ABSTRACT

CHAKRABORTY, JOYMALLYA. Deciphering ML Software Fairness. (Under the direction of Dr. Tim Menzies).

Algorithmic discrimination in the AI software systems has become a matter of serious concern in the machine learning and software engineering community. Prior works have treated finding bias and removing bias as two separate tasks. There is no prior work that covered the whole notion of ML fairness. As a result, it became difficult for the practitioners to have a good understanding of ML fairness by reading a few papers. We, in this work, ask simple questions and seek answers.

- What causes the bias?
- Can we remove bias from the model by removing the cause?
- How does that affect the performance of the model?
- Can we simplify software fairness?

In Stage1, we find the answer to the first question. After experimenting on ten real-world datasets, we realize that bias comes from the historical data. The reason could be imbalance between different populations or interactions among the features.

In Stage2, we develop two novel bias mitigation algorithms by using the knowledge acquired in Stage1. Our first work is Fairway, a combination of a data pre-processor and a hyperparameter optimizer. The pre-processor removes bias from the training data and the hyperparameter optimizer tunes the model to balance the trade-off between fairness and performance. Our second work is Fair-SMOTE, a data balancing approach. Far-SMOTE removes biased labels from the training data and re-balances internal distributions such that they are equal based on class and sensitive attributes.

In Stage3, we try to simplify software fairness. At first, we develop the FairBalance algorithm which is cheaper than Fair-SMOTE and works for more than one sensitive attribute. Then we develop Fair-SSL that uses semi-supervised learning and it only requires 10% training data labels. Thus, it is less expensive than Fair-SMOTE. At last, we simplify the complicated procedure of selecting appropriate fairness metrics.

In Stage4, we try to extend our algorithms in the domain of causality, a vastly explored area in the machine learning community. We propose a new way of fairness testing using propensity score based on causal interaction.

We test our methods on ten popular datasets and with three different machine learning models. Experimental results show that our algorithms reduce bias as well as state-of-the-art bias mitigation algorithms and sometimes improve performance of the learner (measured in terms of recall and F1) where prior algorithms always damage performance of the learner.

Deciphering ML Software Fairness

by
Joymallya Chakraborty

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2022

APPROVED BY:

Dr. Noboru Matsuda	Dr. Jamie Jennings
Dr. John-Paul Ore	Dr. Bradley Reaves Graduate School Representative

Dr. Tim Menzies
Chair of Advisory Committee

DEDICATION

To my parents Gita Chakraborty and Shyamal Chakraborty.

BIOGRAPHY

Joymallya Chakraborty was born in Kolkata, India. He attended Jadavpur University, one of the most prestigious universities of his state, where he got bachelor's degree in Computer Science. Then he worked in the software industry for two years. In 2017, he moved to the United States and joined North Carolina State University. He has been a research assistant at the RAISE lab under the supervision of Dr. Tim Menzies since 2018. His research interest lies in finding and mitigating ethical bias in machine learning models. He has done research internships at Intel Corporation (Bellevue, WA), IBM T.J. Watson Research Labs (Yorktown Heights, NY), & Amazon (Seattle, WA). He is now joining Amazon as an Applied Scientist.

ACKNOWLEDGEMENTS

I would like to whole-heartedly thank,

★ My advisor, **Prof. Tim Menzies**, for his *trust, support, and guidance.*

- **Dr. George N. Rouskas** (Director of Graduate Programs) for his all time support. My dissertation committee members **Dr. Bradley Reaves, Dr. Jamie Jennings, Dr. Noboru Matsuda**, and **Dr. John-Paul Ore** for their valuable feedback.
- All the researchers at RAISE lab. I would like to specially thank my research collaborators **Dr. Zhe Yu**, **Dr. Jianfeng Chen**, **Suvodeep Majumder**, **Dr. Tianpei Xia**, **Dr. Huy Tu**, and **Kewen Peng**.
- My friends **Anjali Malunjkar, Pranoy Kundu, Priyanka Das, Rahul Chakraborty, Utsab Ray,** and **Debanjan Chatterjee** for their support and encouragement.
- My colleagues from Intel Corporation: **Simonjit Dutta, Devinder Singh, Xiangyang (Mark) Guo**; from IBM Research: **Anup Kalia, Maja Vukovic, Jin Xiao, Debasish Banerjee**; from Amazon.com: **Naveed Janvekar,** and **Nitika Bhaskar**.
- The Department of Computer Science staff **Kathy Luca**, **Carol Allen**, **Kiresten Smith**, and all the others.
- Nobody has been more important to me in the pursuit of this honor than the members of my family. I want to thank **Gita Chakraborty (mother), Shyamal Chakraborty (father), Roshni Chakraborty (sister), Mainak Chakraborty (brother-in-law),** and **Riyom Chakraborty (nephew)**.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part 1 - Understanding the Problem

CHAPTER

1

INTRODUCTION

1.1 What is the Problem ?

Software plays an important role in many high-stake applications like finance, hiring, admissions, criminal justice. For example, software generates models that decide whether a patient gets released from hospital or not [Med; Str16]. Also, software helps us to choose what products to buy [Wal]; which loan applications are approved [Fora]; which citizens get bail or sentenced to jail [Pro]. These all are examples of software systems where the core part is machine learning model.

One problem with any machine learning (ML) model is they are all a form of statistical discrimination. Consider, for example, the discriminatory nature of decision tree learners that deliberately selects attributes to divide data into different groups. Such discrimination becomes unacceptable and unethical when it gives certain privileged groups advantages while disadvantaging other unprivileged groups (e.g. groups divided by age, gender, skin color, etc). In such situations, discrimination or bias is not only objectionable, but illegal. Therefore it is essential to understand why a certain model is biased and how to mitigate that bias to generate fair outcomes.

1.2 Statement of Thesis

Bias comes from the historical data. Therefore this work finds out - a) appropriate data pre-processing techniques to remove bias from the training data before supervised model training, b) tuning hyper-parameters of the supervised model with multiple objectives can lead to fair and better predictions.

1.3 Research Questions

At first we describe the software fairness problem and the works done by prior researchers. Then we describe the four different frameworks built by us - **Fairway, Fair-SMOTE, FairBalance**, and **Fair-SSL**. We evaluate the performance of these four frameworks by seeking answers for the following research questions. Fairway is related to RQ2, and RQ3. Fair-SMOTE is related to RQ1, and RQ2. FairBalance and Fair-SSL are related to RQ2, and RQ4.

RQ1: What are the root causes of bias? (Chapter 3)

RQ2: Can data pre-processing help to reduce bias? (Chapter 4, Chapter 5)

RQ3: Can hyperparameter optimization help to reduce bias? (Chapter 4)

RQ4: Can we reduce the cost of the mitigation process? (Chapter 6, Chapter 7)

1.4 Novel Contributions

1. This is the first study in software engineering combining the root causes of bias, the way of finding bias and the way of removing bias.
2. Measured in terms of number of datasets and models used, this is the largest study on software fairness till date to the best of our knowledge.
3. Unlike previous work, we offer a framework that improves both fairness and performance of the model.
4. We focused on binary classification in this work. However, the core ideas of our research can be easily extended to other domains such as regression, text classification and multi-class classification.

1.5 Published Works

1.5.1 Published Full Papers

- [Cha22] **Joymallya Chakraborty**, Suvodeep Majumder, and Huy Tu. “Fair-SSL: Building fair ML Software with less data”, Fairware 2022, Proceedings of the International Workshop on Equitable Data and Technology, 2022.
- [Cha21] **Joymallya Chakraborty**, Suvodeep Majumder, and Tim Menzies. "Bias in Machine Learning Software: Why? How? What to do?" Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2021. [**ACM SIGSOFT Distinguished Paper Award Winner**]

- [Cha20b] **Joymallya Chakraborty**, Suvodeep Majumder, Zhe Yu, and Tim Menzies. "Fairway: A way to build fair ML Software" Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2020.

- [Che19b] Jianfeng Chen, **Joymallya Chakraborty**, Tim Menzies, Philip Clark, Kevin Haverlock, and Snehit Cherian. "Predicting Breakdowns in Cloud Services (with SPIKE)" Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2019.

- [Imt19] Nasif Imtiaz, Justin Middleton, **Joymallya Chakraborty**, Neill Robson, Gina Bai, and Emerson Murphy-Hill. "Measuring the Effects of Gender Bias on GitHub" Proceedings of the International Conference on Software Engineering (ICSE), 2019.

1.5.2 Published Short Papers

- [Cha20c] **Joymallya Chakraborty**, Kewen Peng, and Tim Menzies. "Making Fair ML Software using Trustworthy Explanation" Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2020.

- [Cha19] **Joymallya Chakraborty**, Tianpei Xia, Fahmid M. Fahid, and Tim Menzies. "Software Engineering for Fairness: A Case Study with Hyperparameter Optimization" Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019.

1.5.3 Under Review

- **Joymallya Chakraborty**, Suvodeep Majumder, and Tim Menzies. “Causality-Based Testing for Software Fairness”, ESEC/FSE (2022).

- Suvodeep Majumder, **Joymallya Chakraborty**, Gina R. Bai, Kathryn T. Stolee, and Tim Menzies. “Fair Enough: Searching for Sufficient Measures of Fairness”, The Transactions on Software Engineering and Methodology. (2022).

- Kewen Peng, **Joymallya Chakraborty**, and Tim Menzies. “xFAIR: Better Fairness via Model-based Rebalancing of Protected Attributes”, The IEEE Transactions on Software Engineering (2022).

- Zhe Yu, **Joymallya Chakraborty**, and Tim Menzies. “FairBalance: Improving Machine Learning Fairness on Multiple Sensitive Attributes With Data Balancing”, The International Conference on Machine Learning (2022).

- Suvodeep Majumder, **Joymallya Chakraborty**, and Tim Menzies. “Communication and Code Dependency Effects on Software Code Quality (an Empirical Analysis of Herbsleb Hypothesis)”, Journal of Systems and Software (2022).

1.6 Lessons Learned

This thesis offers a detailed evaluation of all these methods (see later in this document). In summary:

- **Hyperparameter Optimization** - In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. To the best of our knowledge, this is the first work using hyperparameter optimization in the context of fairness.
- **Class Balancing** - A simple way to fix imbalanced data-sets is simply to balance them, either by oversampling instances of the minority class or undersampling instances of the majority class. We, in this work, come up with a new balancing technique called Fair-SMOTE that not only balances data based on class but also sensitive attributes.
- **Situation Testing** - It is a legally grounded technique for analyzing the discriminatory treatment on an individual. We used the concept of situation testing to find bias in the training data and to evaluate our algorithms.

1.7 Resources

- **Datasets** - This study uses ten real-world datasets from UCI ML repository. All the datasets are publicly available. We described the datasets in section 2.2.3.
- **Source Code** - To facilitate open science and the verification and extension of these results, we have placed all our scripts online [1].

1.8 Structure of this Thesis

The rest of this thesis is structured as follows:

- Chapter 2 describes prior works in the software fairness domain. It also gives description of the datasets and metrics used in this work.
- Chapter 3 describes the root causes of bias. It also answers why just removal of sensitive attributes is not sufficient to achieve fairness.
- Chapter 4 and Chapter 5 introduce two frameworks Fairway and Fair-SMOTE respectively.
- Chapter 6 extends Fair-SMOTE for multiple protected attributes and introduces FairBalance.
- Chapter 7 introduces Fair-SSL where we use semi-supervised learning to achieve fairness.
- Chapter 8 simplifies the procedure of selecting appropriate fairness metrics.

[1] https://github.com/joymallyac/Fairway, https://github.com/joymallyac/Fair-SMOTE,https://github.com/joymallyac/FairSSL

- Chapter 9 explains software fairness from the concept of causality.
- Finally, chapter 10 concludes the thesis and contains the future direction of software fairness research.

CHAPTER

2

BACKGROUND AND RELATED WORK

This chapter describes prior works in the software fairness domain. It also gives description of the datasets and metrics used in this work.

2.1 Background

Machine learning software is increasingly being used to make decisions that affect people's lives. Sadly, there are too many examples of machine learning software exhibiting unfair/biased behavior based on some protected attributes like sex, race, age, marital status. For example:

- Amazon had to scrap an automated recruiting tool as it was found to be biased against women [Amad].
- A widely used face recognition software was found to be biased against dark-skinned women [Ski].
- Google Translate, the most popular translation engine in the world, shows gender bias. "She is an engineer, He is a nurse" is translated into Turkish and then again into English becomes "He is an engineer, She is a nurse" [Cal17a].
- Recidivism assessment models that are used by the criminal justice system have been found to be more likely to falsely label black defendants as future criminals at almost twice the rate as white defendants [Mac].
- YouTube makes more mistakes when automatically generating closed captions for videos with female than male voices [Tat17].

We believe that it is the ethical duty of software researchers and engineers to produce quality software that makes fair decisions, especially for high-stake software that makes important decisions about human lives. That is the main motivation behind this entire work.

2.2 Software Fairness

Bias in Machine Learning models is a well-known topic in the ML community. Recently, the SE community is also showing interest in this area. Large SE industries have started putting more and more importance on ethical issues of ML model and software. IEEE [Iee], the European Union [Eu] and Microsoft [Mic] recently published the ethical principles of AI. In all three of them, it is stated that an intelligent system or machine learning software must be fair when it is used in real-life applications. IBM has launched a software toolkit called AI Fairness 360 [Aifa] which is an extensible open-source library containing techniques developed by the research community to help, detect and mitigate bias in machine learning models throughout the AI application lifecycle. Microsoft has created a research group called FATE [Fat] which stands for Fairness, Accountability, Transparency, and Ethics in AI. Facebook announced they developed a tool called Fairness Flow [Faia] that can determine whether a ML algorithm is biased or not. ASE 2019 has organized the first International Workshop on Explainable Software [Exp] where issues of ethical AI were extensively discussed. German et al. have studied different notions of fairness in the context of code reviews[Ger18]. In summary, the importance of fairness in software is rising rapidly. So far, the researchers have concentrated on two specific aspects -

- Testing AI software model to **find ethical bias**
- Making the model prediction fair by **removing bias**

2.2.1 Finding Ethical Bias

Angell et al. [Ang] commented that software fairness is part of software quality. An unfair software is considered as poor quality software. That is why testing software for bias has become an important part of the software development life-cycle.

- Galhotra et al. created THEMIS [Gal17], a testing-based tool for measuring how much a software discriminates, focusing on causality in discriminatory behavior. THEMIS selects random values from the domain for all the attributes to determine if the system discriminates amongst the individuals.
- Udeshi et al. have developed AEQUITAS [Ude18] tool that automatically discovers discriminatory inputs which highlight fairness violation. It generates test cases in two phases. The first phase is to generate test cases by performing random sampling on the input space. The second phase starts by taking every discriminatory input generated in the first phase as input and perturbing it to generate furthermore test cases.

- The researchers from IBM Research AI India have proposed a new testing method for black-box models [Agg19]. They combined dynamic symbolic execution and local explanation to generate test cases for non-interpretable models.

- In ICSE 2020, Zhang et al. presented how adversarial sampling can be used as a white-box testing tool to test fairness of DNN models [Zha20a].

These all are test case generation algorithms that try to find bias in a trained model. We did not use these methods because along with the model, we also wanted to find bias in the training data. We developed our own testing method based on the concept of situation testing[Zha16c].

2.2.2 Removing Ethical Bias

The prior works in this domain can be classified into three groups depending on the approach applied to remove ethical bias.

- **Pre-processing algorithms**: In this approach, before classification, data is pre-processed in such a way that discrimination or bias is reduced. Kamiran et al. proposed *Reweighing* [Kam12a] method that generates weights for the training examples in each (group, label) combination differently to achieve fairness. Calmon et al. proposed an *Optimized pre-processing* method [Cal17b] which learns a probabilistic transformation that edits the labels and features with individual distortion and group fairness.

- **In-processing algorithms**: This is an optimization approach where the dataset is divided into three sets - train, validation and test set. After learning from training data, the model is optimized on the validation set and finally applied on the test set. Zhang et al. proposed *Adversarial debiasing* [Zha18] method which learns a classifier to increase accuracy and simultaneously reduce an adversary's ability to determine the protected attribute from the predictions. This leads to generation of fair classifier because the predictions cannot carry any group discrimination information that the adversary can exploit. Kamishima et al. developed *Prejudice Remover* technique [Kam12c] which adds a discrimination-aware regularization term to the learning objective of the classifier.

- **Post-processing algorithms**: This approach is to change the class labels to reduce discrimination after classification. Kamiran et al. proposed *Reject option classification* approach [Kam18] which gives favorable outcomes to unprivileged groups and unfavorable outcomes to privileged groups within a confidence band around the decision boundary with the highest uncertainty. *Equalized odds post-processing* is a technique which particularly concentrates on the Equal Opportunity Difference(EOD) metric. Two most cited works in this domain are done by Pleiss et al. [Ple17] and Hardt et al [Har16].

2.2.3 Fairness Terminology & Datasets

Before moving forward, we need to define some specific terms related to machine learning fairness. This work is limited to the binary classification models and tabular data(row-column format). We used ten datasets for our experiments. All the datasets are quite popular in the fairness domain and used by previous SE researchers[Gal17; Ude18]. Table 2.1 contains summarized descriptions of these datasets.

- A class label is called a *favorable label* if its value corresponds to an outcome that gives an advantage to the receiver. Examples include - being hired for a job, receiving a loan.
- *Protected attribute* is an attribute that divides a population into two groups (privileged & unprivileged) that have difference in terms of benefits received. An example of such attribute could be "sex" or "race". These attributes are not universal but are specific to the application.
- *Group fairness* is the goal that based on the protected attribute, privileged and unprivileged groups will be treated similarly. *Individual fairness* is the goal of similar individuals will receive similar outcomes.

Now we describe all the datasets in detail -

- Adult Census Income - This dataset contains records of 48,842 people. The class label is yearly income [Adu]. It is a binary classification dataset where the prediction task is to determine whether a person makes over 50K a year. There are fourteen attributes. Among them two are protected attributes.

  For protected attribute "sex", "Male" is privileged, and "Female" is unprivileged. For protected attribute "race", "White" is privileged, and "Non-white" is unprivileged. Here Favorable label is "High income (> 50K)" and unfavorable label is "Low income (<= 50K)".

- Compas - This is a dataset containing criminal history, demographics, jail and prison time, and COMPAS (which stands for Correctional Offender Management Profiling for Alternative Sanctions) risk scores for defendants from Broward County [Com]. The dataset contains 7,214 rows and twenty-eight attributes. Among them there are two protected attributes.

  For protected attribute "sex", "Female is privileged", and "Male" is unprivileged. For protected attribute "race", "Caucasian" is privileged, and "Not Caucasian" is unprivileged. Favorable label is "Did not recidivate" and unfavorable label is "Did recidivate".

- German Credit Data - This dataset contains records of 1,000 people and binary class labels (Good Credit or Bad Credit) [Ger]. There are twenty attributes. Among them one is protected.

  For protected attribute "sex", "Male" is privileged, and "Female" is unprivileged. Favorable label is "Good credit" and unfavorable label is "Bad credit".

- Default Credit - There are 30,000 records of default payments of people from Taiwan [Def]. Binary class label is Default Payment “Yes” or “No”. There are twenty-three attributes. Among them one is protected.

  For the attribute “sex”, “Male” is is privileged, and “Female” is unprivileged. Favorable label is “Yes” and unfavorable label is “No”.

- Heart Health - The Heart Dataset from the UCI ML Repository contains fourteen features from 297 adults [Hea]. The goal is to accurately predict whether or not an individual has heart condition.

  The protected attribute is “age”. People under mean age are privileged and above mean age are unprivileged.

- Bank Marketing - The data is related with direct marketing campaigns of a Portuguese banking institution [Ban]. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (’yes’) or not (’no’) subscribed.

  The protected attribute is “age”. People above mean age are privileged and under mean age are unprivileged. Favorable label is “Term deposit - Yes” and unfavorable label is “Term deposit - No”.

- Home Credit - This dataset contains data related to loan applications for individuals who do not get loan from the traditional banks [Hom]. The target feature is to predict whether an individual who can repay the loan, get the application accepted or not.

