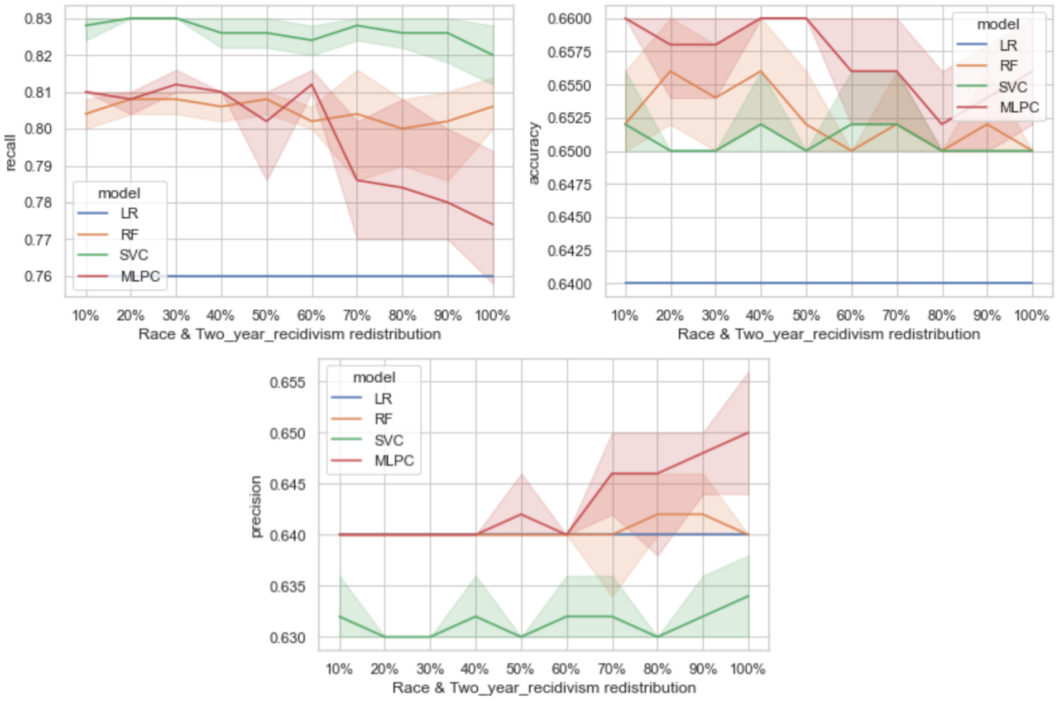


Figure 26: Protected attribute redistributing: COMPAS performance scores

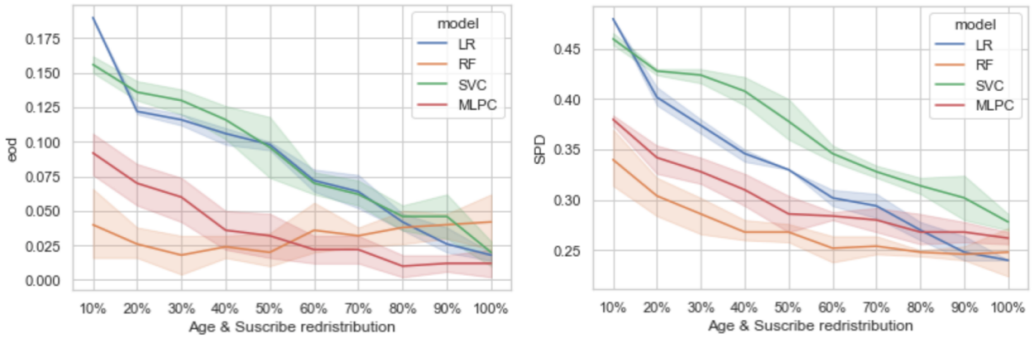


Figure 27: Protected attribute redistributing: Bank fairness scores

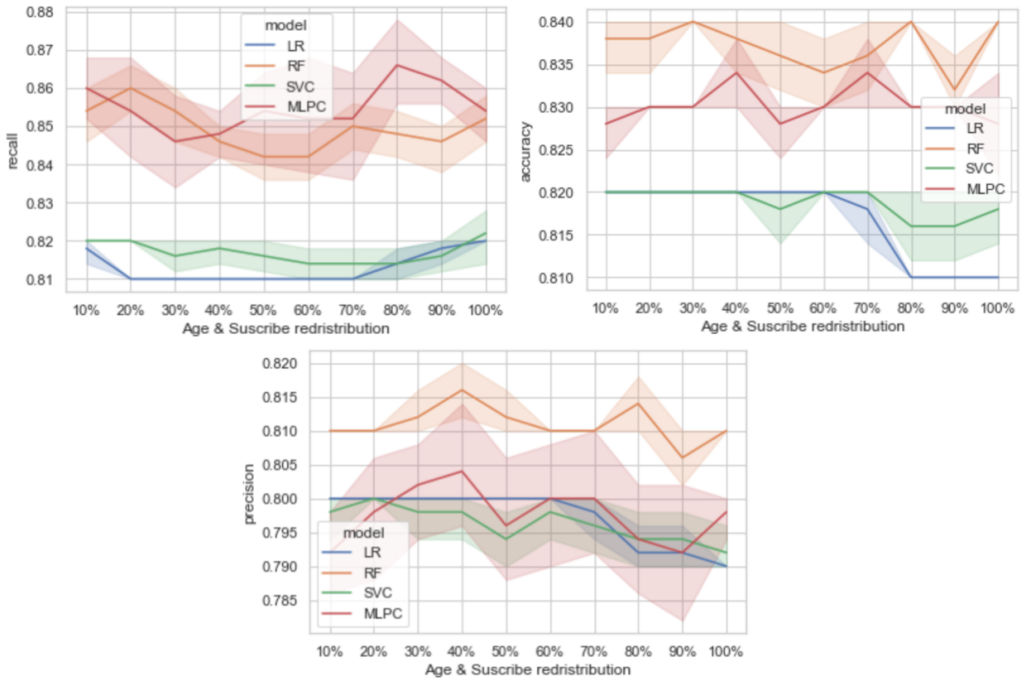


Figure 28: Protected attribute redistributing: Bank performance scores

Table 20 shows these results in the light of the baseline by examining gains and losses in fairness and performance for algorithms trained on mutated training datasets where we fully recovered distributions(100%).

Dataset-Prot.Attr.	Learning Algorithm	Recall	Accuracy	Precision	EOD	SPD
Adult - Sex	Logistic Regression	+4%	+0%	+3%	-18%	-6%
	Random Forest	-3%	+1%	+3%	-3%	-4%
	SVC	-6%	-1%	+2%	-8%	-7%
	MLPC	-10%	-1%	+4%	-5%	-7%
COMPAS - Race	Logistic Regression	+0%	+0%	+0%	+0%	+0%
	Random Forest	-1%	-1%	+0%	-2%	-3%
	SVC	-1%	+0%	+0%	-2%	-2%
	MLPC	+1%	-1%	+0%	-3%	-4%
Bank - Age	Logistic Regression	+0%	-1%	+0%	-18%	-26%
	Random Forest	-1%	+0%	+0%	-6%	-18%
	SVC	+1%	+0%	-1%	-16%	-22%
	MLPC	+0%	-1%	-1%	-12%	-17%

Table 10: Protected attribute redistributing: sensitivity analysis

RQ2.1. Can having the same proportions for each group (privileged/unprivileged) in each class reduce the bias?