  The protected attribute is “sex”; “Male” is privileged, and “Female” is unprivileged.

- Student Performance - This dataset contains student achievements in secondary education of two Portuguese schools [Stu]. The target attribute is the GPA or grade of final year.

  The protected attribute is “sex”; “Male” is privileged, and “Female” is unprivileged.

- MEPS 15, 16 - The Medical Expenditure Panel Survey (MEPS) is a set of large-scale surveys of families and individuals, their medical providers, and employers across the United States [Mep]. MEPS is the most complete source of data on the cost and use of health care and health insurance coverage. We used two versions of MEPS data for 2015 and 2016 respectively.

  For protected attribute “race”, “White” is privileged, and “Non-white” is unprivileged.

2.2.4 Fairness Measures

Here we describe the metrics needed to measure fairness. In 2018, Verma et al. did an empirical study and found out there are 20 different fairness metrics available [Ver18]. Today, IBM AIF360 contains definitions of more than fifty fairness metrics [Aifb]. Our study is limited to binary classification. We

Table 2.1 Details of the datasets used in this research.

Dataset	#Rows	#Cols	Protected Attribute		Class Label	
			Privileged	Unprivileged	Favorable	Unfavorable
Adult Census Income	48,842	14	Sex-Male Race-White	Sex-Female Race-Non-white	High Income	Low Income
Compas	7,214	28	Sex-Female Race-Caucasian	Sex-Male Race-Not Caucasian	Did not reoffend	Reoffended
German Credit	1,000	20	Sex-Male	Sex-Female	Good Credit	Bad Credit
Default Credit	30,000	23	Sex-Male	Sex-Female	Default Payment-Yes	Default Payment-No
Heart Health	297	14	Age-Young	Age-Old	Not Disease	Disease
Bank Marketing	45,211	16	Age-Old	Age-Young	Term Deposit - Yes	Term Deposit - No
Home Credit	37,511	240	Sex-Male	Sex-Female	Approved	Rejected
Student Performance	1,044	33	Sex-Male	Sex-Female	Good Grade	Bad Grade
MEPS15, MEPS16	35,428	1,831	Race-White	Race-Non-white	Good Utilization	Bad Utilization

decided to use four metrics widely used by prior researchers [Cha20b; Bis20; Har16] in the context of binary classification.

- **Average Odds Difference (AOD)**: Average of difference in False Positive Rates(FPR) and True Positive Rates(TPR) for unprivileged and privileged groups [Bel18].

$$TPR = TP/(TP+FN), FPR = FP/(FP+TN)$$

$$AOD = [(FPR_U - FPR_P) + (TPR_U - TPR_P)] * 0.5$$

- **Equal Opportunity Difference (EOD)**: Difference of True Positive Rates (TPR) for unprivileged and privileged groups [Bel18].

$$EOD = TPR_U - TPR_P$$

- **Statistical Parity Difference (SPD)**: Difference between the probability of unprivileged group gets favorable prediction and the probability of privileged group gets favorable prediction [Cal10].

$$SPD = P[\hat{Y} = 1|PA = 0] - P[\hat{Y} = 1|PA = 1]$$

- **Disparate Impact (DI)**: Similar to SPD but instead of the difference of probabilities, the ratio

Table 2.2 Combined Confusion Matrix for Privileged(P) and Unprivileged(U) Groups.

	Predicted No	Predicted Yes	Predicted No	Predicted Yes
	Privileged		Unprivileged	
Actual No	TN_P	FP_P	TN_U	FP_U
Actual Yes	FN_P	TP_P	FN_U	TP_U

is measured [Fel15].

$$DI = P[\hat{Y} = 1|PA = 0]/P[\hat{Y} = 1|PA = 1]$$

All four of them are computed using input and output to a classifier. For AOD, EOD and SPD, a value of 0 implies that both groups have equal benefit and a higher/lower value shows benefit difference between privileged and unprivileged group. We measured absolute value of these three metrics. DI is a ratio and there is no bias when value of DI is 1. For readability, While showing results we compute abs(1 - DI) so that all four fairness metrics are lower the better (0 means no bias).

2.2.5 Performance Measures

We have used five well-known metrics for measuring performance of the prediction.

- **Recall**: Ratio of true positives and total positives.

$$Recall = TP/P = TP/(TP+FN)$$

- **False alarm**: Ratio of false positives and total negatives.

$$FA = FP/N = FP/FP+TN$$

- **Accuracy**: Ratio between sum of true positives and true negatives with total number of samples.

$$Accuracy = (TP+TN)/(TP+FP+TN+FN)$$

- **Precision**: Ratio of True Positives and predicted positives.

$$Precision = TP/(TP+FP)$$

- **F1 Score**: Harmonic mean of precision and recall.

$$F1 = 2*(Precision*Recall)/(Precision+Recall)$$

Recall, accuracy, precision, F1 score are higher the better and false alarm is lower the better. The value of all of them lie between 0 and 1.

In the first chapter, we have described what software fairness is and what problem we are trying to solve. In this chapter, we have showcased the prior works in this field. We also described the datasets and metrics needed. Now we will find out the source of bias. Where does the bias come from? We will find the answers in the next chapter.

CHAPTER

3

ROOT CAUSES OF BIAS

Prior fairness works tried to find bias or remove bias. But the main reason behind bias still remains a mystery. That is why practitioners find it really hard to choose an appropriate bias mitigation algorithm. Figure 3.1 shows how challenging it is for a software engineer to see so many algorithms for finding bias and removing bias but none of them explains where bias comes from. In this chapter, we will find out the root causes of the bias. Later, we will remove the causes to figure out whether that removes bias too or not.

3.1 Why not Remove the Protected Attributes?

This section describes one of the methods we explored, before arriving at the advanced strategies.

When we think of prediction model discriminating over a protected attribute, the first solution

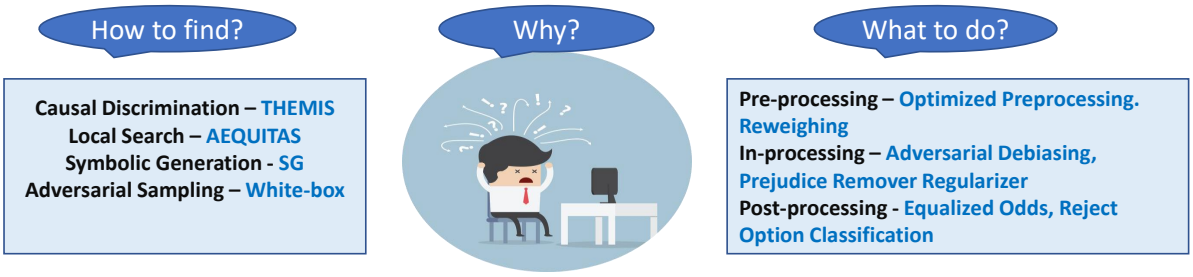

Figure 3.1 Many tools try to find or explain or mitigate bias. We address all three problems, at the same time.

that comes to mind is that why not train the model without that protected attribute. Being novice in fairness domain, we tried that for the ten datasets. Two of the datasets have two protected attributes (Adult, Compas - Sex, Race) and other datasets have only one protected attribute. We removed the protected attribute column from the train and test data so that the model has no information about that attribute. Surprisingly, there was minimal change in bias metrics even after that.

Brun et al. have mentioned one reason behind this surprising result. They mentioned that if there is high correlation between attributes of the dataset, then even after removing the protected attribute, the bias stays [Faic]. In 2016, Amazon created a model for same-day delivery service offered to Prime users around the major US cities[Amaa]. But the model turned out to be highly discriminatory against black neighborhoods. While training this model, "Race" attribute was not used but the model became biased against a certain "Race" because the "Zipcode" attribute highly correlates with "Race". The training data had "Zipcode" and the model induced "Race" from that. Initially, we also thought maybe correlation is the reason for our datasets also. But when we checked for the correlation between attributes, we found that bias is not coming from the correlation.

For the datasets we are using here, the bias does not directly come from the protected attribute column. That is why just removal of protected attribute does not help. Then the question comes from where bias comes. How does a machine learning model acquire bias? We will find the answers soon.

In this work, we postulate that data has a history and that history introduces bias. For example, consider a team of humans labeling data as "risky loan applicant" or otherwise. If that team has any biases (conscious or otherwise) against certain social groups, then the bias of those humans introduces improper/unfair data labels.

Another way history can affect the data is *selection bias*. Suppose we are collecting package data relating to the kinds of items shipped via Amazon to a particular suburb. Suppose there is some institutional bias that tends to result in low annual incomes for persons in certain suburbs[1]. The delivery data collected from those suburbs would contain all the political and economic biases that tend to perpetuate lower socio-economic neighborhoods.

The next two sections explore the intuitions of the last two paragraphs in more detail. Specifically, we will look into data imbalance and improper data labeling. This in turn will lead to the Fair-SMOTE algorithm that deletes biased labels and balances all distributions between positive and negative examples.

3.2 Data Imbalance

When a classification model is trained on imbalanced data, the trained model shows bias against a certain group of people. We mention that such imbalances are quite common. Figure 3.2 displays distributions within our datasets. Note that, in most cases, we see a *class imbalance*; i.e the number

[1] E.g. If politicians spend less money on schools in a poorer district; then that district has (a) fewer exceptional schools; (b) fewer graduates with job skills for high-paying jobs in high demand; (c) consistently lower incomes from generation to generation.

of observations per class is not equally distributed. Further, we see that *the imbalance is not just based on class, but also on protected attribute*.

For example, consider the Adult dataset. Here we are predicting the annual income of a person. There are two classes. "High income" (≥ $50,000) which is *favorable label* and "low income" (< $50,000) which is *unfavorable label*. The first grouped bar in Figure 3.2 has two bars grouped together. The first bar is for "high income" class (24%) and the second bar is for "low income" (76%) class. It is clear that the number of instances for "low income" is more than three times of instances for "high income". This is an example of class imbalance. Now, both the bars are subdivided based on *protected attribute* "sex" ("male" and "female"). For "high income" or *favorable label*, 86% instances are "male" and only 14% are female. For "low income" or *unfavorable label*, 60% instances are "male" and 40% are female. Overall, this dataset contains more examples of "male" (privileged) getting *favorable label* and female (unprivileged) getting *unfavorable label*.

Class imbalance is a widely studied topic in the ML domain. There are mainly two ways of balancing imbalanced classes:

- Oversample the minority class;
- or undersample the majority class.

We want to see how various class balancing techniques affect fairness. In Table 3.1 we are showing the results for five most commonly used class balancing techniques on Adult dataset. One of them is undersampling (RUS- Random Under Sampling), other three (ROS- Random Over Sampling, SMOTE- Synthetic Minority Over Sampling Technique [Cha02] and KMeans-SMOTE [Dou18]) are oversampling techniques. Table 3.1 shows values of nine metrics - first five of them are performance metrics and last four of them are fairness metrics. We used logistic regression model here.

The important observation here is all four of the class balancing techniques are increasing bias scores mean damaging fairness (lower is better here). To better understand this, see Figure

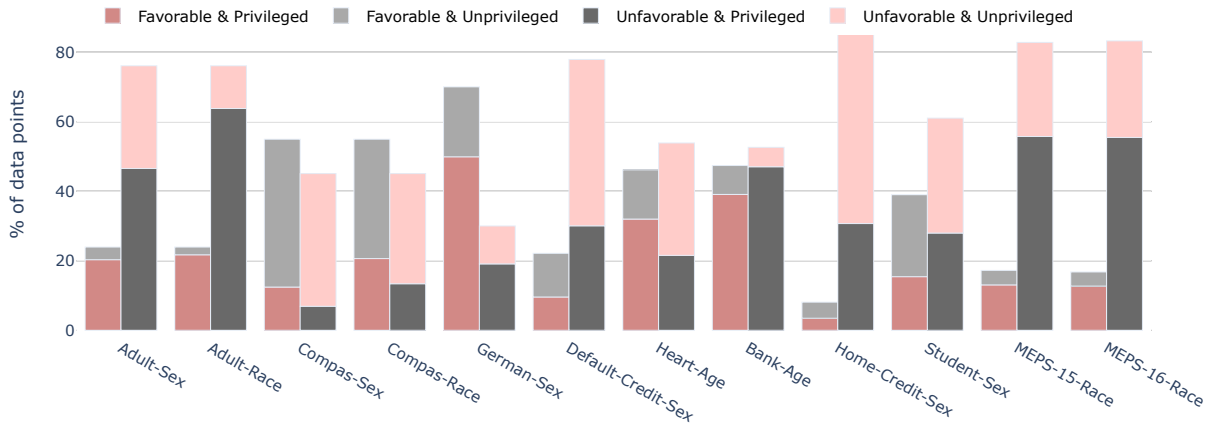

Figure 3.2 Most of the datasets showing not only class imbalance but also imbalance based on the protected attribute.

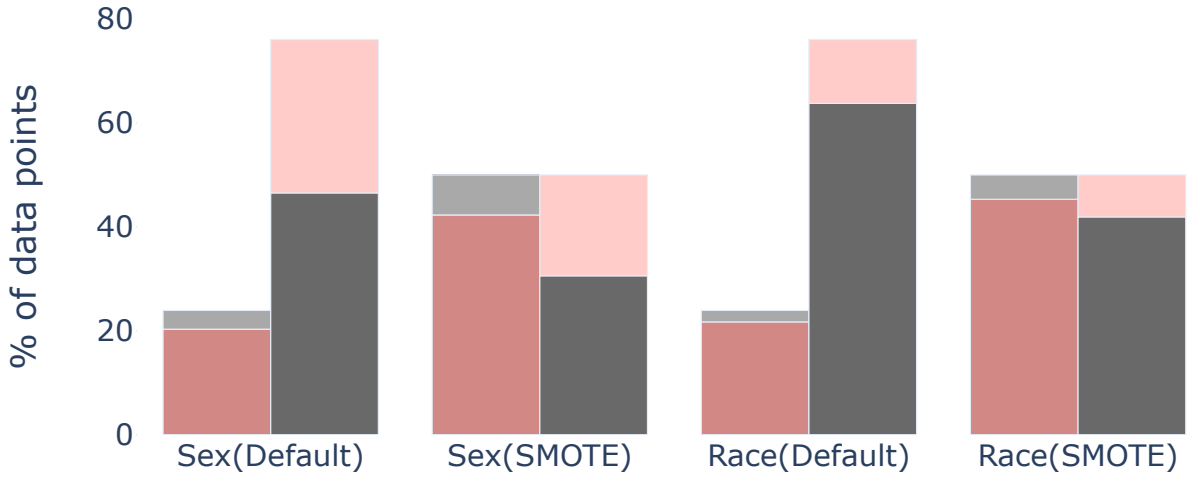

Figure 3.3 Effects of SMOTE class balancing technique on Adult dataset for two protected attributes "sex" and "race".

Table 3.1 Effects of various class balancing techniques on Adult dataset (note: for the metrics with '+' more is better and for the metrics with '-' less is better). For each metric, cell with the best score is highlighted.

	Recall(+)	False alarm(-)	Precision(+)	Accuracy(+)	F1 Score(+)	AOD(-)	EOD(-)	SPD(-)	DI(-)
Default	0.42	0.07	0.69	0.83	0.54	0.12	0.24	0.21	0.56
RUS	0.74	0.23	0.48	0.76	0.59	0.09	0.36	0.37	0.61
ROS	0.74	0.24	0.47	0.75	0.59	0.08	0.34	0.35	0.64
SMOTE	0.70	0.25	0.49	0.70	0.64	0.17	0.37	0.33	0.58
KMeans-SMOTE	0.73	0.24	0.48	0.74	0.58	0.08	0.35	0.36	0.62

3.3. It gives a visualization of using SMOTE [Cha02] on Adult dataset. SMOTE generates synthetic samples for minority class data points to equalize two classes. Suppose a data point from minority class is denoted as X where $x_1, x_2, .., x_n$ are the attributes and its nearest neighbor is $X^{'}$ $(x_1^{'}, x_2^{'}, .., x_n^{'})$. According to SMOTE, a new data point Y $(y_1, y_2, .., y_n)$ is generated by the following formula:

$$Y = X + rand(0,1) * (X - X^{'})$$

SMOTE definitely balances the majority class and minority class but it damages the protected attribute balance even more. Thus Figure 3.3 explains the results of Table 3.1.

We get the idea that traditional class balancing methods improve performance of the model but damage fairness (usually). The reason is *these techniques randomly generate/discard samples just to equalize two classes and completely ignore the attributes and hence damage the protected attribute balance even more*.

3.3 Improper Data Labeling

Some prior works [Das20; Jia19; Sim20] argue that improper labeling could be a reason behind bias. We used the concept of *situation testing* to validate how labeling can affect fairness of the model.

Situation testing is a research technique used in the legal field [Usa] where decision makers' candid responses to applicant's personal characteristics are captured and analyzed. For example:

- A pair of research assistants (a male and a female with almost equivalent qualities) undergo the same procedure, such as applying for a job.
- Now the treatments they get from the decision-maker are analyzed to find discrimination.

Situation testing as a legal tactic has been widely used both in the United States [Usa] and the European Union [Eus]. Luong et al. [Luo11] first used the concept of situation testing in classification problems. They used K-NN approach to find out similar individuals getting different outcomes to find discrimination. Later, Zhang et al. [Zha16c] used causal bayesian networks to do the same. The core idea of our situation testing is much simpler:

- Flip the value of protected attribute for every data point.
- See whether prediction given by the model changes or not.

In our implementation, a logistic regression model is trained first and then all the data points are predicted. After that, the protected attribute value for every data point is flipped (e.g. male to female, white to non-white) and tested again to validate whether model prediction changes or not. If the result changes, that particular data point fails the situation testing.

Table 3.2 shows the median of ten runs for all the datasets. We see all the datasets more or less contain these kinds of biased data points. That means we have found *biased labels*. Note that, we use logistic regression model for *situation testing* but any other supervised model can be chosen. We picked logistic regression because it is a simple model and can be trained with a very low amount of data (compared to DL models) [Buj18]. Choosing a different model may change the outcome a bit. We will explore that in future.

In this chapter, we have found out that two major reasons of bias are imbalanced training data and unfair labels. That said, bias comes from the training data. From now on, we will use various strategies to remove bias from the training data and optimize models for better performance. The next three chapters will describe our three different approaches to achieve fair and better prediction results.

Table 3.2 Percentage of data points failing situation testing for 10 datasets.

	Adult (Sex)	Adult (Race)	Compas (Sex)	Compas (Race)	German (Sex)	Default Credit (Sex)	Heart-Health (Age)	Bank Marketing (Age)	Home Credit (Sex)	Student (Sex)	MEPS-15 (Race)	MEPS-16 (Race)
% of Rows failed	11%	3%	18%	8%	6%	6%	8%	19%	17%	4%	4%	4%

Part 2 - Fixing the Problem

CHAPTER

4

FAIRWAY STUDY (PRE-PROCESSING + HYPERPARAMETER OPTIMIZATION)

This chapter describes our first bias mitigation framework - Fairway. It is a combination of data pre-processing and hyperparameter optimization. This study was published on ESEC/FSE 2020 [Cha20b]. Note that the methods of this chapter were superseded by our subsequent FSE'21 paper (see Chapter 5). Hence we conclude that hyperparameter optimization (which is a common method for fairness mitigation) is not the best way to address our problem.

4.1 Introduction

In the recent software engineering literature, we have found some works to identify bias in machine learning software systems [Ang; Agg19]. But there is no prior work done to explain the reason behind the bias and also removing the bias from the software. We see some recent works from the ML community to mitigate ML model bias. All of these works trust the ground truth or the original labels of the training data. But any human being or algorithm can make biased decisions and introduce biased labels. For example, white male employees were given higher priority to be selected for company leadership by human evaluators [Forb]; COMPAS Recidivism algorithm was found biased against black people[Pro]. If this kind of biased data is used for machine learning model training, then trusting the ground truth could introduce unfair decisions in future. So, training data validation, testing model for bias and bias mitigation are equally important. This work covers all the concerns. The idea of *Fairway* comes from two research directions:

- Chen et al. mentioned that a model acquires bias from training data[Che18]. They bolstered on data collection process and training data sampling. Their work motivated us to find bias in the training data rather than the model.

- Berk et al. have stated that achieving fairness has a cost [Ber17a]. Most of the bias mitigation algorithms damage the performance of the prediction model while making it fair. This is called *accuracy-fairness trade-off*. When trading off competing goals, it is useful to apply multiobjective optimization. While doing so one objective is to reduce bias or achieve fairness and another objective is to keep the performance of the model similar.

Drawing inspiration from both these works, we propose a new algorithm, "Fairway", which is a combination of pre-processing and in-processing methods. Following the motivation of Chen et al, we evaluate the original labels of the training data and identify biased data points which can eventually make the machine learning model biased. Then following the idea of Berk et al, we apply multiobjective optimization approach to keep the model performance same while making it fair. The combination of these two approaches makes Fairway a handy tool for bias detection and mitigation. Overall, this work makes the following contributions:

- We explain how a machine learning model acquires bias from training data.

- We find out the specific data points in training data that cause the bias. Thus, this work includes finding bias in AI software.

- We are first to combine two bias mitigation approaches - pre-processing (before model training) and in-processing(while model training). This combined method, *Fairway*, performs better than each individual.

- Our results show that we can achieve fairness without much damaging the performance of the model.

- Our Fairway replication package is publicly available on GitHub[1] and figshare[Cha20a]. This last point is not so much a research contribution but a systems contribution since it enables other researchers to repeat/confirm and perhaps even refute/improve our results.

4.2 Methodology

Fairway consists of two major components. In the first part, we remove biased data points from the training data and in the second part we use multi-objective optimization approach to find optimal parameter settings of the model. We will describe both the parts here.

[1] https://github.com/joymallyac/Fairway

4.2.1 Removal of Ambiguous (Biased) Data Points

Depending upon the protected attribute, there is a privileged group and an unprivileged group in each dataset. Which group is privileged and which group is unprivileged depend on the application. For example:

- In credit card applications, “Male” might be considered privileged and “Female” as unprivileged;
- In criminal prediction, “White” people might be considered privileged and “non-white” as unprivileged.

In this step, we try to find and remove the data points which are responsible for creating the bias based on the protected attribute. We call these data points the **ambiguous** data points.

Fig. 4.1 describes the approach we applied to find out the ambiguous data points depending on the protected attribute. We divide the training data into two groups based on the protected attribute - privileged and unprivileged. Then we train two separate models on those two groups. Once we get the two trained models, for all the training data points, we check the prediction of these two models:

- If the prediction matches in both cases, the data point being examined is unbiased.
- If two models contradict each other for a data point, there is a possibility of this data point being biased, this is an ambiguous data point. We remove that data point from training data. Later we will describe why this works and how to validate.

We call this data cleaning process as “Bias Removal” from training data. Once we are done removing the probable biased data points, we train a new model on the rest of the training data and make prediction using that model. Table 4.1 shows the total number of rows in each dataset and the number of rows we removed. We see that at most we lose 15% of training data after bias removal step. Later we will show that this does not affect much of the performance of the prediction model.

We remove the ambiguous(bias causing) data points by constructing two separate logistic regression models conditioned upon the protected attribute of the dataset. Let’s assume the original data points are denoted as X where $x_1, x_2, x_3, ..., x_n$ are the attributes of the dataset and the protected attribute is denoted as s (s = x_k, where k is a number between 1 to n) and $\hat{y}$ is the model prediction. The original dataset is further divided into subsets based on the values of a protected attribute, in this case, $X_1 \subset X \forall s = 1$ and $X_2 \subset X \forall s = 0$. We use these two subsets to build two logistic regression models such as -

$$p(\hat{y} = 1|s = 1) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_{n-1} x_{n-1} \tag{4.1}$$

$$p(\hat{y} = 1|s = 0) = \beta_0^{'} + \beta_1^{'} x_1 + \beta_2^{'} x_2 + ... + \beta_{n-1}^{'} x_{n-1} \tag{4.2}$$

$$f_1(x) = \log_e \frac{p(\hat{y} = 1|s = 1)}{p(\hat{y} = 0|s = 1)} \tag{4.3}$$

$$f_2(x) = \log_e \frac{p(\hat{y}=1|s=0)}{p(\hat{y}=0|s=0)} \tag{4.4}$$

Next, we use these logistic regression models to check for each training data point, by retaining the data points where

$$\forall x \in X \left(f_1(x_1) == f_2(x_1) \right)$$

This results in retaining only the data points where there is no contradiction about the models' outcome irrespective of data distribution conditioned upon the protected attribute, thus removing the data points which add ambiguity to the model and introduce bias into the model's prediction.

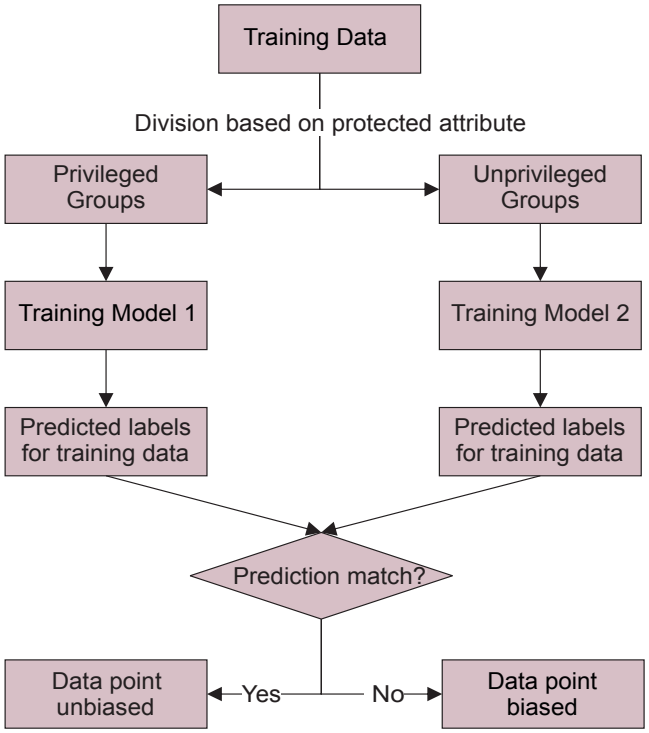

Figure 4.1 Pre-processing technique for bias removal from training data

4.2.2 Model Optimization

IBM has created a GitHub repo to combine some promising prior works on the fairness domain[Aifa]. The results show that most of the prior methods damage the performance of the model while making it fair. So, prediction performance and fairness are competitive goals[Ber17a]. When there is a trade-off between competing performance goals, multi-objective optimization is the way to explore the goal space. In our case, the goal of such optimizer would be to make the model as fair as possible while also not degrading other performance measures such as recall or false alarm.

To explore such multiobjective optimization, we divided the dataset into three groups - Training (70%), Validation (15%) and Test (15%)[Sev]. During the pre-processing step, we removed biased data points from the training set. After that, Logistic Regression model is trained on the training set with the standard default parameters[2]. Then we used the FAIR_FLASH algorithm (discussed below) to find out the best set of parameters to achieve optimal value of four metrics (Higher Recall, Lower False Alarm, Lower AOD, and Lower EOD) on the validation set. Finally, the tuned model is applied on the test set.

Nair et al. proposed FLASH [Nai18], a novel optimizer, that utilizes sequential model-based optimization(SMBO). The concept of SMBO is very simple. It starts with "What we already know about the problem" and then decides "what should we do next". The first part is done by a machine learning model and the second part is done by an acquisition function. Initially, a few points are randomly selected and measured. These points along with their performance measurements are used to build a model. Then the model is used to predict the performance measurements of other unevaluated points. This process continues until a stopping criterion is reached. FLASH improves over traditional SMBO as follows:

- FLASH models each objective as a separate Classification and Regression Tree (CART) model. Nair et al. report that the CART algorithm can scale much better than other model constructors (e.g. Gaussian Process Models).
- FLASH replaces the actual evaluation of all combinations of parameters(which can be a very slow process) with a *surrogate evaluation*, where the CART decision trees are used to guess the objective scores (which is a very fast process). Such guesses may be inaccurate but, as shown by Nair et al., such guesses can rank guesses in (approximately) the same order as that generated by other, much slower, methods [Nai17].

[2] In Scikit-Learn, those details are C=1.0, penalty='l2', solver='liblinear', max_iter=100.

Table 4.1 #Rows = Total number of Rows, #Dropped Rows = Total number of rows detected as ambiguous(biased)

Dataset	Protected Attribute	#Rows	#Dropped Rows	% of Rows Dropped
ADULT	Sex Race	48,842	6,178 2,315	12.6 4.7
COMPAS	Sex Race	7,214	1,128 724	15.6 10.0
DEFAULT CREDIT	Sex	30,000	505	1.7
HEART HELTH	Age	297	32	10.8
GERMAN	Sex	1,000	38	3.8

FLASH was invented to solve software configuration problem and it performed faster than more traditional optimizers such as Differential Evolution[Sto97b] or NSGA-II[Deb02]. For our work, we modified FLASH and generated FAIR_FLASH that seeks best parameters for *Logistic regression* model with four goals - higher recall, lower false alarm, lower AOD, lower EOD. Algorithm 1 shows the pseudocode of FAIR_FLASH. It has two layers - one learning layer and one optimization layer. When training data arrives, the estimator in the learning layer is being trained, and the optimizer in optimizing layer provides better parameters to the learner to help improve the performance of estimators. Such trained learner is evaluated on the validation data afterward. Once some stopping criteria is met, the generated learner is then passed to the test data for final testing.

Algorithm 1: Pseudocode of FAIR_FLASH inspired from [Nai18]

```
def FAIR_FLASH():
    # pick a number of data into build_pool, evaluate the build_pool,
    # and put the rest into rest_pool
    while life > 0:
        # build CART model by using build_pool
        next_point = max(model.predict(rest_pool))
        build_pool += next_point
        rest_pool −= next_point
        if model.evaluate(next_point) < max (build_pool):
            life −= 1
    return max(build_pool)
```

In summary, Fairway consists of two parts - bias removal from training data and model optimization to make trained model fair. Fig. 4.2 shows an overview of the method.

4.3 Evaluation

Before moving to the results, we need to describe the statistical test we used for comparison purpose.

4.3.1 Statistical Tests

While comparing Fairway with other techniques, we use the Scott-Knott test [Gho15; Mit13] to compare two distributions. Scott-Knott is a recursive bi-clustering algorithm that terminates when the difference between the two split groups is not significant. Scott-Knott searches for split points that maximize the expected value of the difference between the means of the two resulting groups. If a group l is split into two groups m and n, Scott-Knott searches for the split point that maximizes

$$E[\Delta] = |m|/|l|(E[m]-E[l])^2 + |n|/|l|(E[n]-E[l])^2$$

where $|m|$ is the size of group m. The result of the Scott-Knott test is ranks assigned to each result set; higher the rank, better the result. Scott-Knott ranks two results the same if the difference between the distributions is not significant.

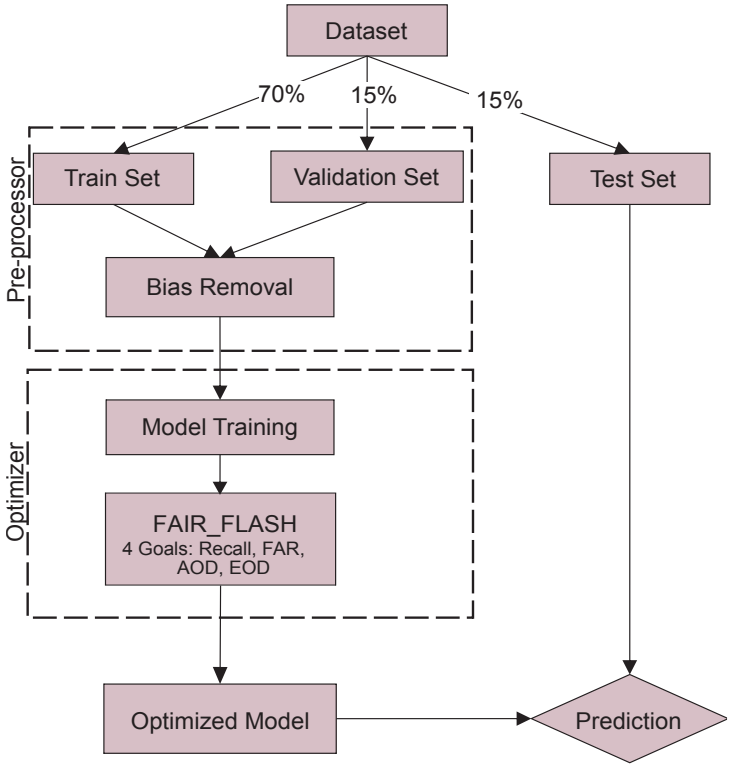

Figure 4.2 Block diagram of Fairway. For details on FAIR_FLASH, see Algorithm 1.

4.3.2 Results

Our results are structured around six research questions. For all the results, we repeated our experiments ten times with data shuffling and we report the median.

RQ1. What is the problem with just using standard learners?

The premise of the work is our methods offer some improvement over common practices. To justify that we first need to show that there are open issues with standard methods. We trained a logistic regression model with default scikit-learn parameters and tested on the five datasets. The “Orange” column in Fig. 4.3 shows the results achieved using that model. The recall is higher the better, and false alarm, AOD, EOD are lower the better. Recall and False alarm are showing the prediction performance of the model. The high value of fairness metrics(AOD, EOD) in all five datasets signifies that model prediction is not fair means depending upon protected attribute, privileged group is getting advantage over unprivileged group. We treat this results as baseline for our experiment. We need to make the prediction fair without damaging the performance much.

RQ2. How well does Pre-processing improve the results?

Fairway is a two-part procedure- data pre-processing (ambiguity removal) and learner optimiza-

tion(FAIR_FLASH). It is reasonable to verify the contribution of both parts. Accordingly **RQ2** tests the effects of just doing ambiguity removal.

Before training Logistic regression model, training data was cleaned to remove ambiguous data points(having improper labels) using the approach mentioned in section 4.2.1. Table 4.1 shows this step causes loss of maximum 15% of the training data. After that logistic regression model was trained on remaining data points and tested. The "Blue" column in Fig. 4.3 shows the results achieved using that model. We see minor damage in recall for some cases and significant improvement in case of fairness metrics (lower AOD, EOD). It is evident that pre-processing the data before model training makes the model prediction fairer.

RQ3. How well does Optimization improve the results?

Moving on from **RQ2**, the third research question is to check the effect of just optimization(no pre-processing).

To do that, we tuned the Logistic regression model parameters using FAIR_FLASH to optimize the model for higher recall, lower false alarm and lower fairness metrics(AOD, EOD). Then the tuned model was used for prediction. The "Green" column in Fig. 4.3 shows the results achieved using that

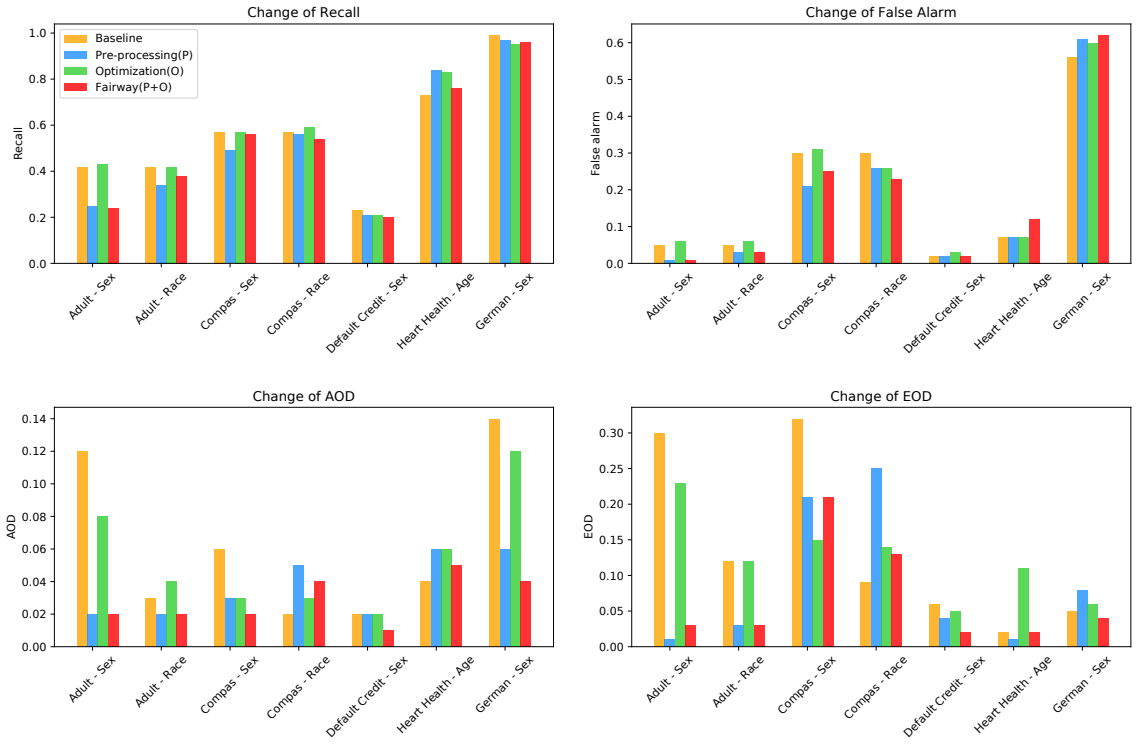

Figure 4.3 Performance and fairness metrics for (a) default state in Orange; (b) after pre-processing in Blue; (c) after just optimization in Green; and (d) after performing pre-processing + optimization in Red. In these charts, *higher* recalls are *better* while for all other scores, *lower values* are *better*.