The results obtained in Figures 23 and 27 are consistent with our hypothesis that redistribution of the protected attribute in a directed manner can improve fairness. From what we can see, there are significant fairness enhancements for directed redistribution on the protected attribute in the Adult and Bank dataset. Indeed, the SPD of each ML algorithm decreases significantly as we redistribute the protected attribute, while the EOD varies differently depending on the ML algorithm.
However, the results in Figure 25 do not allow us to say that redistribution on the protected attribute in a directed manner in COMPAS dataset improves fairness. Unlike Figures 23 and 27, the fairness results obtained in Figure 25 does not show for each algorithm the same decreasing tendency of fairness scores.
Therefore, this redistribution strategy needs to be further explored in other datasets to be validated. Although we do not obtain the expected results on COMPAS, this strategy performs well on Adult and Bank with significant fairness improvements. Moreover, Table 10 also shows that the method can improve fairness while not seriously affecting performance.

- **Protected attribute and non-protected attributes redistributing**

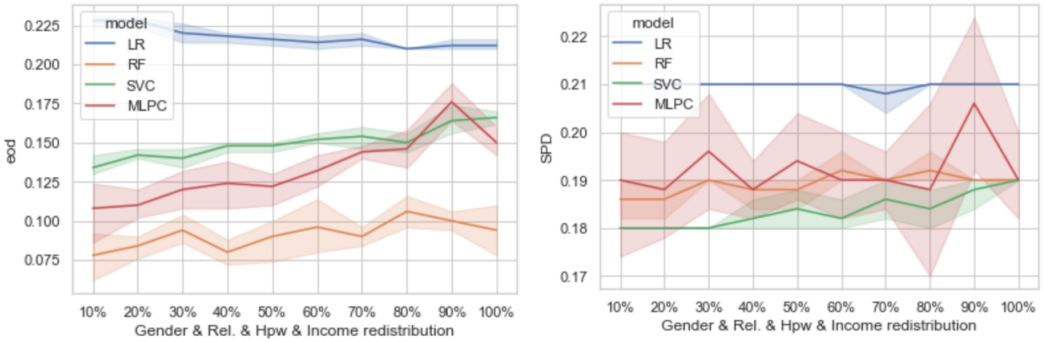


Figure 29: Protected attribute redistributing: sensitivity analysis

RQ2.2. Can making proportions equal based on class, protected, and non-protected attributes reduce bias?

Results obtained in Figure 29 do not present fairness enhancements. On the contrary, the fairness results for most algorithms increase as we recover the distribution. Therefore, modifying the training dataset such that privileged and unprivileged groups have the same distribution in each class and in non-protected attributes does not reduce bias for Adult. Nevertheless, this remains to be confirmed on other datasets.

RQ2. What targeted mutations in training data can improve the fairness of a machine learning model?

In summary, we obtain interesting results for the first redistribution strategy, suggesting that having the same proportions for each class; group improves fairness. Nevertheless, this remains to be confirmed on other datasets, as this strategy does not improve fairness as expected for the COMPAS dataset (Figure 25).
However, the second redistribution strategy, which suggests having the same proportions for each group in each class while having the same proportions for each group in the non-protected attributes, does not improve the fairness of the ML models.

6 Threats to validity

Evaluation Bias. Although IBM AIF360 [23] contains over 50 metrics, we used only two of the most popular fairness metrics. We should explore the impact of mutations in training data with further evaluation criteria.

Internal Validity. *Default configurations.* For each learning algorithm, we used the default configuration as a representative of its learning algorithm "family" and explored its behaviour on altered datasets. During our experiments in the blind mutation scenario, we have shown that a few injected perturbations in the training dataset can lead to high variations in the fairness scores of these learning algorithms with the default configuration. We postulate that we would obtain the same results with different hyperparameter configurations. Nevertheless, we should verify this assumption by using other hyperparameter configurations of the candidate learning algorithms.
Naive mutations. Our blind mutation scenario treats the original training dataset as a black box and, therefore, it can lead to unrealistic mutated training datasets. As we naively modify the original training dataset, some mutations may be harmful since they may modify an attribute that does not need to be changed. For instance, we may blindly modify *age* attribute in Adult and create absurd instances such as 14-year-olds, married, with a master's degree.
Furthermore, as our blind mutation scenario represents the first attempt to see mutation-induced fairness changes, we may miss important information to inject critical perturbations in the training data. Therefore, we should pay more attention to the definition of mutation techniques to improve our mutation approach so that it is effective for the fairness analysis of ML algorithms.