Table 4.2 Results for RQ4, RQ5 (learner= Logistic Regression). In this table. "Default" denotes off-the-shelf logistic regression; OP is Calmon et al.'s Optimized Pre-processing from NIPS'17 [Cal17b]; Fairway is the algorithm introduced by this work. Cells show medians for 10 runs. Here, the darker pink cells show top rank (note: for the metrics with '+' more is better and for the metrics with '-' less is better). The lighter pink cells show rank two; white shows lowest rank (worst performance). Rankings were calculated via the Scott-Knott test (§4.3.1)

Dataset	Protected Attribute	Algorithms	Recall (+)	False alarm (-)	Precision (+)	Accuracy (+)	F1 Score (+)	AOD (-)	EOD (-)	SPD (-)	DI (-)
Adult Census Income	Sex	Default	**0.42**	**0.07**	**0.69**	**0.83**	**0.54**	**0.12**	**0.24**	**0.21**	**0.56**
		OP	**0.41**	**0.09**	**0.61**	**0.76**	**0.51**	**0.04**	**0.03**	**0.04**	**0.14**
		Fairway	**0.25**	**0.04**	**0.70**	**0.72**	**0.42**	**0.02**	**0.03**	**0.01**	**0.11**
Adult Census Income	Race	Default	**0.42**	**0.05**	**0.69**	**0.81**	**0.52**	**0.06**	**0.15**	**0.16**	**0.52**
		OP	**0.38**	**0.06**	**0.66**	**0.78**	**0.48**	**0.03**	**0.02**	**0.05**	**0.21**
		Fairway	**0.36**	**0.04**	**0.70**	**0.73**	**0.44**	**0.02**	**0.03**	**0.06**	**0.32**
Compas	Sex	Default	**0.73**	**0.38**	**0.66**	**0.64**	**0.61**	**0.05**	**0.14**	**0.18**	**0.28**
		OP	**0.71**	**0.36**	**0.64**	**0.62**	**0.60**	**0.04**	**0.05**	**0.06**	**0.09**
		Fairway	**0.56**	**0.22**	**0.57**	**0.58**	**0.58**	**0.03**	**0.03**	**0.06**	**0.08**
Compas	Race	Default	**0.69**	**0.39**	**0.65**	**0.64**	**0.68**	**0.05**	**0.11**	**0.12**	**0.21**
		OP	**0.68**	**0.33**	**0.63**	**0.62**	**0.67**	**0.03**	**0.06**	**0.06**	**0.12**
		Fairway	**0.55**	**0.21**	**0.58**	**0.58**	**0.56**	**0.02**	**0.04**	**0.07**	**0.07**
German Credit	Sex	Default	**0.94**	**0.81**	**0.72**	**0.76**	**0.82**	**0.11**	**0.08**	**0.14**	**0.15**
		OP	**0.75**	**0.73**	**0.71**	**0.73**	**0.71**	**0.04**	**0.05**	**0.06**	**0.12**
		Fairway	**0.78**	**0.76**	**0.68**	**0.65**	**0.73**	**0.05**	**0.04**	**0.07**	**0.11**
Default Credit	Sex	Default	**0.25**	**0.07**	**0.7**	**0.78**	**0.34**	**0.05**	**0.08**	**0.06**	**0.45**
		OP	**0.28**	**0.06**	**0.65**	**0.70**	**0.32**	**0.01**	**0.02**	**0.03**	**0.19**
		Fairway	**0.21**	**0.04**	**0.67**	**0.67**	**0.33**	**0.01**	**0.04**	**0.03**	**0.22**
Heart Health	Age	Default	**0.72**	**0.2**	**0.74**	**0.72**	**0.74**	**0.11**	**0.13**	**0.32**	**0.44**
		OP	**0.69**	**0.19**	**0.72**	**0.69**	**0.69**	**0.04**	**0.06**	**0.09**	**0.14**
		Fairway	**0.65**	**0.21**	**0.69**	**0.67**	**0.68**	**0.05**	**0.05**	**0.04**	**0.18**
Bank Marketing	Age	Default	**0.73**	**0.21**	**0.76**	**0.77**	**0.77**	**0.14**	**0.22**	**0.24**	**0.51**
		OP	**0.72**	**0.20**	**0.74**	**0.75**	**0.75**	**0.05**	**0.05**	**0.12**	**0.24**
		Fairway	**0.71**	**0.17**	**0.73**	**0.71**	**0.73**	**0.04**	**0.03**	**0.13**	**0.25**
Home Credit	Sex	Default	**0.31**	**0.18**	**0.28**	**0.86**	**0.29**	**0.07**	**0.06**	**0.08**	**0.59**
		OP	**0.3**	**0.17**	**0.30**	**0.83**	**0.31**	**0.02**	**0.04**	**0.04**	**0.16**
		Fairway	**0.28**	**0.12**	**0.25**	**0.65**	**0.26**	**0.02**	**0.04**	**0.03**	**013**
Student Performance	Sex	Default	**0.81**	**0.06**	**0.85**	**0.88**	**0.83**	**0.06**	**0.05**	**0.06**	**0.12**
		OP	**0.79**	**0.06**	**0.83**	**0.83**	**0.82**	**0.03**	**0.02**	**0.03**	**0.06**
		Fairway	**0.76**	**0.05**	**0.81**	**0.84**	**0.84**	**0.03**	**0.02**	**0.04**	**0.07**
MEPS - 15	Race	Default	**0.36**	**0.03**	**0.68**	**0.85**	**0.45**	**0.04**	**0.05**	**0.08**	**0.36**
		OP	**0.35**	**0.04**	**0.66**	**0.83**	**0.44**	**0.04**	**0.02**	**0.06**	**0.15**
		Fairway	**0.35**	**0.04**	**0.42**	**0.77**	**0.41**	**0.03**	**0.02**	**0.04**	**0.12**
MEPS - 16	Race	Default	**0.35**	**0.05**	**0.65**	**0.85**	**0.44**	**0.04**	**0.1**	**0.07**	**0.43**
		OP	**0.34**	**0.04**	**0.65**	**0.83**	**0.44**	**0.05**	**0.02**	**0.05**	**0.12**
		Fairway	**0.32**	**0.04**	**0.55**	**0.76**	**0.42**	**0.03**	**0.02**	**0.04**	**0.16**

model. We see that in cases of prediction performance(recall, false alarm) it performs similar or better than pre-processing but in case of fairness metrics(AOD, EOD), pre-processing does better. So, optimized learner is significantly better than baseline learner but combining pre-processing may perform even better.

RQ4. How well does Fairway improve the results?

Our fourth research question explores the effect of Fairway which is a combination of pre-processing and optimization.

The "Red" column in Fig. 4.3 shows the results achieved after applying Fairway. Fairway is performing better than pre-processing and optimization in most of the cases. For example:

- In case of Adult dataset, for the protected attribute race, Fairway achieves almost similar recall with optimization but much better in the other three metrics.
- In case of Default Credit dataset, for the protected attribute sex, Fairway is providing best results for all four metrics.

In some cases, recall is slightly damaged. But overall, Fairway is making the model fair without much affecting the performance. So, pre-processing the data before model training and tuning the model while training both are important.

RQ5. How well does Fairway perform compared to previous fairness algorithms?

We have decided to compare Fairway with one of the most popular prior works in fairness domain - Optimized Pre-processing by Calmon et al. [Cal17b]. Table 4.2 shows the results for ten datasets. It shows the change of five performance metrics - recall, false alarm, precision, accuracy, F1 and four fairness metrics AOD, EOD, SPD, DI before and after the algorithms are applied. In most of the cases, Fairway is performing better or the same with Optimized Pre-processing in case of reducing ethical bias. In case of performance metrics also, Fairway is similar or better than Optimized Pre-processing. Like Fairway, Optimized Pre-processing also slightly damages the performance. In some situations, this may become a matter of concern. We see a scope of improvement here where future researchers should focus.

Fairway is not just another bias mitigation approach. It differs from prior works in several ways -

- The first part of Fairway is finding bias in training data. So, even before model training, Fairway shows which data points in the training data have improper/biased labels and can affect prediction in future. If labeling was done by human reviewers, it leads to finding bias in human decisions. Instead of blindly trusting the ground truth of training data, Fairway can be used to find bias in the ground truth.
- Prior bias mitigation algorithms come from the core concepts of machine learning. Software practitioners having little ML knowledge may face difficulties to use these algorithms[Hol19].

In case of Fairway, users can clearly see how two different models trained on privileged and unprivileged groups give different predictions on biased data points. This makes Fairway much more comprehensible. FAIR_FLASH gives user the flexibility to choose which parameters to optimize. In this work, Logistic regression model is used. But FAIR_FLASH is easily extensible for other classification models. So, FAIR_FLASH is adjustable too.

- Fairway is a combination of bias testing and mitigation. This is described in **RQ6**.

RQ6. Can Fairway be used as a combined tool for detection and mitigation of bias?

In section 2.1, it is shown that there are mainly two types of previous works done by researchers - finding the bias in AI software and mitigating the bias. As per our knowledge, we are the first one to combine these two. Fairway finds the data points which have unfair labeling in the training data and remove those data points so that prediction is not affected by protected attribute. We used *Situation testing* [Zha16c] to verify whether after bias removal, the role of a protected attribute on the prediction changes or not. we switched the protected attribute value for all the remaining data points (e.g. we changed Male to Female and Female to Male). Then we checked whether these changes lead to prediction changes or not. If the prediction changes for a data point, we say that it fails situation testing. Figure 4.4 shows the percentage of data points failing situation testing before and after pre-processing step of Fairway:

- The “pink” and “gray” columns show results before/after applying Fairway.
- In all cases, the values on the gray columns are far smaller than pink columns.

So, Fairway can find the data points responsible for bias in the training data. Now, it is an engineering decision to set the threshold of what percentage of training data can be ambiguous where prediction may change depending on the protected attribute value. Fairway provides the percentage and depending on the application, user can decide whether bias is present in the system or not. So, Fairway can be applied as a discrimination finder tool. If discrimination is above the tolerable threshold, then Fairway can be applied for removing bias from training data and optimizing model without damaging predictive performance. So, Fairway can be used as a combined tool for detection and mitigation of discrimination or ethical bias. One unique feature of Fairway is it is **model-agnostic**. It finds bias by verifying prediction of a model and mitigates bias by cleaning training data and tuning model parameters. So, it can work for any black box model. As Fairway only works on the output space of a model, it can be easily used in industrial purposes where revealing core algorithm of the underlying model is not possible.

So, to summarize the results, we say that we have explained the reasons of bias in the five datasets we used. We have developed a comprehensible method *Fairway* which can remove bias from training data and the model. Unlike prior works, *Fairway* is not just a bias mitigation approach, it is a combined tool for ground truth validation, bias detection and mitigation.

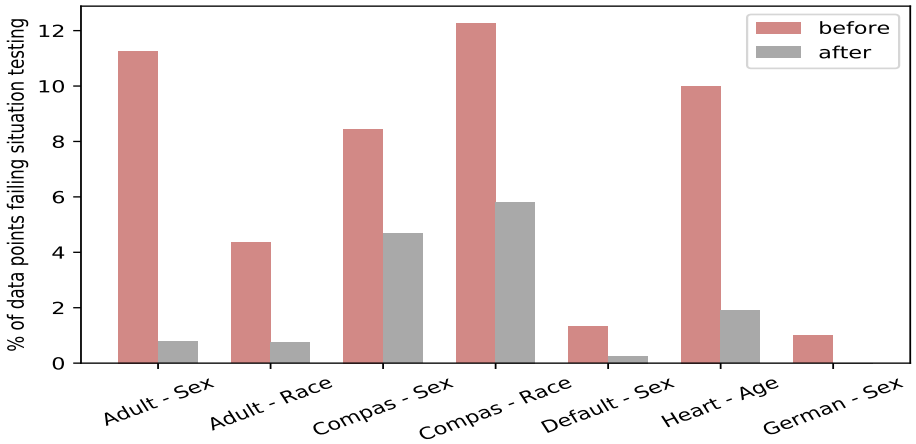

Figure 4.4 Percentage change of data points failing situation testing (showing bias) before and after pre-processing.

4.4 Threats to Validity

- **Sampling Bias** - We have used ten datasets from UCI machine learning repository where most of prior works in fairness domain use only one or two datasets. These are well-known datasets and used by previous researchers in ML and software fairness domain. It is an open issue if these data sets reflect an interesting range of fairness issues for other data sets. In future work, we would explore more data sets.
- **Evaluation Bias** - We have used four fairness metrics - AOD, EOD, SPD and DI. IBM AIF360 contains more than 50 fairness metrics. In future work, we would explore more performance criteria.
- **Construct Validity** - In our work we trained different models on privileged and unprivileged groups. The datasets contained one or two protected attributes, so our method is feasible. All the prior works we have seen treated each protected attribute individually. In future work, we would explore larger data sets with more protected attributes.
- **External Validity** - Fairway is limited to classification models which are very common in AI software. We are currently working on extending it to Regression models. In future work, we would extend this work to other kinds of data mining problems; e.g. to text mining or video processing systems.

4.5 Conclusion

We have explained how a model acquires bias from improper labels of training data and have demonstrated an approach called “Fairway” which removes “ethical bias” from the training data

and optimizes a trained model for fairness and performance. We have shown that Fairway is comprehensible and can be used as a combined tool for detection and mitigation of bias. Unlike some prior ML works, Fairway is not just a bias mitigation tool, it validates ground truth labels, finds bias and mitigates bias. We have made the source code of "Fairway" publicly available for software researchers and practitioners. To the best of our knowledge, we claim this is the first work in SE domain which concentrates on mitigating ethical bias from software and making software fair using optimization methods augmented with some data pre-processing. In future, we hope more and more software researchers will work on this domain and industries will consider publishing more datasets. When that data becomes available, it would be appropriate to rerun this study.

CHAPTER

5

FAIR-SMOTE STUDY (DATA BALANCING)

This chapter describes our second bias mitigation framework - Fair-SMOTE. It is a pre-processing approach that balances training data based on class and protected attributes. This study was published on ESEC/FSE 2021 [Cha21]. We received the ***ACM SIGSOFT Distinguished Paper Award*** *for this work.*

5.1 Introduction

Prior works on managing fairness in machine learning software have tried mitigating bias by exploring a very wide space of control parameters for machine learners. For example, Johnson et al. [Joh20] executed some (very long) grid search that looped over the set of control parameters of a learner to find settings that reduced bias. Our previous work, Fairway [Cha20b], published in FSE'20 used stochastic sampling to explore the space of control options (using some incremental feedback operator to adjust where to search next). While these approaches were certainly useful in the domains tested by Johnson and Chakraborty et al., these methods are "dumb" in a way because they do not take advantage of domain knowledge. This is a problem since, as shown by this paper, such domain knowledge can lead to a more direct (and faster and more effective) bias mitigation strategy.

The insight we offer here is that the root causes of bias might be the *prior decisions that generated the training data*. Those prior decisions affect (a) what data was collected and (b) the labels assigned to those examples. Hence, to fix bias, it might suffice to apply mutators to the training data in order to

(A) remove biased labels; and (B) rebalance internal distributions such that they are

equal based on class and sensitive attributes.

This "Fair-SMOTE" tactic uses *situation testing* [Usa; Eus] to find biased labels. Also, it balances frequencies of sensitive attributes and class labels (whereas the older SMOTE algorithm [Cha02] just balances the class labels). We recommend Fair-SMOTE since:

- Fair-SMOTE is *faster* (220%) than Fairway.
- Models generated via Fair-SMOTE are more *effective* (measured in terms of recall and F1) than Fairway and another state-of-the-art bias mitigation algorithm [Cal17b].

Overall, this work makes the following contributions:

- We demonstrate two main reasons for the training data being biased - a) data imbalance, b) improper data label.
- We combine finding bias, explaining bias, and removing bias.
- We show that traditional class balancing techniques damage the fairness of the model.
- Prior works compromised performance while achieving fairness. We achieve fairness with better recall and F1 score.
- To the best of our knowledge, this study explores more learners and datasets than prior works that were based on just a handful of datasets and/or learners (e.g. [Cal17b; Zha18; Kam18]).

Before beginning, we digress to clarify two points. Firstly, in this paper, *training data* is mutated by Fair-SMOTE but the *test data* remains in its original form.

Secondly, one danger with mutating data is that important associations between variables can be lost. Hence, in this work, we take care to mutate by extrapolating between the values seen in two neighboring examples. In that mutation process, Fair-SMOTE *extrapolates all the variables by the same amount*. That is, if there exists some average case association between the pre-mutated examples, then that association is preserved in the mutant. To facilitate open science, the source code and datasets used in this study are all available online[1].

5.2 Methodology

5.2.1 Fair-SMOTE

Fair-SMOTE algorithm solves data imbalance. At first, the training data is divided into subgroups based on class and protected attribute. If class and protected attribute both are binary, then there will be 2*2 = 4 subgroups (Favorable & Privileged, Favorable & Unprivileged, Unfavorable & Privileged, Unfavorable & Unprivileged). Initially, these subgroups are of unequal sizes.

[1] https://github.com/joymallyac/Fair-SMOTE

Fair-SMOTE synthetically generates new data points for all the subgroups except the subgroup having the maximum number of data points. As a result, all subgroups become of equal size (same as the maximum one).

As stated in the introduction, one danger with mutating data is that important associations between variables can be lost. Hence, in this work, we take care to mutate by extrapolating between the values seen in two neighboring examples. In that mutation process, Fair-SMOTE *extrapolates all the variables by the same amount*. That is, if there exists some average case association between the pre-mutated examples, then that association is preserved in the mutant.

For data generation, we use two hyperparameters "mutation amount" (f) and "crossover frequency" (cr) like *Differential Evolution* [Sto97a]. They both lie in the range [0, 1]. The first one denotes at which probability the new data point will be different from the parent point (0.8 means 80% of the times) and the latter one denotes how much different the new data point will be. We have tried with 0.2 (< 50% probability), 0.5 (= 50%), & 0.8 (> 50%) and got best results with 0.8.

Algorithm 2 describes the pseudocode of Fair-SMOTE. It starts with randomly selecting a data point (parent point p) from a subgroup. Then using K-nearest neighbor, two data points (c1, c2) are selected which are closest to p. Next, according to the algorithm described, a new data point is created. Separate logic is used for boolean, symbolic, and numeric columns.

Algorithm 2: Pseudocode of Fair-SMOTE

```
Input: Dataset, Protected Attribute(p_attrs), Class Label(cl)
Output: Balanced Dataset
Def get_ngbr(Dataset, p_attrs, cl):
    count_groups = get_count(Dataset, p_attrs, cl)
    max_size = max(count_groups)
    cr, f = 0.8, 0.8 (user can pick any value in [0,1])
    for c in cl do
        for attr in p_attrs do
            sub_data = Dataset(cl=c & p_attrs=attr)
            sub_group_size = count_groups[c][attr]
            to_generate = max_size - sub_group_size
            knn = NearestNeighbors(sub_data)
            for i in range(to_generate) do
                parent = Dataset[rand_sample_id]
                ngbr = knn.kneighbors(parent, 2)
                c1, c2 = Dataset[ngbr[0]], Dataset[ngbr[1]]
                new_candidate = []
                for col in parent.columns do
                    if cr > random(0,1) then
                        new_val = p[col] + f*(c1[col]-c2[col])
                    else
                        new_val = p[col]
                    new_candidate.add(new_val)
            Dataset.add(new_candidate)
    return Dataset
```

Fair-SMOTE does not randomly create a new data point. Rather it creates a data point that is very close to the parent point. Thus the generated data points belong to the same data distribution. This process is repeated until all the subgroups become of similar size.

After applying Fair-SMOTE, the training data contains equal proportion of both classes and the protected attribute.

5.2.2 Fair Situation Testing

At first, we use Fair-SMOTE to balance the training data. Fair-SMOTE is an oversampling technique and thus increases the size of train set. We then use *situation testing* as mentioned in §3.3 to find out biased data points in the training data. We call this *situation testing* as *fair situation testing* because this is making the training data fairer. After finding biased data points, we remove them from training data. As Table 3.2 shows small percentage numbers, we do not lose much of the training data. We will show in the 'Results' section that this does not affect performance of the model much. After removal of biased data points, we train the model on the remaining training set and finally make the prediction on test data.

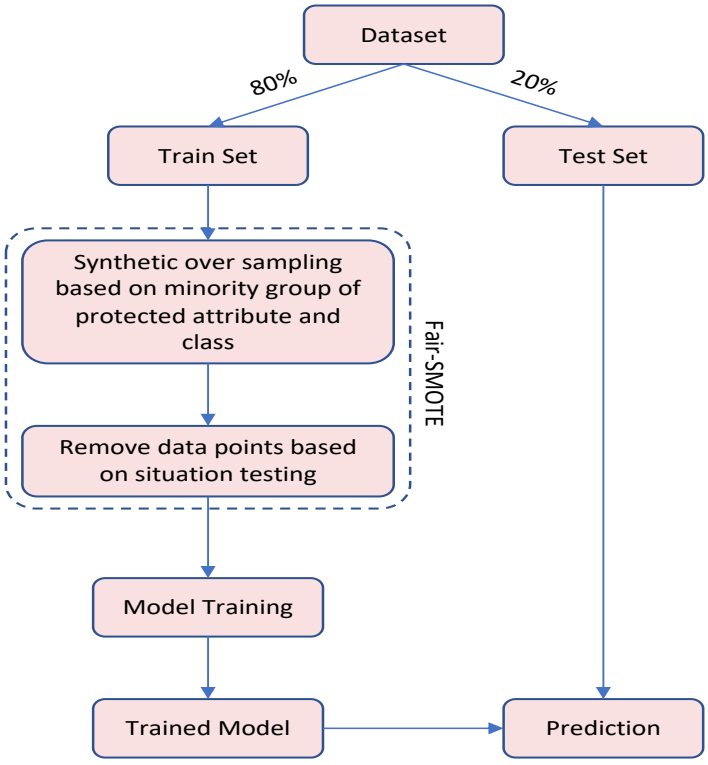

Figure 5.1 Block diagram of Fair-SMOTE

5.2.3 Experimental Design

Here we describe how we prepared the data for experiments to answer the research questions in §9.3. Our study used 10 datasets (Table 2.1) and 3 classification models - logistic regression (LSR), random forest (RF), and support vector machine (SVM). In fairness domain, datasets are not very large in size and also have small dimensions. That is why we see most of the prior works [Cha20b; Kam12c; Cal17b; Bis20] choose simple models like us instead of deep learning models. For every experiment, we split the datasets using 5 fold cross-validation (train - 80%, test - 20%) and repeat 10 times with random seeds and finally report the median. The rows containing missing values are ignored, continuous features are converted to categorical (e.g., age<25: young, age>=25: old), non-numerical features are converted to numerical (e.g., male: 1, female: 0), finally, all the feature values are normalized (converted between 0 to 1). These basic conversions are done for all the experiments. Classification model is first trained on training data and then tested on test data; we report the median of ten runs.
Figure 5.1 shows the block-diagram of one repeat of our framework.