External Validity. We conducted our experiments on binary classifiers and popular datasets, which are commonly used in the literature of machine learning fairness research. We should extend this work to other datasets and other ML algorithms to explore the usefulness of our approach.

7 Conclusion

Fairness is an important property that well-designed ML systems should have. As machine learning (ML) systems are increasingly used in critical systems (e.g., recruitment and lending), it is crucial to ensure that decisions computed by such systems do not exhibit unfair behaviour. Therefore, diagnosing, exposing and mitigating bias in ML systems is important.

To this end, this thesis has explored whether ML algorithms produce similar behaviours for several altered datasets with potential injected fairness issues. As ML algorithms have different learning procedures, they may show different sensitivity to biases in data. To explore and analyse their sensitivity, we proposed an approach relying on mutation testing to inject potential biases in data and measure the sensitivity of ML algorithms to these biases.
Assuming fairness problems are strongly linked to issues in the data, we first designed data alteration techniques. Through our mutation operators, we proposed different ways to inject fairness defects/advantages that could potentially be introduced into the data.
Then, we proposed two mutation scenarios to explore the impact of data alterations on ML systems' fairness.

We initially studied the impact of slight data alterations through a blind mutation scenario on three popular datasets with four ML algorithms. By naively applying our mutation operators to small parts of the data, we demonstrated high variations in ML systems' fairness (i.e, high sensitivity of the algorithms to injected perturbations in data).
As the previous result convinced us that data mutation has an important effect on ML systems' fairness, we investigated bias mitigation approaches through a distribution-aware mutation scenario. We implemented two redistribution strategies to reduce bias in data. We demonstrated the usefulness of redistribution by considering the proportions of the different groups in relation to the target y on two datasets, Adult and Bank, with four algorithms. However, we failed to improve fairness while we redistributed by considering the proportions of the different groups in relation to the target y and non-sensitive attributes.

Based on the above, we believe the mutation approach is a promising technique that could discover fairness violations in ML systems for some training data patterns. This master's thesis represents an initial exploratory attempt to demonstrate the usefulness of the mutation approach for a further in-depth understanding of fairness in ML systems. For future work, we plan to perform a more comprehensive study to propose smarter data mutation operators to inject critical fairness issues in data and investigate the relations of mutation operators and how well such operators introduce fairness faults in data.

References

[1] Aggarwal A., Lohia P., Nagar S., Dey K., and Saha D. Black box fairness testing of machine learning models. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 625–635, 2019.

[2] Rico Angell, Brittany Johnson, Yuriy Brun, and Alexandra Meliou. Themis: Automatically testing software for discrimination. In *Proceedings of the 2018 26th ACM Joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pages 871–875, 2018.

[3] Calmon F., Wei D., Vinzamuri B., Natesan Ramamurthy K, and Varshney KR. Optimized pre-processing for discrimination prevention. *Advances in neural information processing systems*, 30, 2017.

[4] Kamiran F. and Calders T. Data preprocessing techniques for classification without discrimination. *Knowledge and information systems*, 33(1):1–33, 2012. Publisher: Springer.

[5] Suresh H. and Guttag JV. A framework for understanding unintended consequences of machine learning. *arXiv preprint arXiv:1901.10002*, 2:8, 2019.

[6] Angwin J., Larson J., Mattu S., and Kirchner L. How we analyzed the compas recidivism algorithm. `https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm`, May 2016.