5.3 Results

The results are structured around five research questions.

RQ1. Can we find bias by just looking at the training data?

Previously, we have said a machine learning model acquires bias from training data. RQ1 asks for the signals we must see to identify whether training data has bias or not. That is an important question to ask because if that is doable and that bias can be removed before model training, then the chances of bias affecting the final decision reduce significantly. Table 7.1 shows results for three different models (logistic regression (LSR), random forest (RF), support vector machine (SVM)), and Table 5.2 shows results for one model (LSR) only. The row "Default" is when we train the model on raw data. Results show that "Default" row has significantly high bias scores for all the datasets that means trained model is showing discrimination. Previously we have dived deep into these datasets to find reasons for bias. Here we are summarizing the sanity checks every dataset should go through before model training to avoid discrimination.

- **Data distribution** - The training data should be almost balanced based on class and protected attribute.
- **Data label** - Every data point should go through *situation testing* to see whether label has bias or not.

If we find bias in the training data, we should apply Fair-SMOTE to remove that and get fair outcomes. Thus, the answer for RQ1 is "**Yes, we can find bias by just looking at the training data**"

Table 5.1 Results for RQ1, RQ2 & RQ3. In this table "Default" means off-the-shelf learner; SMOTE is an algorithm by Chawla et al. [Cha02] from 2002; and Fair-SMOTE is the algorithm introduced by this paper. Cells show medians for 10 runs. Here, the darkest pink cells show top rank (note: for the metrics with '+' more is better and for the metrics with '-' less is better). The lighter pink and lightest pink cells show rank two and rank three respectively; the white cells show the worst rank. Rankings were calculated via Scott-Knott test (§4.3.1).

Dataset	Protected Attribute	Algorithms	Recall (+)	False alarm (-)	Precision (+)	Accuracy (+)	F1 Score (+)	AOD (-)	EOD (-)	SPD (-)	DI (-)
Adult Census Income	Sex	Default - LSR	0.42	0.07	0.69	0.83	0.54	0.12	0.24	0.21	0.56
		Default - RF	0.51	0.06	0.72	0.83	0.59	0.09	0.17	0.17	0.33
		Default - SVM	0.35	0.04	0.81	0.82	0.49	0.1	0.24	0.14	0.43
		SMOTE - LSR	0.70	0.25	0.49	0.70	0.64	0.17	0.37	0.33	0.58
		SMOTE - RF	0.61	0.07	0.71	0.83	0.62	0.08	0.16	0.16	0.32
		SMOTE - SVM	0.71	0.19	0.55	0.78	0.62	0.11	0.52	0.42	0.46
		Fair-SMOTE - LSR	0.71	0.25	0.51	0.73	0.62	0.01	0.02	0.03	0.15
		Fair-SMOTE - RF	0.69	0.2	0.53	0.78	0.6	0.03	0.04	0.1	0.22
		Fair-SMOTE - SVM	0.73	0.23	0.51	0.76	0.6	0.02	0.02	0.08	0.21
Adult Census Income	Race	Default - LSR	0.42	0.05	0.69	0.81	0.52	0.06	0.15	0.16	0.52
		Default - RF	0.53	0.07	0.7	0.83	0.6	0.12	0.16	0.12	0.57
		Default - SVM	0.35	0.03	0.8	0.82	0.49	0.08	0.11	0.10	0.35
		SMOTE - LSR	0.71	0.23	0.49	0.72	0.61	0.16	0.19	0.23	0.56
		SMOTE - RF	0.66	0.08	0.67	0.83	0.63	0.09	0.12	0.12	0.52
		SMOTE - SVM	0.58	0.11	0.62	0.81	0.6	0.09	0.12	0.17	0.32
		Fair-SMOTE - LSR	0.7	0.22	0.51	0.72	0.62	0.04	0.03	0.05	0.26
		Fair-SMOTE - RF	0.73	0.2	0.52	0.8	0.61	0.01	0.02	0.08	0.29
		Fair-SMOTE - SVM	0.71	0.26	0.5	0.75	0.61	0.01	0.01	0.06	0.18
Compas	Sex	Default - LSR	0.73	0.38	0.66	0.64	0.61	0.05	0.14	0.18	0.24
		Default - RF	0.75	0.45	0.66	0.67	0.72	0.11	0.18	0.2	0.34
		Default - SVM	0.77	0.45	0.66	0.67	0.71	0.1	0.15	0.22	0.28
		SMOTE - LSR	0.65	0.33	0.62	0.6	0.65	0.08	0.19	0.22	0.31
		SMOTE - RF	0.72	0.42	0.67	0.65	0.7	0.11	0.22	0.26	0.3
		SMOTE - SVM	0.7	0.36	0.68	0.66	0.69	0.1	0.21	0.31	0.38
		Fair-SMOTE - LSR	0.62	0.32	0.56	0.55	0.65	0.02	0.05	0.08	0.04
		Fair-SMOTE - RF	0.71	0.44	0.66	0.65	0.7	0.04	0.03	0.1	0.02
		Fair-SMOTE - SVM	0.79	0.5	0.65	0.66	0.71	0.02	0.01	0.06	0.08
Compas	Race	Default - LSR	0.69	0.39	0.65	0.64	0.68	0.05	0.11	0.12	0.21
		Default - RF	0.75	0.44	0.66	0.66	0.7	0.07	0.17	0.21	0.24
		Default - SVM	0.77	0.45	0.66	0.67	0.71	0.07	0.14	0.18	0.24
		SMOTE - LSR	0.61	0.32	0.61	0.6	0.63	0.06	0.16	0.14	0.27
		SMOTE - RF	0.75	0.42	0.67	0.66	0.7	0.07	0.13	0.19	0.31
		SMOTE - SVM	0.7	0.39	0.68	0.66	0.69	0.09	0.12	0.16	0.24
		Fair-SMOTE - LSR	0.62	0.30	0.56	0.55	0.66	0.01	0.05	0.06	0.11
		Fair-SMOTE - RF	0.66	0.39	0.67	0.65	0.67	0.01	0.03	0.02	0.10
		Fair-SMOTE - SVM	0.7	0.41	0.67	0.65	0.68	0.02	0.06	0.08	0.12
MEPS - 16	Race	Default - LSR	0.35	0.05	0.65	0.85	0.44	0.04	0.1	0.07	0.43
		Default - RF	0.38	0.11	0.62	0.81	0.41	0.08	0.12	0.08	0.36
		Default - SVM	0.31	0.08	0.66	0.79	0.4	0.09	0.1	0.09	0.32
		SMOTE - LSR	0.65	0.22	0.58	0.78	0.49	0.17	0.11	0.11	0.49
		SMOTE - RF	0.63	0.18	0.56	0.78	0.52	0.12	0.19	0.13	0.31
		SMOTE - SVM	0.62	0.23	0.49	0.79	0.5	0.13	0.18	0.15	0.47
		Fair-SMOTE - LSR	0.66	0.2	0.41	0.77	0.51	0.01	0.03	0.04	0.17
		Fair-SMOTE - RF	0.62	0.21	0.38	0.77	0.48	0.03	0.04	0.03	0.19
		Fair-SMOTE - SVM	0.61	0.18	0.39	0.76	0.49	0.04	0.03	0.05	0.19

RQ2. Are standard class balancing techniques helpful to reduce bias?

For answering RQ2, we chose the most used class balancing technique, which is SMOTE [Cha02]. Table 7.1 contains results for three datasets and three different learners (LSR, RF, SVM). In that table, for a particular dataset and for a particular performance metric:

- Cells with darker pink backgrounds denote treatments that are performing *better* than anything else.
- Conversely, cells with a white background denote treatments that are performing *worse* than anything else;.

Table 5.2 Results for RQ3, RQ4 (learner= Logistic Regression). In this table. "Default" denotes off-the-shelf logistic regression; OP is Calmon et al.'s system from NIPS'17 [Cal17b]; Fairway is Chakraborty et al.'s system from FSE'20 [Cha20b]; and Fair-SMOTE is the algorithm introduced by this paper. Cells show medians for 10 runs. Here, the darker pink cells show top rank (note: for the metrics with '+' more is better and for the metrics with '-' less is better). The lighter pink cells show rank two; white shows lowest rank (worst performance). Rankings were calculated via the Scott-Knott test (§4.3.1)

Dataset	Protected Attribute	Algorithms	Recall (+)	False alarm (-)	Precision (+)	Accuracy (+)	F1 Score (+)	AOD (-)	EOD (-)	SPD (-)	DI (-)
Adult Census Income	Sex	Default	0.42	0.07	0.69	0.83	0.54	0.12	0.24	0.21	0.56
		OP	0.41	0.09	0.61	0.76	0.51	0.04	0.03	0.04	0.14
		Fairway	0.25	0.04	0.70	0.72	0.42	0.02	0.03	0.01	0.11
		Fair SMOTE	0.71	0.25	0.51	0.73	0.62	0.01	0.02	0.03	0.12
Adult Census Income	Race	Default	0.42	0.05	0.69	0.81	0.52	0.06	0.15	0.16	0.52
		OP	0.38	0.06	0.66	0.78	0.48	0.03	0.02	0.05	0.21
		Fairway	0.36	0.04	0.70	0.73	0.44	0.02	0.03	0.06	0.32
		Fair SMOTE	0.7	0.22	0.51	0.72	0.62	0.04	0.03	0.05	0.26
Compas	Sex	Default	0.73	0.38	0.66	0.64	0.61	0.05	0.14	0.18	0.28
		OP	0.71	0.36	0.64	0.62	0.60	0.04	0.05	0.06	0.09
		Fairway	0.56	0.22	0.57	0.58	0.58	0.03	0.03	0.06	0.08
		Fair SMOTE	0.62	0.32	0.56	0.55	0.65	0.02	0.05	0.08	0.09
Compas	Race	Default	0.69	0.39	0.65	0.64	0.68	0.05	0.11	0.12	0.21
		OP	0.68	0.33	0.63	0.62	0.67	0.03	0.06	0.06	0.12
		Fairway	0.55	0.21	0.58	0.58	0.56	0.02	0.04	0.07	0.07
		Fair SMOTE	0.62	0.30	0.56	0.55	0.66	0.01	0.05	0.06	0.11
German Credit	Sex	Default	0.94	0.81	0.72	0.76	0.82	0.11	0.08	0.14	0.15
		OP	0.75	0.73	0.71	0.73	0.71	0.04	0.05	0.06	0.12
		Fairway	0.78	0.76	0.68	0.65	0.73	0.05	0.04	0.07	0.11
		Fair SMOTE	0.62	0.36	0.71	0.64	0.71	0.05	0.05	0.05	0.13
Default Credit	Sex	Default	0.25	0.07	0.7	0.78	0.34	0.05	0.08	0.06	0.45
		OP	0.28	0.06	0.65	0.70	0.32	0.01	0.02	0.03	0.19
		Fairway	0.21	0.04	0.67	0.67	0.33	0.01	0.04	0.03	0.22
		Fair SMOTE	0.58	0.26	0.39	0.68	0.44	0.02	0.03	0.05	0.14
Heart Health	Age	Default	0.72	0.2	0.74	0.72	0.74	0.11	0.13	0.32	0.44
		OP	0.69	0.19	0.72	0.69	0.69	0.04	0.06	0.09	0.14
		Fairway	0.65	0.21	0.69	0.67	0.68	0.05	0.05	0.04	0.18
		Fair SMOTE	0.66	0.20	0.69	0.68	0.66	0.08	0.07	0.08	0.20
Bank Marketing	Age	Default	0.73	0.21	0.76	0.77	0.77	0.07	0.18	0.31	0.68
		OP	0.72	0.20	0.74	0.75	0.75	0.05	0.05	0.12	0.24
		Fairway	0.71	0.17	0.73	0.71	0.73	0.04	0.03	0.13	0.25
		Fair SMOTE	0.76	0.18	0.72	0.72	0.74	0.05	0.07	0.15	0.23
Home Credit	Sex	Default	0.31	0.18	0.28	0.86	0.29	0.07	0.06	0.08	0.59
		OP	0.3	0.17	0.30	0.83	0.31	0.02	0.04	0.04	0.16
		Fairway	0.28	0.12	0.25	0.65	0.26	0.02	0.04	0.03	013
		Fair SMOTE	0.33	0.18	0.31	0.75	0.32	0.03	0.02	0.03	0.15
Student Performance	Sex	Default	0.81	0.06	0.85	0.88	0.83	0.06	0.05	0.06	0.12
		OP	0.79	0.06	0.83	0.83	0.82	0.03	0.02	0.03	0.06
		Fairway	0.76	0.05	0.81	0.84	0.84	0.03	0.02	0.04	0.07
		Fair SMOTE	0.91	0.10	0.84	0.87	0.86	0.04	0.04	0.04	0.08
MEPS - 15	Race	Default	0.36	0.03	0.68	0.85	0.45	0.04	0.05	0.08	0.36
		OP	0.35	0.04	0.66	0.83	0.44	0.04	0.02	0.06	0.15
		Fairway	0.35	0.04	0.42	0.77	0.41	0.03	0.02	0.04	0.12
		Fair SMOTE	0.68	0.22	0.41	0.77	0.53	0.02	0.02	0.05	0.15
MEPS - 16	Sex	Default	0.35	0.05	0.65	0.85	0.44	0.04	0.1	0.07	0.43
		OP	0.34	0.04	0.65	0.83	0.44	0.05	0.02	0.05	0.12
		Fairway	0.32	0.04	0.55	0.76	0.42	0.03	0.02	0.04	0.16
		Fair SMOTE	0.66	0.2	0.41	0.77	0.51	0.01	0.03	0.04	0.17

SMOTE consistently increases bias scores (AOD, EOD, SPD, DI) mean damaging fairness but performs similar/better than "Default" in case of performance metrics (as measured by recall, false alarm, precision, accuracy & F1). Thus, the answer for RQ2 is "**No, standard class balancing techniques are not helpful since, in their enthusiasm to optimize model performance, they seem to also amplify model bias.**"

RQ3. Can Fair-SMOTE reduce bias?

Table 7.1 answers this question. Looking at the bias metrics (AOD, EOD, SPD, DI), Fair-SMOTE significantly reduces all the bias scores mean increasing fairness (see the darker colored cells).

As to the performance measures (recall, false alarm, precision, accuracy, and F1) it is hard to get a visual summary of the results just by looking at Table 7.1. For that purpose, we turn to rows 1,2,3,4 of Table 5.3. Based on Scott-Knott tests of §4.3.1, these rows count the number of times Fair-SMOTE wins, losses or ties compared to SMOTE (here, "tie" means "is assigned the same rank by Scott-Knott"). Those counts, in rows 1,2,3,4 confirm the visual patterns of Table 7.1:

- As to the performance measures (recall, false alarm, precision, accuracy, and F1), these methods often tie.
- But looking at the pink highlighted bias metrics results for AOD, EOD, SPD, & DI, Fair-SMOTE is clearly performing better than SMOTE.

Thus, the answer for RQ3 is "**Yes, Fair-SMOTE reduces bias significantly and performs much better than SMOTE.**"

RQ4. How well does Fair-SMOTE perform compared to the state of the art bias mitigation algorithms?

Table 5.3 compares Fair-SMOTE against other tools that try to find and fix bias. "Default" shows the off-the-shelf learners; "Fairway" is the Chakraborty et al. [Cha20b] system from FSE'20; and OP is the Optimized Pre-processing method from NIPS'17 [Cal17b]. Here, the learner is logistic regression since, for performance measures, LSR has best results in the Fair-SMOTE results of Table 7.1.

Rows 5,6,7,8,9,10,11,12 of Table 5.3 summarize these results. Measured in terms of bias reduction, all the methods often tie. But observing the pink highlighted cells in that table, we see that the Fair-SMOTE performs much better (in terms of recall and F1) than anything else.

Thus, the answer for RQ4 is "**Fair-SMOTE performs similar or better than two state of the art bias mitigation algorithms in case of fairness and consistently gives higher recall and F1 score.**" That means we do not have to compromise performance anymore to achieve fairness. Fair-SMOTE is able to provide both - better fairness and performance. This is the biggest achievement of this work.

RQ5. Can Fair-SMOTE reduce bias for more than one protected attribute?

In the literature, we are unaware of any work trying to reduce bias for more than one protected attribute at a time. Typically what researchers do is try to eliminate bias based on one protected attribute, one at a time. We experimented on Adult dataset having two protected attributes (sex, race). The idea is to balance the training data with respect to class and two protected attributes. That means we need to find out among the eight (2^3) possible subgroups which one has the most number of data points. Then we need to make other subgroups of the same size with the one having the most number of data points. That is done by generating new data points using Fair-SMOTE. Table 5.4 shows results that bias is reduced for both the attributes along with higher recall and F1 score. Hence, the answer of RQ5 is "**Yes, Fair-SMOTE can simultaneously reduce bias for more than one protected attribute.**"

5.4 Discussion: Why Fair-SMOTE?

Here we discuss what makes Fair-SMOTE unique and more useful than prior works in the fairness domain.

Combination: This is only the second SE work (after Fairway [Cha20b]) in fairness domain where primary focus is bias mitigation. There are a lot of papers in ML domain where various techniques are provided to remove bias from model behavior. Still, when it comes to applicability, we see software practitioners find ML fairness as a complicated topic. Because finding bias, explaining bias, and

Table 5.3 RQ3, RQ4 results. Summarized information of comparing Fair-SMOTE with SMOTE [Cha02], Fairway [Cha20b] & Optimized Pre-processing [Cal17b] based on results of 10 datasets and 3 learners (LSR, RF, SVM). Number of wins, ties, and losses are calculated based on Scott-Knott ranks for each metric. Highlighted cells show Fair-SMOTE significantly outperforming others.

		Recall	False alarm	Precision	Accuracy	F1 Score	AOD	EOD	SPD	DI	Total
		SMOTE vs Fair-SMOTE									
1	**Win**	4	4	1	6	3	33	33	34	32	150
2	**Tie**	25	27	29	28	30	2	3	2	2	148
3	**Loss**	7	5	6	2	3	1	0	0	2	26
4	**Win + Tie**	29	31	30	34	33	35	36	36	34	298/324
		Fairway vs Fair-SMOTE									
5	**Win**	14	4	6	5	20	3	2	3	4	61
6	**Tie**	20	20	27	28	14	30	31	32	31	233
7	**Loss**	2	12	3	3	2	3	3	1	1	28
8	**Win + Tie**	34	24	33	33	34	33	33	35	35	294/324
		Optimized Pre-processing vs Fair-SMOTE									
9	**Win**	10	7	4	3	12	1	2	2	3	44
10	**Tie**	21	22	26	30	20	34	33	32	31	249
11	**Loss**	5	7	6	3	4	1	1	2	2	31
12	**Win + Tie**	31	29	30	33	32	35	35	34	34	293/324

Table 5.4 RQ5 results. Fair-SMOTE reducing bias for 'sex' and 'race' simultaneously (Adult dataset). Best cells are highlighted.