[7] Chakraborty J., Majumder S., and Menzies T. Bias in machine learning software: why? how? what to do? In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 429–440, 2021.

[8] Chakraborty J., Xia T., Fahid F., and Menzies T. Software engineering for fairness: A case study with hyperparameter optimization. *arXiv preprint arXiv:1905.05786*, 2019.

[9] Dastin J. Amazon scraps secret AI recruiting tool that showed bias against women. In *Ethics of Data and Analytics*, pages 296–299. Auerbach Publications, 2018.

[10] Zhang J and Harman M. " Ignorance and Prejudice" in Software Fairness. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1436–1447. IEEE, 2021.

[11] Zhang JM., Harman M., Ma L., and Liu Y. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020. Publisher: IEEE.

[12] Michael I. Jordan and Tom M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015. Publisher: American Association for the Advancement of Science.

[13] Michael I. Jordan and Tom M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015. Publisher: American Association for the Advancement of Science.

[14] Kaggle. Bank marketing uci. `https://www.kaggle.com/c/bank-marketing-uci/`, 2017.

[15] Ma L., Zhang F., Sun J., Xue M., Li B., Juefei-Xu F., Xie C., Li L., Liu Y., and Zhao J. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, pages 100–111. IEEE, 2018.

[16] Feldman M., Friedler SA., Moeller J., Scheidegger C., and Venkatasubramanian S. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.

[17] Hardt M., Price E., and Srebro N. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[18] Hort M., Zhang JM., Sarro F., and Harman M. Fairea: A model behaviour mutation approach to benchmarking bias mitigation methods. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 994–1006, 2021.

[19] Zafar MB., Valera I., Rogriguez MG, and Gummadi K. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*, pages 962–970. PMLR, 2017.

[20] Mehrabi N., Morstatter F., Saxena N., Lerman K., and Galstyan A. A Survey on Bias and Fairness in Machine Learning. *arXiv:1908.09635 [cs]*, September 2019. arXiv: 1908.09635.

[21] Zhang P., Wang J., Sun J., Wang X., Dong G., Wang X., Dai T., and Dong JS. Automatic Fairness Testing of Neural Classifiers through Adversarial Sampling. *IEEE Transactions on Software Engineering*, 2021. Publisher: IEEE.

[22] Propublica. Data for the propublica story 'machine bias'. `https://github.com/propublica/compas-analysis`, 2015.

[23] Bellamy R, Dey K., Hind M., Hoffman S., Houde S., Kannan K., Lohia P., Martino J., Mehta S., and Mojsilovic A. AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.

[24] Zemel R., Wu Y., Swersky K., Pitassi T., and Dwork C. Learning fair representations. In *International conference on machine learning*, pages 325–333. PMLR, 2013.

[25] Galhotra S., Brun Y., and Meliou A. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint meeting on foundations of software engineering*, pages 498–510, 2017.

[26] Hamimoune S. and Falah B. Mutation testing techniques: A comparative study. In *2016 international conference on engineering & MIS (ICEMIS)*, pages 1–9. IEEE, 2016.

[27] Udeshi S., Arora P., and Chattopadhyay S. Automated directed fairness testing. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 98–108, 2018.

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

[29] Le Quy T., Roy A., Iosifidis V., Zhang W., and Ntoutsi E. A survey on datasets for fairness-aware machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2022. Publisher: Wiley Online Library.

[30] Uci. Adult data set. `https://archive.ics.uci.edu/ml/datasets/adult`, 1994.

[31] Knight W. The apple card didn't 'see' gender—and that's the problem. `https://www.wired.com/story/the-apple-card-didnt-see-genderand-thats-the-problem/`, 2019.

[32] Brun Y. and Meliou A. Software fairness. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pages 754–759, 2018.