	Protected attribute	Recall(+)	False alarm(-)	Precision(+)	Accuracy(+)	F1 Score(+)	AOD(-)	EOD(-)	SPD(-)	DI(-)
Default	Sex	0.42	0.07	0.69	0.83	0.52	0.12	0.24	0.21	0.56
	Race						0.06	0.15	0.16	0.52
Fair-SMOTE	Sex	0.71	0.24	0.49	0.75	0.59	0.02	0.05	0.09	0.27
	Race						0.01	0.03	0.08	0.22

removing bias have been treated as separate problems and thus created more confusion. That's where this work makes a significant contribution by combining all of them together. We first find data imbalance and improperly labeled data points (by situation testing) and then use oversampling to balance the data and remove improperly labeled data points. As an outcome, we generate fair results.

Uncompromising: Our framework improves fairness scores along with F1 score and recall. It does not damage accuracy and precision much also. That said, unlike much prior work [Cal17b; Kam12a; Kam12c; Zha18; Kam18; Har16], we can do bias mitigation *without* compromising predictive performance. We attribute our success in this regard to our sampling policy. None of our mutators damage the associations between attributes. Rather, we just carefully resample the data to avoid certain hard cases (where the training data can only see a few examples of each kind of row).

Group & Individual: Our data balancing approach takes care of *group fairness* where goal is based on the protected attribute, privileged and unprivileged groups will be treated similarly. Our situation testing method takes care of *individual fairness* where goal is to provide similar outcomes to similar individuals.

Generality: We entirely focused on data to find and remove bias. There are works where optimization tricks have been used while model training to remove bias [Zha18; Kam12c]. These works are model specific and most of the time combine with internal model logic. However, Fair-SMOTE does not require access to inside model logic. Thus it is much more general as it can be used for any kind of model. In this work, we used three simple models and got promising results. In future, we will explore deep learning models.

Versatility: We used Fair-SMOTE for only classification datasets here. However, the core idea of Fair-SMOTE is keeping equal proportion of all the protected groups in the training data. We believe the same approach can be applied to regression problems. In future we would like to explore that. Besides, the same approach can be applied for image data (face recognition) to train model with equal proportion of white and black faces so that model does not show racial bias. That means Fair-SMOTE can be easily adopted by other domains to solve bias issues caused by data imbalance.

5.5 Threats to validity

Sampling Bias - As per our knowledge, this is the most extensive fairness study using 10 real-world datasets and 3 classification models. Still, conclusions may change a bit if other datasets and models are used.

Evaluation Bias - We used the four most popular fairness metrics in this study. Prior works [Cha20b; Har16; Kam12c] only used two or three metrics although IBM AIF360 contains more than 50 metrics. In future, we will explore more evaluation criteria.

Internal Validity - Where prior researchers [Zha16b; Lof18] focused on attributes to find causes of bias, we concentrated on data distribution and labels. However, there could be some other reasons. A recent Amazon paper comments on some other reasons such as objective function bias,

homogenization bias [Das20]. We could not validate these biases in our datasets as it was out of scope. In future, we would like to explore those reasons if industry datasets become available.
External Validity - This work is based on binary classification and tabular data which are very common in AI software. We are currently working on extending it to regression models. In future work, we would extend this work to other domains such as text mining and image processing.

5.6 conclusion

This paper has tested the Fair-SMOTE tactic for mitigating bias in ML software. Fair-SMOTE assumes the root causes of bias are the prior decisions that control (a) what data was collected and (b) the labels assigned to the data. Fair-SMOTE:

> *(A) Removes biased labels; and (B) rebalances internal distributions such that they are equal based on class and sensitive attributes.*

As seen above, Fair-SMOTE was just as effective at bias mitigation as two other state-of-the-art algorithms [Cha20b; Cal17b] and more effective in terms of achieving higher performance (measured in terms of recall and F1). Also, Fair-SMOTE runs 220% faster (median value across ten data sets) than Chakraborty et.al [Cha20b].
Based on the above, we offer three conclusions:

1. We can recommend Fair-SMOTE for bias mitigation.
2. We can reject the pessimism of Berk et al. [Ber17b] who, previously, had been worried that the cost of fairness was a reduction in learner performance.
3. More generally, rather than blindly applying some optimization methods it can be better to:
    - Reflect on the domain;
    - Use insights from that reflection to guide improvements in that domain.

CHAPTER

6

FAIRBALANCE STUDY (MULTIPLE SENSITIVE ATTRIBUTES WITH DATA BALANCING)

This work was done in collaboration with Dr. Zhe Yu. This work aims to improve machine learning fairness on multiple sensitive attributes. Most existing solutions for machine learning fairness either target only one sensitive attribute (e.g. sex) at a time, or have magic parameters to tune, or have expensive computational overhead. To overcome these challenges, we propose FairBalance to balance the group distribution of training data across every sensitive attribute before training the machine learning models. This study was submitted to the ICML 2022.

6.1 Introduction

With machine learning and artificial intelligence software increasingly being used to make decisions that affect people's lives, much concern has been raised on the fairness in machine learning. Studies have shown that, sometimes the machine learning software behaves in a biased manner that gives undue advantages to a specific group of people (where those groups are determined by sex, race, etc.). Such bias in the machine learning software can have serious consequences in deciding whether a patient gets released from the hospital [Med; Str16], which loan applications are approved [Fora], which citizens get bail or sentenced to jail [Pro], who gets admitted/hired by universities/companies [Amad].

Many research studies have been done trying to mitigate the ethical bias in the machine learning software. However, most existing solutions for machine learning fairness target only one sensitive attribute (e.g. sex) at a time. For example, on a dataset with two sensitive attributes sex and race, most existing approaches can learn a fair model on sex or a fair model on race, but cannot learn a model which is unbiased on both sex and race [Kam12b; Cal17c; Zha18]. This hinders the application of bias mitigation algorithms since a fair machine learning model cannot be biased on any sensitive attribute.

Some in-processing bias mitigation algorithms can tackle multiple sensitive attributes at the same time by optimizing for both prediction performance and fairness metrics [Low21]. However, such in-processing bias mitigation algorithms are usually very expensive. Magic parameters also need to be decided beforehand to trade off between prediction performance and fairness metrics. Another recent work from Chakraborty et al. [Cha21] also works with multiple sensitive attributes. This work utilizes an oversampling technique along with a data selection technique to preprocess the data for learning a fairer model. Unfortunately, this approach has expensive computational overhead as well.

In this paper, we propose a simple yet effective algorithm, FairBalance, to learn a fairer machine learning model on every sensitive attribute. FairBalance is a pre-processing technique which balances the training data across every sensitive group so that:

1. Training data in each **group** have the same total weight.

2. Class distributions are the same across all **groups**.

Here, each **group** is a possible combination of different values from each sensitive attribute. The hypothesis behind is:

On datasets with unbiased ground truth labels, ethical biases in the learned models largely attribute to the training data having (1) difference in group size and (2) difference in class distribution within each group.

For example, (1) in a certain dataset if the majority of the data belong to a certain gender, the cost for misclassifying data points of the majority gender would be much higher than that of the minority gender, the machine learning model trained on this dataset will be inclined to underfit on data points of the minority gender; (2) in a certain dataset if the majority of a certain gender would be more successful than the other, the machine learning model trained on this dataset will be inclined to believe these falsehoods.

In addition to improving fairness, FairBalanceClass, a variant of FairBalance, also balances the class distribution. This can be very useful when we care about the model's prediction performance (e.g. precision, recall, and F_1 score) on a minority class.

To validate our hypothesis and the effectiveness of FairBalance, we explore and answer the following research questions with experiments on three commonly used datasets with multiple sensitive attributes and class labels derived from truth:

6.1.0.0.1 RQ1: Can FairBalance mitigate machine learning bias against multiple sensitive attributes with low computational overhead?

Tested on three widely-used machine learning fairness datasets each with two sensitive attributes, five commonly applied machine learning classifiers trained on the FairBalance preprocessed training data had reduced bias on every sensitive attribute while maintaining similar prediction performances with $O(n)$ computational overhead.

6.1.0.0.2 RQ2: Can FairBalance balance class distributions (FairBalanceClass) as well?

With the same experiment setup as RQ1, FairBalanceClass achieved higher F_1 score on the minority class while reducing bias on every sensitive attribute. This suggests that class balancing is effective in FairBalanceClass.

6.1.0.0.3 RQ3: How does FairBalance perform compared to the existing state-of-the-art bias mitigation algorithms?

With the same experiment setup as RQ1, the performance of FairBalance was compared against other state-of-the-art bias mitigation algorithms. As discussed in Section 6.4), FairBalance is the only known algorithm that improves fairness on multiple sensitive attributes *without* damaging prediction performance *and* does not require hyperparameter tuning *and* is compatible with different classifiers *and* has far lower computational overhead than other algorithms.

6.1.0.0.4 RQ4: Are (1) difference in group size and (2) difference in class distribution within each group the main reasons for machine learning bias when class labels are unbiased?

In Section 6.3, we compared the bias mitigation performance of (0) no data balancing with (1) only balancing the class distributions within each group and (2) balancing both group sizes and class distributions within each group. With results in Section 6.4, we confirmed that when class labels are unbiased, (2) difference in class distribution within each group is the most important reason for machine learning bias, and (1) difference in group size also contributes to it. Therefore, our hypothesis is valid and these two are the main reasons causing machine learning bias when class labels are unbiased.

Overall, the **significance** of this paper includes:

- On datasets with unbiased class labels and multiple sensitive attributes, we show that the proposed approach FairBalance significantly outperformed other state-of-the-art algorithms in terms of reducing ethical bias while maintaining prediction performance. FairBalance can be considered as a standard procedure for training machine learning software on datasets with unbiased class labels and multiple sensitive attributes, given its advantage in performance, low computational overhead ($O(n)$), easy-to-use, and compatibility with most existing machine learning algorithms.

- **RQ4** validated our hypothesis: on datasets with unbiased ground truth labels, ethical biases in the learned models largely attribute to the training data having (1) difference in group size and (2) difference in class distribution within each group. This provides our software engineering community with a better understanding of the causes of ethical bias in machine learning software.

6.2 FairBalance

The intuition behind FairBalance is that, a machine learning model will be most likely biased if the training data violates either of the following two balance criteria:

1. Training data in each **group** have the same total weight.
2. Class distributions are the same across all **groups**.

Here, a **group** $g \in G$ is a set of data points sharing the same combination of sensitive attributes. For example, all the data points with *sex=male* and *race=white* form one group.

Given that most machine learning algorithms learn to minimize a loss function of misclassification errors, we can analyze the following two violation examples:

1. Difference in group sizes:
$$|X(g_1)| < |X(g_2)|.$$
The cost for misclassifying data points in g_1 will be lower than that in g_2 since data points in g_1 appear less frequently. As a result, the learned model will tend to predict more accurately on data points in g_2 than g_1. This problem can be resolved by assigning higher weight on misclassification errors for g_1.
2. Difference in class distributions within each group:
$$|X(g_1,T)|/|X(g_1,F)| < |X(g_2,T)|/|X(g_2,F)|.$$
The learned model will tend to predict more as negatives (resulting in fewer false positives but more false negatives) in g_1 than in g_2, since positive examples appear less frequently in g_1. This also leads to large SPD, AOD, and EOD. This problem can be resolved by adding higher weight on the positive data points in g_1 and negative data points in g_2.

As shown in (6.1), FairBalance assigns different weights to data from different groups to satisfy the above two criteria. FairBalance is different from **Reweighing** since **Reweighing** does not guarantee the first criterion— training data in each **group** g have the same total weight. To validate the importance of this first criterion, in Section 6.4 we will also compare FairBalance with **Reweighing-Multiple** in (6.2), an extended version of **Reweighing** on multiple sensitive attributes.

$$W(g,c) = \frac{|X(c)|}{|X(g,c)|}, \qquad \forall g \in G, c \in C. \tag{6.1}$$

$$W(g,c) = \frac{|X(g)| \times |X(c)|}{|X(g,c)| \times |X|}, \qquad \forall g \in G, c \in C. \tag{6.2}$$

In addition to balancing the data for fairness, it is sometimes as important to balance the classes in the training data. This is especially true when we care about the prediction performance on the minority class. As Yan et al. [Yan20a] stressed, machine learning bias may increase after class balancing, due to the process being oblivious of the inherent properties of the datasets. To resolve the class imbalance problem, we propose FairBalanceClass, which is a variant of FairBalance. FairBalanceClass balances the training data for both fairness and class balance, as shown in (6.3).

$$W(g,c) = \frac{1}{|X(g,c)|}, \qquad \forall g \in G, c \in C. \tag{6.3}$$

Algorithm 3 shows the pseudo code for the proposed FairBalance (class_balance = False) and FairBalanceClass (class_balance = True) algorithms. The computational overhead for FairBalance is low— $O(n)$. Although in this study, we use binary classification problems with two sensitive attributes, FairBalance can be applied to multi-class classification problems with multiple sensitive attributes according to Algorithm 3. FairBalance is also compatible with different classifiers since it only preprocesses the training data.

Algorithm 3: FairBalance.

Input **:X**, training data. **G**, groups. **C**, classes.
class_balance, whether to balance class.
Output **:W(X)**, balanced weights on the training data.

```
if class_balance == True then
    // FairBalanceClass
    for c ∈ C do
        W(c) = 1
else
    // FairBalance
    for c ∈ C do
        W(c) = |X(c)|
for g ∈ G do
    for c ∈ C do
        W(g, c) = W(c) / |X(g) ∩ X(c)|
for x ∈ X do
    // Weights by group and class.
    W(x) = W(x(G), x(C))
return W(X)
```

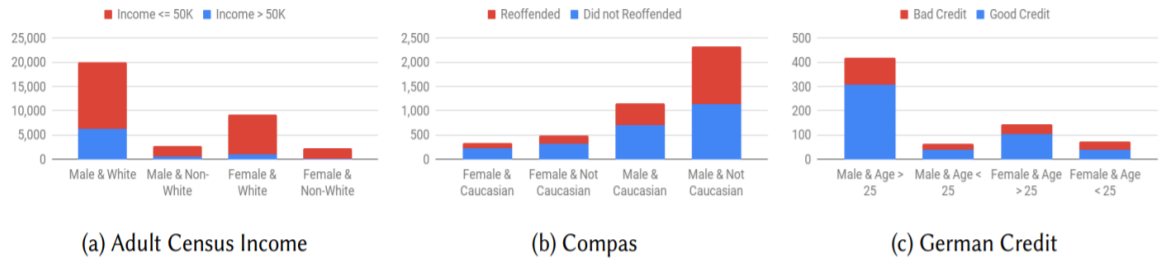

Figure 6.1 Group distribution in absolute values. Group sizes are very different in all three datasets.

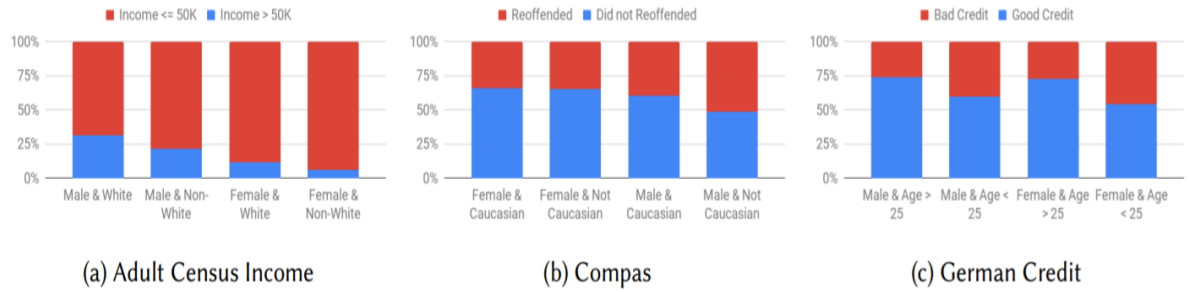

Figure 6.2 Group distribution in percentage. Class distributions within each group are very different in all three datasets.

6.3 Experiments

In this section, we present the experiment setups for answering the four research questions.

6.3.1 Experiment Design

We conducted two separate experiments to answer the four research questions. For each experiment, the following process was repeated for 50 times to enable statistical analysis (described in the next subsection).

Our first experiment answers **RQ1** and **RQ2** by comparing FairBalance and its class balancing variant FairBalanceClass against no bias mitigation (**None**). In this experiment, each dataset is randomly split into 70% training data and 30% test data every time. Standard scaler is then applied to normalize both training and test data based on the training data. The three different treatments are applied to the normalized training data:

- **None**: No transformation of the training data.
- **FairBalance**: Assign different weights to data points belonging to different groups to reach the same total weight for each group, as shown in (6.1) and Algorithm 3.
- **FairBalanceClass**: Assign different weights to data points belonging to different groups and

classes to reach the same total weight for each group and class, as shown in (6.3) and Algorithm 3.

After applying different treatments, five classical classification algorithms are applied to learn from the preprocessed training data and then collect their performances on the test data. The five classification algorithms are:

- **LR:** Logistic Regression Classifier implemented with scikit-learn[1].
- **SVM:** Linear Support Vector Machine Classifier implemented with scikit-learn.
- **DT:** Decision Tree Classifier implemented with scikit-learn.
- **RF:** Random Forest Classifier implemented with scikit-learn.
- **NB:** Gaussian Naive Bayes Classifier from scikit-learn.

The second experiment answers **RQ3** by comparing FairBalance and FairBalanceClass against five selected state-of-the-art bias mitigation algorithms. In this experiment, each dataset is randomly split into 70% training data and 30% test data every time. Standard scaler is then applied to normalize both training and test data based on the training data. Logistic regression classifier is applied as the base classifier (when the treatment does not specify a classifier). This process was repeated for 10 times for Reject Option Classification due to a memory leak problem in AIF360 [Bel18]. Here we describe the five baseline bias mitigation algorithms used in this study:

- **Reweighing [Kam12b]:** The most commonly applied pre-processing bias mitigation algorithm implemented with AIF360[2] under Apache License 2.0.
- **Fair-SMOTE [Cha21]:** The latest open source pre-processing bias mitigation algorithm implemented at `https://github.com/joymallyac/Fair-SMOTE` under Apache License 2.0.
- **Adversial Debiasing [Zha18]:** An in-processing bias mitigation algorithm implemented with AIF360.
- **Reject Option Classification [Kam18]:** A post-processing bias mitigation algorithm implemented with AIF360.
- **FERMI:** The most state-of-the-art in-processing bias mitigation algorithm which is able to mitigate bias on multiple sensitive attributes simultaneously[3].

Here, Reweighing, Adversial Debiasing, and Reject Option Classification are selected due to their popularity in each category and the fact that they have already been implemented in IBM AIF360 [Bel18].

[1] `https://scikit-learn.org`
[2] `https://github.com/Trusted-AI/AIF360`
[3] `https://github.com/optimization-for-data-driven-science/FERMI`

Fair-SMOTE is selected since it is a similar pre-processing algorithm and is proposed in a recent high-profile paper [Cha21] in software engineering. FERMI [Low21] is selected since it is the most state-of-the-art in-processing to handle multiple sensitive attributes.

Since most of the above bias mitigation algorithms (except for **FERMI**) are designed for data with single sensitive attribute, we also include the following variants of **Reweighing** and **Fair-SMOTE**:

- **Reweighing-Multiple: Reweighing** only works for single sensitive attribute when proposed [Kam12b]. However, it is easy to extend it to multiple sensitive attributes, as described in (6.2). Compared to FairBalance, this algorithm only balances class distributions within each group, does not balance the group sizes. Therefore, comparing the performance of **Reweighing-Multiple** with **None** will show the effectiveness of balancing class distributions within each group. Comparing the performance of **Reweighing-Multiple** with **FairBalance** will show the effectiveness of balancing the group sizes. **RQ4** can thus be answered with these comparisons.
- **Fair-SMOTE-Multiple:** an extended version of Fair-SMOTE to data with multiple sensitive attributes [Cha21]. This will be the first time **Fair-SMOTE-Multiple** being thoroughly tested on datasets with multiple sensitive attributes.

RQ4 will be answered by comparing **Reweighing-Multiple** (fixing only the second balance criterion) with **FairBalance** (fixing both balance criteria) and **None** (fixing none of the balance criteria).

Algorithm 4: Nonparametric ranking.

Input : **T**, performances to rank, a list of list.
Output : **R**, rankings of the each treatment (row) in **T**.

```
medians = []
for t ∈ T do
    medians.append(median(t))
asc = argsort(medians)
base = T[asc[0]]
rank = 0
R = []
R[asc[0]] = 0
for i=1, i<m, i++ do
    if MannWhitneyU (T[asc[i]], base) < 0.05 & CliffsDelta (T[asc[i]], base) > 0.33 then
        rank = rank + 1
        base = T[asc[i]]
    R[asc[i]] = rank
return R
```

Table 6.1 Comparisons of performances before and after FairBalance. Each dataset has two sensitive attributes. The second sensitive attribute is "Age" for the German dataset and "Race" for the other two. Medians (and IQRs) are reported for 50 repeats. Colored cells represent whether they are significantly better than the chosen baseline (None) with effect size of small (light green), medium (darker green), or large (darkest green) or significantly worse than the chosen baseline (None) with effect size of small (light red), medium (darker red), or large (darkest red).

Base Classifier	Dataset	Treatment	F_1	Accuracy	Runtime (sec)	Sex			Race / Age		
						AOD	EOD	SPD	AOD	EOD	SPD
LR	compas	None	60 (1)	67 (1)	1 (0)	16 (3)	20 (5)	19 (2)	16 (2)	20 (5)	17 (2)
		FairBalance	60 (1)	66 (1)	2 (0)	3 (4)	4 (7)	6 (4)	6 (7)	9 (8)	9 (7)
		FairBalanceClass	62 (1)	65 (0)	2 (0)	-2 (5)	-1 (5)	1 (5)	-2 (5)	0 (5)	0 (4)
	adult	None	48 (1)	80 (0)	10 (0)	-28 (1)	-45 (1)	-21 (0)	-11 (1)	-17 (3)	-10 (0)
		FairBalance	50 (0)	78 (0)	23 (0)	0 (1)	0 (3)	-6 (0)	0 (1)	1 (3)	-4 (1)
		FairBalanceClass	55 (0)	73 (1)	22 (1)	0 (4)	-1 (5)	-8 (4)	-2 (2)	-1 (5)	-6 (2)
	german	None	21 (6)	70 (3)	0 (0)	16 (9)	25 (14)	12 (8)	38 (19)	48 (17)	36 (20)
		FairBalance	2 (6)	70 (3)	0 (0)	0 (1)	0 (3)	0 (1)	0 (1)	0 (2)	0 (1)
		FairBalanceClass	50 (4)	59 (3)	0 (0)	2 (6)	4 (13)	4 (7)	6 (13)	4 (14)	11 (14)
SVM	compas	None	60 (1)	66 (1)	0 (0)	14 (4)	19 (6)	16 (3)	14 (3)	19 (4)	17 (2)
		FairBalance	60 (2)	66 (1)	2 (0)	4 (5)	4 (5)	8 (5)	7 (8)	9 (7)	10 (9)
		FairBalanceClass	62 (1)	65 (1)	2 (0)	0 (5)	0 (6)	3 (5)	-2 (4)	0 (4)	0 (4)
	adult	None	48 (0)	80 (0)	13 (1)	-27 (0)	-45 (0)	-20 (0)	-10 (1)	-16 (3)	-9 (0)
		FairBalance	49 (8)	79 (0)	24 (0)	-7 (8)	-12 (13)	-10 (6)	0 (4)	2 (6)	-3 (3)
		FairBalanceClass	54 (0)	73 (1)	24 (1)	0 (5)	0 (6)	-7 (5)	-1 (2)	0 (4)	-6 (2)
	german	None	16 (15)	70 (2)	0 (0)	12 (14)	20 (25)	11 (12)	26 (39)	34 (48)	24 (38)
		FairBalance	0 (3)	70 (2)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (1)	0 (0)
		FairBalanceClass	49 (4)	60 (3)	0 (0)	4 (7)	5 (10)	4 (8)	7 (7)	3 (11)	12 (7)
DT	compas	None	60 (2)	66 (1)	0 (0)	14 (5)	17 (8)	17 (6)	15 (4)	19 (4)	17 (4)
		FairBalance	60 (1)	66 (1)	2 (0)	3 (7)	5 (9)	6 (6)	7 (6)	11 (6)	10 (5)
		FairBalanceClass	61 (1)	64 (1)	2 (0)	3 (8)	4 (7)	6 (8)	-4 (8)	-1 (10)	-2 (8)
	adult	None	49 (2)	80 (0)	8 (0)	-28 (2)	-46 (3)	-21 (2)	-10 (2)	-17 (4)	-10 (1)
		FairBalance	42 (1)	78 (0)	19 (0)	8 (2)	13 (3)	0 (0)	0 (4)	1 (7)	-2 (1)
		FairBalanceClass	53 (0)	71 (0)	19 (0)	1 (5)	0 (5)	-5 (4)	4 (5)	6 (6)	-1 (6)
	german	None	22 (6)	70 (2)	0 (0)	17 (7)	23 (11)	15 (8)	36 (6)	48 (8)	33 (6)
		FairBalance	18 (13)	68 (3)	0 (0)	9 (14)	10 (23)	9 (12)	17 (26)	22 (37)	16 (23)
		FairBalanceClass	48 (4)	58 (3)	0 (0)	7 (13)	10 (15)	9 (12)	-7 (19)	-5 (20)	-4 (22)
RF	compas	None	60 (2)	66 (1)	17 (0)	14 (7)	17 (9)	17 (6)	14 (4)	19 (4)	17 (5)
		FairBalance	60 (1)	66 (1)	19 (0)	4 (5)	6 (5)	7 (5)	6 (6)	10 (6)	9 (6)
		FairBalanceClass	61 (1)	64 (0)	19 (0)	3 (11)	4 (13)	6 (11)	-2 (7)	0 (7)	0 (7)
	adult	None	48 (1)	80 (0)	106 (1)	-28 (1)	-45 (1)	-21 (0)	-10 (1)	-16 (3)	-9 (0)
		FairBalance	42 (1)	78 (0)	118 (2)	8 (1)	13 (3)	0 (0)	0 (2)	2 (4)	-2 (1)
		FairBalanceClass	54 (0)	71 (1)	117 (1)	0 (5)	0 (5)	-6 (4)	2 (5)	4 (6)	-4 (4)
	german	None	25 (5)	70 (2)	10 (0)	18 (8)	27 (13)	16 (7)	36 (9)	49 (11)	33 (10)
		FairBalance	20 (11)	69 (2)	10 (0)	13 (17)	17 (26)	11 (16)	27 (23)	36 (37)	26 (22)
		FairBalanceClass	50 (3)	59 (3)	10 (0)	5 (10)	7 (10)	7 (10)	-5 (17)	-6 (15)	-3 (14)
NB	compas	None	64 (1)	66 (1)	0 (0)	35 (2)	39 (4)	38 (2)	23 (3)	27 (5)	25 (3)
		FairBalance	62 (1)	64 (1)	2 (0)	9 (2)	7 (4)	13 (2)	12 (2)	14 (2)	14 (2)
		FairBalanceClass	62 (1)	64 (1)	2 (0)	8 (2)	8 (4)	12 (3)	13 (3)	14 (4)	15 (3)
	adult	None	50 (0)	56 (1)	8 (0)	-6 (1)	-5 (1)	-14 (1)	-4 (2)	-3 (1)	-8 (3)
		FairBalance	48 (0)	53 (0)	18 (0)	0 (1)	0 (1)	-5 (1)	-1 (1)	-1 (1)	-5 (1)
		FairBalanceClass	45 (3)	46 (7)	18 (0)	2 (1)	1 (1)	-1 (3)	-1 (1)	-1 (1)	-4 (1)
	german	None	48 (5)	62 (3)	0 (0)	17 (14)	22 (18)	19 (11)	26 (16)	24 (18)	29 (14)
		FairBalance	47 (4)	61 (3)	0 (0)	7 (11)	9 (13)	8 (8)	12 (7)	12 (16)	15 (5)
		FairBalanceClass	51 (4)	60 (3)	0 (0)	4 (6)	5 (12)	5 (6)	8 (8)	7 (9)	12 (8)

6.4 Results

In this section, we present the experimental results following the setups in Section 6.3.

RQ1. Can FairBalance mitigate machine learning bias against multiple sensitive attributes with low computational overhead?

Table 6.1 shows the results of the first experiment. Comparing the performance of FairBalance with None, we observe:

- F_1 score of FairBalance is usually comparable to that of None, except for the german dataset.

Accuracy of FairBalance is significantly worse than None in 8 out of 15 settings (with medium or large effect size). When comparing the median values of accuracy, FairBalance is only slightly worse than None (1-2%). This amount of decrease in Accuracy is as expected after applying a bias mitigation algorithm [Kam12b; Low21; Cha21].

- In every setting, the fairness metrics (AOD, EOD, and SPD) for every sensitive attribute are significantly reduced after applying FairBalance. This also indicates that FairBalance is compatible with all the base classifiers under test.
- The computational overhead for FairBalance is around 1 second on compas dataset, 10 seconds on adult dataset, and <1 second on german dataset. Given the size of datasets in Table 2.1, this observation is consistent with the analysis that the computational cost for FairBalance is $O(n)$.

The above three observations suggest that:

Answer to RQ1: FairBalance can reduce bias on every sensitive attribute while maintaining similar prediction performances with very low computational overhead— $O(n)$ and good compatibility to different classifiers.

RQ2. Can FairBalance balance class distributions (FairBalanceClass) as well?

Comparing the performance of FairBalanceClass with None and FairBalance, we observe:

- F_1 score of FairBalanceClass is significantly higher than that of None and FairBalance in every setting (except for NB). Accuracy of FairBalanceClass is significantly lower than that of FairBalance, which is as expected. This suggests that class balancing in FairBalanceClass is effective in trading accuracy for higher F_1 score on the minority class.
- In every setting, the fairness metrics (AOD, EOD, and SPD) for every sensitive attributes are significantly reduced (compared to None) after applying FairBalanceClass.

The above two observations suggest that:

Answer to RQ2: FairBalanceClass can reduce bias on every sensitive attribute while increasing the prediction performance on the minority class (when data is imbalanced).

RQ3. How does FairBalance perform compared to the existing state-of-the-art bias mitigation algorithms?

As described in Section 6.3, some of the baseline bias mitigation algorithms only work with single sensitive attribute. We split the experimental results into two tables. Table 6.2 shows the results of FairBalance and other bias mitigation algorithms focusing on single sensitive attribute. From this table, we observe:

Table 6.2 Comparisons between FairBalance and baseline bias mitigation algorithms focusing on single sensitive attribute. Logistic regression classifier is utilized as the base classifier if the treatment does not specify a classifier. The second column in Treatment represent the target sensitive attribute. *Rank: Medians (IQRs)* are reported for 50 repeats (10 repeats for Reject Option Classification due to memory leak). Treatments of the same rank are not significantly different (p-value for Mann-Witney Utest is larger than 5% or effect size is smaller than medium). Lower rank the better. All numbers reported are in percentage. The Column Total Rank sums up rankings of all metrics except for Runtime. Colored cells represent top ranks in each metric R0 (dark green) and R1 (light green).

Dataset	Treatment		F_1	Accuracy	Sex			Race / Age			**Total Rank**	Runtime (sec)
					AOD	EOD	SPD	AOD	EOD	SPD		
compas	Reweighing	sex	R1: 60 (1)	R0: 66 (0)	R1: 7 (5)	R2: 11 (6)	R1: 9 (5)	R2: 14 (3)	R4: 19 (5)	R3: 17 (3)	14	R1: 1 (0)
		race	R1: 61 (1)	R0: 66 (0)	R3: 19 (4)	R3: 22 (5)	R3: 22 (3)	R0: 0 (10)	R1: 4 (11)	R0: 1 (10)	11	R0: 1 (0)
	Fair-SMOTE	sex	R0: 61 (2)	R1: 65 (1)	R1: -5 (7)	R1: -3 (7)	R0: -1 (7)	R0: 5 (10)	R2: 9 (10)	R1: 7 (10)	6	R5: 579 (82)
		race	R0: 62 (1)	R1: 65 (1)	R3: 20 (6)	R3: 21 (9)	R3: 23 (4)	R0: -6 (6)	R1: -2 (5)	R0: -3 (6)	11	R4: 395 (64)
	Adversial Debiasing	sex	R1: 61 (2)	R0: 66 (1)	R5: 27 (11)	R5: 33 (15)	R5: 29 (9)	R3: 19 (5)	R5: 24 (7)	R4: 21 (5)	28	R3: 226 (20)
		race	R1: 60 (2)	R0: 66 (1)	R2: 14 (9)	R3: 19 (10)	R2: 17 (8)	R1: 11 (8)	R3: 15 (8)	R2: 14 (8)	14	R3: 221 (19)
	Reject Option Classification	sex	R0: 62 (2)	R1: 66 (1)	R0: -2 (9)	R1: 2 (15)	R0: 0 (7)	R3: 19 (4)	R5: 22 (4)	R4: 21 (4)	14	R6: 1029 (66)
		race	R0: 62 (2)	R1: 66 (0)	R4: 22 (3)	R4: 23 (7)	R4: 25 (2)	R0: -1 (5)	R1: 2 (10)	R0: 0 (5)	14	R6: 1058 (43)
	FairBalance		R1: 60 (2)	R1: 66 (1)	R0: 4 (5)	R1: 5 (6)	R1: 8 (5)	R0: 3 (10)	R1: 6 (11)	R0: 6 (10)	5	R2: 2 (0)
	FairBalanceClass		R0: 62 (1)	R1: 65 (1)	R0: -1 (5)	R0: -1 (5)	R0: 2 (5)	R0: -3 (3)	R0: 0 (5)	R0: -1 (3)	1	R2: 2 (0)
adult	Reweighing	sex	R5: 49 (0)	R2: 78 (0)	R0: 0 (1)	R0: 0 (3)	R1: -6 (0)	R2: -14 (9)	R3: -21 (17)	R3: -12 (4)	16	R0: 14 (0)
		race	R6: 49 (1)	R0: 80 (0)	R2: -28 (0)	R2: -46 (0)	R4: -21 (0)	R1: 0 (1)	R0: 0 (3)	R0: -4 (0)	15	R1: 16 (1)
	Fair-SMOTE	sex	R2: 54 (0)	R6: 69 (2)	R0: -1 (1)	R0: -2 (1)	R3: -9 (1)	R3: -26 (4)	R4: -26 (5)	R5: -28 (5)	23	R7: 7710 (2871)
		race	R0: 57 (0)	R4: 72 (0)	R3: -32 (1)	R1: -35 (2)	R5: -37 (0)	R1: -2 (2)	R0: 0 (3)	R2: -8 (2)	16	R7: 8863 (1507)
	Adversial Debiasing	sex	R4: 50 (0)	R1: 79 (0)	R1: -3 (2)	R0: -5 (4)	R3: -8 (1)	R2: -14 (3)	R3: -21 (4)	R3: -13 (2)	17	R4: 1445 (33)
		race	R6: 48 (1)	R0: 80 (0)	R2: -29 (0)	R2: -46 (1)	R4: -22 (0)	R1: 0 (4)	R0: 1 (6)	R0: -3 (3)	15	R4: 1445 (31)
	Reject Option Classification	sex	R3: 53 (0)	R6: 70 (1)	R1: 4 (4)	R0: 3 (4)	R0: -4 (4)	R2: -17 (2)	R2: -17 (5)	R4: -21 (1)	18	R6: 5954 (242)
		race	R1: 57 (0)	R5: 71 (0)	R4: -34 (0)	R1: -37 (2)	R6: -39 (1)	R1: 2 (2)	R1: 3 (2)	R0: -3 (3)	19	R5: 5666 (52)
	FairBalance		R4: 50 (1)	R2: 78 (0)	R0: 0 (7)	R0: 0 (12)	R2: -6 (3)	R0: 0 (1)	R0: 1 (3)	R0: -4 (1)	8	R3: 19 (0)
	FairBalanceClass		R2: 54 (1)	R3: 73 (1)	R0: 0 (5)	R0: -2 (5)	R3: -8 (4)	R1: -1 (1)	R0: 0 (2)	R1: -6 (2)	10	R2: 18 (0)
german	Reweighing	sex	R3: 12 (12)	R0: 69 (3)	R1: 0 (11)	R1: 0 (14)	R1: 0 (10)	R2: 20 (24)	R2: 20 (29)	R3: 20 (23)	13	R0: 0 (0)
		age	R4: 4 (6)	R0: 69 (2)	R1: 1 (3)	R1: 2 (4)	R1: 1 (3)	R0: -1 (2)	R1: -1 (4)	R0: 0 (1)	8	R0: 0 (0)
	Fair-SMOTE	sex	R0: 50 (5)	R1: 62 (2)	R1: 2 (9)	R1: 0 (13)	R2: 4 (8)	R2: 14 (11)	R2: 13 (16)	R2: 18 (10)	11	R3: 103 (25)
		age	R1: 50 (5)	R1: 60 (3)	R1: 2 (8)	R1: 4 (12)	R2: 2 (10)	R1: 4 (12)	R1: 1 (15)	R1: 7 (13)	9	R4: 123 (24)
	Adversial Debiasing	sex	R2: 28 (19)	R0: 68 (8)	R3: 13 (80)	R3: 22 (107)	R3: 11 (66)	R2: 21 (35)	R3: 26 (54)	R3: 20 (30)	19	R2: 67 (1)
		age	R2: 35 (14)	R1: 63 (34)	R2: -4 (38)	R2: -8 (49)	R2: -1 (37)	R3: -26 (144)	R4: -30 (148)	R3: -20 (143)	19	R2: 68 (0)
	Reject Option Classification	sex	R0: 53 (4)	R1: 60 (2)	R1: 3 (6)	R1: 0 (10)	R2: 5 (6)	R2: 16 (14)	R2: 14 (16)	R3: 22 (9)	12	R5: 272 (10)
		age	R1: 49 (4)	R1: 58 (4)	R2: 6 (9)	R1: 5 (8)	R2: 8 (7)	R1: 1 (9)	R1: -1 (11)	R1: 5 (12)	10	R5: 268 (3)
	FairBalance		R4: 4 (7)	R0: 70 (2)	R0: 0 (1)	R0: 0 (2)	R0: 0 (1)	R0: 0 (2)	R0: 0 (5)	R0: 0 (1)	4	R1: 0 (0)
	FairBalanceClass		R1: 50 (4)	R1: 59 (3)	R1: 1 (7)	R1: 0 (10)	R2: 2 (7)	R1: 5 (8)	R1: 4 (13)	R1: 10 (9)	9	R1: 0 (0)

- Overall, FairBalance and FairBalanceClass are the best treatments according to the total ranks. This is because the other treatments often perform poorly in terms of metrics on the non-target attribute. For example, Reweighing: race has very low AOD, EOD, and SPD on race but high AOD, EOD, and SPD on sex.
- Runtime of FairBalance and FairBalanceClass is orders of magnitude lower than the other treatments except for Reweighing. Runtime of Reweighing is a bit lower than the proposed method but they are at the same order of magnitude ($O(n)$).
- The median results of each fairness metric from FairBalance and FairBalanceClass are all below 0.10 on all three datasets. This suggests that applying either FairBalance or FairBalanceClass with logistic regression classifier can reduce the machine learning bias to a very low level.

The above observations show that it is very necessary to mitigate machine learning bias on multiple sensitive attributes simultaneously. For this reason, FairBalance is a better choice than the other baseline treatments on data with multiple sensitive attributes.

[3]Median values report the 50th percentile of the corresponding results while IQR values in the brackets present 75th percentile - 25th percentile of the results.

Table 6.3 Comparisons between FairBalance and baseline bias mitigation algorithms focusing on multiple sensitive attributes. FERMI: 30K represents its hyperparameter $\lambda = 30,000$ while FERMI: 10K represents $\lambda = 10,000$. *Rank: Medians (IQRs)* are reported for 50 repeats (10 repeats for Reject Option Classification due to memory leak). Treatments of the same rank are not significantly different (p-value for Mann-Witney Utest is larger than 5% or effect size is smaller than medium). Lower rank the better. All numbers reported are in percentage. The Column Total Rank sums up rankings of all metrics except for Runtime. Colored cells represent top ranks in each metric R0 (dark green) and R1 (light green).