A Datasets' characteristics

Attributes	Type	Values	#Missing values	Description
name	Categorical	7,118	0	First and last name of the defendant
first	Categorical	2,800	0	First name
last	Categorical	3,950	0	Last name
compas_screening_date	Categorical	690	0	The date of which the decile score was given
sex	Binary	{Male, Female}	0	Gender
dob	Categorical	5,452	0	Date of birth
age	Numerical	[18 - 96]	0	Age in years
age_cat	Categorical	3	0	Age in a categorical form
race	Categorical	6	0	Race
juv_fel_count	Numerical	[0 - 20]	0	The juvenil felony count
decile_score	Numerical	[1 - 10]	0	The COMPAS Risk o Recidivism score
juv_misd_count	Numerical	[0 - 13]	0	The juvenil misdemeanor count
juv_other_count	Numerical	[0 - 17]	0	The juvenil other offenses count
priors_count	Numerical	[0 - 38]	0	The prior offenses count
days_b_screening_arrest	Numerical	[-414 - 1,057]	307	The number of days between COMPAS screening and arrest
c_jail_in	Categorical	6,907	307	The jail entry date for original crime
c_jail_out	Categorical	6,880	307	The jail exit date for original crime
c_case_number	Categorical	7,192	22	The case number for original crime
c_offense_date	Categorical	927	1,159	The offense date of original crime
c_arrest_date	Categorical	580	6,077	The arrest date for original crime
c_days_from_compas	Numerical	[0 - 9,485]	22	Between the COMPAS screening and the original crime offense date
c_charge_degree	Binary	{F, M}	0	Charge degree of original crime
c_charge_desc	Categorical	437	29	Description of charge for original crime
is_recid	Binary	{0, 1}	0	The binary indicator of recidivation
r_case_number	Categorical	3,471	3,743	The case number of follow-up crime
r_charge_degree	Categorical	10	3,743	Charge degree of follow-up crime
r_days_from_arrest	Numerical	[-1 - 993]	4,898	Between the follow-up crime and the arrest date (days)
r_offense_date	Categorical	1,075	3,743	The date of follow-up crime
r_charge_desc	Categorical	340	3,801	Description charge for follow-up crime
r_jail_in	Categorical	972	4,898	The jail entry date for follow-up crime
r_jail_out	Categorical	938	4,898	The jail exit date for follow-up crime
violent_recid		NULL	7,214	Values are all NA. This column is ignored
is_violent_recid	Binary	{0, 1}	0	The binary indicator of violent follow-up crime
vr_case_number	Categorical	819	6,395	The case number for violent follow-up crime
vr_charge_degree	Categorical	9	6,395	Charge degree for violent follow-up crime
vr_offense_date	Categorical	570	6,395	The date of offense for violent follow-up crime
vr_charge_desc	Categorical	83	6,395	Description of charge for violent follow-up crime
type_of_assessment	Categorical	1	0	The type of COMPAS score given for decile score
decile_score.1	Numerical	[1 - 10]	0	Repeat column of *decile_score*
score_text	Categorical	3	0	Propublica-defined category of *decile_score*
screening_date	Categorical	690	0	Repeat column of *compas_screening_date*
v_type_of_assessment	Categorical	1	0	The type of COMPAS score given for *v_decile_score*
v_decile_score	Numerical	[1 - 10]	0	The COMPAS Risk of Violence score from 1 to 10
v_score_text	Categorical	3	0	Propublica-defined category of *v_decile_score*
v_screening_date	Categorical	690	0	The date on which *v_decile_score* was given
in_custody	Categorical	1,156	236	The date on which individual was brought into custody
out_custody	Categorical	1,169	236	The date on which individual was released from custody
priors_count.1	Numerical	[0 - 38]	0	Repeat column of *priors_count*
start	Numerical	[0 - 937]	0	No information
end	Numerical	[0 - 1,186]	0	No information
event	Binary	{0, 1}	0	No information
two_year_recid	Binary	{0, 1}	0	Whether the defendant is rearrested within two years

Table 11: COMPAS: attributes characteristics