Dataset	Treatment	F_1	Accuracy	Sex			Race / Age			**Total Rank**	Runtime (sec)
				AOD	EOD	SPD	AOD	EOD	SPD		
compas	Fair-SMOTE-Multiple	R0: 62 (1)	R1: 65 (1)	R0: 0 (10)	R0: 0 (8)	R0: 2 (9)	R0: -2 (5)	R0: 1 (7)	R0: 0 (5)	1	R3: 400 (54)
	FERMI: 30K	R0: 62 (1)	R1: 65 (0)	R0: 1 (3)	R0: 2 (6)	R1: 4 (3)	R1: -6 (3)	R0: -3 (4)	R0: -4 (4)	3	R2: 53 (5)
	FERMI: 10K	R0: 62 (1)	R1: 65 (1)	R0: 4 (4)	R0: 4 (7)	R2: 7 (4)	R0: -4 (5)	R0: -1 (6)	R0: -2 (4)	3	R2: 51 (2)
	Reweighing-Multiple	R1: 60 (1)	R0: 66 (1)	R0: 2 (3)	R0: 3 (5)	R2: 7 (3)	R1: 5 (4)	R1: 9 (6)	R1: 8 (4)	6	R1: 2 (0)
	FairBalance	R1: 60 (2)	R0: 66 (1)	R0: 4 (5)	R0: 5 (6)	R2: 8 (5)	R0: 3 (10)	R1: 6 (11)	R1: 6 (10)	5	R0: 2 (0)
	FairBalanceClass	R0: 62 (1)	R1: 65 (1)	R0: -1 (5)	R0: -1 (5)	R0: 2 (5)	R0: -3 (3)	R0: 0 (5)	R0: -1 (3)	1	R0: 2 (0)
adult	Fair-SMOTE-Multiple	R0: 55 (0)	R2: 72 (1)	R2: -9 (6)	R1: -10 (6)	R3: -17 (6)	R2: -8 (6)	R2: -8 (7)	R2: -13 (5)	14	R5: 6002 (83)
	FERMI: 30K	R2: 51 (0)	R4: 64 (1)	R1: 3 (4)	R0: 3 (4)	R0: -3 (4)	R2: 7 (4)	R2: 8 (4)	R0: 3 (4)	11	R4: 1264 (23)
	FERMI: 10K	R0: 55 (0)	R3: 69 (0)	R3: -26 (1)	R2: -26 (2)	R4: -33 (1)	R2: -9 (3)	R1: -6 (3)	R2: -15 (2)	17	R3: 63 (0)
	Reweighing-Multiple	R3: 50 (7)	R0: 78 (0)	R0: 0 (5)	R0: -2 (9)	R1: -6 (1)	R1: 1 (4)	R1: 4 (7)	R0: -3 (3)	6	R2: 24 (1)
	FairBalance	R3: 50 (1)	R0: 78 (0)	R0: 0 (7)	R0: 0 (12)	R1: -6 (3)	R0: 0 (1)	R0: 1 (3)	R0: -4 (1)	4	R1: 19 (0)
	FairBalanceClass	R1: 54 (1)	R1: 73 (1)	R0: 0 (5)	R0: -2 (5)	R2: -8 (4)	R1: -1 (1)	R0: 0 (2)	R1: -6 (2)	6	R0: 18 (0)
german	Fair-SMOTE-Multiple	R0: 49 (5)	R1: 61 (3)	R2: 3 (7)	R2: 6 (10)	R2: 5 (10)	R2: 9 (13)	R2: 10 (18)	R2: 11 (13)	13	R4: 121 (5)
	FERMI: 30K	R2: 0 (0)	R0: 69 (2)	R0: 0 (0)	R0: 0 (0)	R0: 0 (0)	R0: 0 (0)	R0: 0 (0)	R0: 0 (0)	2	R3: 32 (0)
	FERMI: 10K	R2: 0 (0)	R0: 69 (3)	R0: 0 (0)	R0: 0 (0)	R0: 0 (0)	R0: 0 (0)	R0: 0 (0)	R0: 0 (0)	2	R2: 31 (0)
	Reweighing-Multiple	R1: 2 (5)	R0: 69 (2)	R1: 0 (1)	R1: 0 (0)	R1: 0 (0)	R1: 0 (1)	R1: 0 (3)	R1: 0 (1)	7	R1: 0 (0)
	FairBalance	R1: 4 (7)	R0: 70 (2)	R1: 0 (1)	R1: 0 (2)	R1: 0 (1)	R1: 0 (2)	R1: 0 (5)	R1: 0 (1)	7	R0: 0 (0)
	FairBalanceClass	R0: 50 (4)	R1: 59 (3)	R2: 1 (7)	R2: 0 (10)	R2: 2 (7)	R2: 5 (8)	R2: 4 (13)	R2: 10 (9)	13	R0: 0 (0)

Next, we compare FairBalance against baseline treatments which are able to mitigate machine learning bias on multiple sensitive attributes simultaneously. From Table 6.3, we observe:

- On the german dataset, only **Fair-SMOTE-Multiple** and **FairBalanceClass**, the two treatments with class balancing, achieved acceptable F_1 scores. The other treatments all have close to 0 F_1 scores on the minority class. This suggests that the learned model predicted (almost) every data point as the majority class. Therefore, although **Fair-SMOTE-Multiple** and **FairBalanceClass** have the worst total ranks, they are the best and the only acceptable treatments on the german dataset. This also suggests that class balancing is very important on datasets with severe class imbalance.

- On compas and german datasets, **FairBalanceClass** performed similarly with **Fair-SMOTE-Multiple**. However, **FairBalanceClass** outperformed **Fair-SMOTE-Multiple** in the adult dataset.

- **FairBalanceClass** outperformed **FERMI** on all three datasets.

- **FairBalanceClass** outperformed **Reweighing-Multiple** on compas and german datasets. On the adult datasets, these two treatments achieved the same total ranks.

- **FairBalance** outperformed **Reweighing-Multiple** on compas and adult datasets. On the german datasets, these two treatments achieved the same total ranks.

- **FairBalanceClass** outperformed **FairBalance** on compas and german datasets while **FairBalance** outperformed **FairBalanceClass** on the adult dataset.

- **FairBalance** and **FairBalanceClass** have the shortest runtime on every dataset.

From the above observations, **FairBalanceClass** outperformed all other treatments except for **FairBalance**. Overall, we would recommend using **FairBalanceClass** to avoid critical failures such as the result of **FairBalance** on the german dataset.

Answer to RQ3: FairBalanceClass outperformed the existing state-of-the-art bias mitigation algorithms. It is recommended to apply FairBalanceClass to mitigate bias and balance the classes on datasets with multiple sensitive attributes and unbiased labels.

RQ4. Are (1) difference in group size and (2) difference in class distribution within each group the main reasons for machine learning bias when class labels are unbiased?

1. The fact that **FairBalance** outperformed **Reweighing-Multiple** in Table 6.3 indicates that difference in group size is a reason causing the machine learning bias when class labels are unbiased.

2. The fact that the fairness metrics of **Reweighing-Multiple** in Table 6.3 are much lower than those of **None** with logistic regression classifier in Table 6.1 indicates that difference in class distribution within each group is another reason causing the machine learning bias when class labels are unbiased.

3. Difference in class distribution within each group is more important for machine learning bias than difference in group size. This is because the reduction of fairness metrics after balancing the class distributions within each group is much larger than after balancing the group sizes.

Answer to RQ4: Yes, both (1) difference in group size and (2) difference in class distribution within each group are reasons for machine learning bias when class labels are unbiased. Difference in class distribution within each group is a more important reason than difference in group size.

6.5 Discussion

6.5.1 Limitations

The scope of this work limits its application. The results and conclusions of this work are limited to

- Datasets with unbiased class labels.

- Known sensitive attributes.

6.5.2 Threats to validity

Sampling Bias - We have used three datasets in this work. We could find only these three datasets that have more than one sensitive attribute. We have experimented with five different classification models. Conclusions may change a bit if other datasets and models are used.
Evaluation Bias - We used the three most popular fairness metrics in this study. Prior works [Cha;

Har16; Kam12c] only used one or two metrics although IBM AIF360 contains more than 50 metrics. In future, we will explore more evaluation criteria.

Conclusion Validity - One assumption of evaluating our experiments is that the test data is unbiased. Prior fairness studies also made similar assumption [Cha21; Bis20]. This assumption is true for the three datasets we used since their class labels were derived from truth. On other datasets with human decisions as class labels, this assumption may fail and we will need other ways to make sure the test data is unbiased.

External Validity - This work focuses on classification problems which are very common in AI software. We are currently working on extending it to regression models. The scope of this work also limits its generalizability to problems with unknown sensitive attributes and potentially biased class labels.

6.6 Conclusion

This paper proposes FairBalance, a pre-processing technique with low computational overhead for mitigating machine learning bias on multiple sensitive attributes. Our results show that, within the scope of this paper, FairBalance can significantly reduce machine learning bias while maintaining the prediction performance. Also, its class balancing variant FairBalanceClass consistently outperforms other state-of-the-art bias mitigation algorithms on datasets with multiple sensitive attributes. Note that the scope of this paper also limits the usage of FairBalance and FairBalanceClass—it cannot reduce the bias inherited from biased training labels. Our results also validated the hypothesis that on datasets with unbiased ground truth labels, ethical biases in the learned models largely attribute to the training data having (1) difference in group size and (2) difference in class distribution within each group.

To sum up, the best practice suggested in this paper is to apply FairBalance when the dataset has unbiased labels and multiple known sensitive attributes, and to apply FairBalanceClass if the classes are imbalanced as well.

CHAPTER

7

FAIR-SSL STUDY (SEMI-SUPERVISED LEARNING IN FAIRNESS)

All prior fairness works are supervised methods that require labeled training data. However, getting data with trustworthy ground truth is challenging and also ground truth can contain human bias. That is why, in this study, we used semi-supervised learning to achieve fairness. This study was published on Fairware workshop as part of ICSE 2022 [Cha22].

7.1 Introduction

Machine learning software has become ubiquitous in our society. Software is making autonomous decisions in criminal sentencing [Tol19], loan approvals [Fora], patient diagnosis [Med], hiring candidates [Hir], and whatnot. It is the duty of software researchers and engineers to produce high-quality software that always makes fair decisions. However, in recent times, there are numerous examples where machine learning software is found to have biased behavior based on some protected attributes like sex, race, age, marital status, etc. Google translate, the most popular translation engine in the world, shows gender bias. “She is an engineer, He is a nurse” is translated into Turkish and then again into English becomes “He is an engineer, She is a nurse” [Cal17a]. YouTube makes more mistakes when it automatically generates closed captions for videos with female than male voices [Tat17]. Amazon’s automated recruiting tool was found to be biased against women [Amad]. A widely used face recognition software was found to be biased against dark-skinned women [Ski]. Angel et al. commented that software showing bias is considered as poor quality software and should

not be used in real life applications [Ang]. It is time for software engineering researchers to dive into the field of software fairness and try to build fairer software to prevent discriminative behaviors.

A machine learning software can acquire bias in various ways [Bru18a]. Prior studies [Jia19; Che18] mentioned that most of the time bias comes from the training data. If training data contains improper labels, that bias gets induced into model while training. In an ACM SIGSOFT Distinguished award winning paper, Chakraborty et al. [Cha21] found out that bias comes from improper data labels and imbalanced data distribution. They said if the training data contains more examples of a certain group getting privileged (males being hired for a job) and another group getting betrayed (females getting more rejections); the machine learning model acquires that bias while training and in the future makes unfair predictions. Their algorithm, Fair-SMOTE, improved both the fairness and performance of the model and broke the premise of Berk et al. [Ber17b] who claimed *"It is impossible to achieve fairness and high performance simultaneously (except in trivial cases)"*.

We consider Fair-SMOTE [Cha21] as our baseline method. It is a supervised approach and uses 100% training data labels. But gathering good quality labeled data is very challenging. Human labeling is an extremely costly process [3ds; Goo; Tu20a; Tu21a] and there is a high possibility of human bias getting injected into the training data [JM19; Xia19]. That said, blindly trusting ground truth labels may induce bias in the machine learning model. Hence, it is timely to ask:

Can we reduce the labelling effort associated with building fair models?

In this work, we try to answer that question by using semi-supervised learning [Zhu06] that works with a small amount of labeled data and a large amount of unlabeled data. We build a framework called **Fair-Semi-Supervised-Learning (Fair-SSL)** that uses four state-of-the-art semi-supervised techniques - self-training, label propagation, label spreading, & co-training. Fair-SSL is a pseudo-labeling framework. It learns from the combination of labeled & unlabeled data and then pseudo-labels the unlabeled data. Results show that Fair-SSL performs as good as three other state-of-the-art fairness algorithms [Cha20b; Cha21; Cal17b]. That means even if available ground truth is corrupted or a very few labeled data points are available initially, fairness could still be achievable. Overall, this paper makes the following contributions:

- This is the first SE work using semi-supervised learning to generate fair classification models.
- Fair-SSL works with a very small amount of labeled training data (10%). Thus, we can avoid the costly process of data labeling. Hence, it is cost effective.
- We have shown a technique based on "situation testing" [Usa] to create fairly labeled data without using any human intervention.
- We have given a comparative analysis of four popular semi-supervised algorithms in the context of software fairness.
- Our results show that semi-supervised algorithms can be used to generate fairer and better performing models.

- To facilitate open science and replication, all our source code and datasets are publicly available at `https://github.com/joymallyac/FairSSL`.

7.2 Background: Semi-supervised Learning

Supervised machine learning models, especially the deep learning models, require a huge amount of labeled data for training. Gathering good quality labeled training data is the most expensive part of ML pipeline [3ds; Goo]. The majority of the time, data comes in the form of partly labeled and mostly unlabeled. While human beings can be used for data labeling, gathering human labelers with the appropriate domain knowledge is challenging and expensive [3ds; Goo]. Even then, the possibility of human bias getting injected into data is quite high [JM19; Xia19].

Semi-supervised learning (SSL) can address all these issues. SSL requires a small amount of labeled data to begin with [Zhu06], then using an incremental approach, unlabeled data is pseudo-labeled, and the combined data is used for model training. SSL has been used in various domains of software engineering such as defect prediction [Tu20a; Zha17], finding relevant papers [Yu18], test case prioritization [Yu19], static warning analysis [Tu21a], software vulnerability prediction [Yu19] and many more.

To the best of our knowledge, this paper is the first to try SSL in the context of fairness. Outside of SE, we found only one study by Zhang et al. [Zha20b] studying SSL and fairness. They used self-training (a type of SSL) with ensemble models to generate fair results. They made one assumption that the initial labeled data is fair and also used ensemble models which are very slow to train. Our work differs to theirs, as follows: (a) we did not assume the initial data as fair; (b) we used the concept of *situation testing* [Usa] to select fairly labeled data from the available data; (c) whereas they used only self-training, we used three other semi-supervised methods along with self-training. (d) we evaluated our results based on nine metrics, ten datasets, and three different learners whereas they used only two metrics, three datasets and two learners.

7.3 Methodology

Fair-SSL contains three major steps:

1. Select a small amount of labeled data (10%) in a way that initial labeling does not contain bias (see §7.3.1).
2. Pseudo-label the unlabeled data using semi-supervised approaches (see §7.3.2).
3. Balance the combined training data (labeled + pseudo labeled) based on protected attribute and class label (see §7.3.3).

Finally, we train ML models on the generated balanced data and test on the test data. For all the datasets, we divide them into 80% training and 20% testing. We keep test data completely separate and use them only for final score reporting.

7.3.1 Prepare the Fairly Labeled data

Our first task is preparing the initial fairly labeled set. All the datasets in Table 2.1 are already labeled. But are those labels fair? Can we just randomly pick one portion of that data as fairly labeled? How much labeled data is required to start with? We are going to find all the answers soon.

Prior studies [Cha21; Das20; Jia19; Sim20] have experimented with the datasets of Table 2.1 and found out that these datasets contain unfair labels. That means there are examples of biased ground truth based on protected attributes. Chakraborty et al. [Cha21] used the same datasets and found out that more or less 10% data labels contain unfair decisions. That means if we randomly pick up some portion of the data, we may end up selecting some improperly labeled rows and training ML models on that corrupted data will introduce bias. Thus, at an early stage, we decided to discard the available ground truth (or labels) and use human evaluation to re-label the data.

In literature, there are mainly two approaches for executing the labelling process. The first one is *manual labeling/crowdsourcing*. The second one is *semi-supervised pseudo-labeling* that starts with a small amount of labeled data and using that, pseudo-labels the remaining data. In this work, we tried both of them and got success with the latter one.

7.3.1.1 Crowdsourcing:

We decided to use crowdsourced workers via Amazon Mechanical Turk [Amac] to label these datasets. We applied best practices in crowdsourcing as described by Chen et al. [Che19a] and gave the workers more than the USA minimum wage [Sil18]. Instead of depending on a single opinion, for every data point, we considered using majority voting from five workers. That means if a data point is labeled positive by three workers and negative by two other workers, we mark it as positive. We used 200 Mechanical Turk workers and then took a break to evaluate their labeling. Unfortunately, we found out the labels given by crowdsourced workers are extremely noisy. One point to mention here is that we used "Gold" standard tasks [Alo11; Sar12] in our study to make sure the workers pay full attention while labeling. We selected only those responses where workers gave correct answers for gold questions. We understood lack of attention is not the reason for noise, the reason is lack of domain expertise. The datasets we use in the fairness domain are not very easy to be labeled by common people. Most of them [Adu; Ban; Def] are financial datasets; Compas [Com] is a criminal sentencing dataset; Heart health [Hea] is a medical dataset. Thus special expertise is needed to label these kinds of data. It was out of our scope to find that kind of experienced people. Also, this manual labeling is a super expensive process. We not only had to bear the charges from Amazon but also the university was taking a 50% overhead tax on grants for using crowdworkers. Another reason for caution is even if we could hire people having the domain expertise to label these kinds of datasets, there would still be a chance of human bias getting injected into the ground truth [JM19; Xia19]. Thus we decided to stop using human intervention and started to find an alternative cheap way of labeling.

7.3.1.2 Situation Testing:

Chakraborty et al. reported that in these datasets, around 10% of the labels are unfair labels [Cha21]. Thus we could select only fairly labeled rows from the available data and treat the rest as unlabeled data. By that way, our initial set will not contain unfair labels. To achieve that, we used the concept of *situation testing* [Zha16c]. It is a research technique used in the legal field [Usa] where decision makers' candid responses to applicant's personal characteristics are captured and analyzed.

In every dataset, a protected attribute divides the population into two groups - privileged and unprivileged. For example, in case of the "Adult" dataset, based on protected attribute "sex", privileged group is "male" and unprivileged group is "female". At first, we divide the data based on the protected attribute. Then, we train two different logistic regression models (any other simple statistical model can be used) on those two subgroups (for example - "male" and "female"). Then for all the training data points, we check the predictions of these two models. For a particular data point, if the predictions of two models match with the ground truth label, we keep the data point with the same label as it was fairly labeled. If the model predictions contradict with each other or the ground truth, we discard the label and treat the data point as unlabeled.

After this *situation testing* phase, we get fairly labeled data points. We call this set the *fairly labeled set*. But we do not use this whole set for training. We take a small portion (10%) of this set to start. This selection is not a random selection. The normal trend of semi-supervised learning is keeping the initial labeled data balanced based on class so that the semi-supervised model gets an opportunity to learn features from all the classes equally [Hyu20; Li11; Sta14; Zha17]. In fairness, data balancing helps more if instead of just class, protected attributes are also balanced [Che18; Yan20b; Cha21]. We apply the same balancing strategy while making the *initial fairly labeled set*. We select data points in a way that the *initial fairly labeled set* has equal proportion of "favorable-privileged (FP)" (for "Adult" dataset - "high income" & "male"), "favorable-unprivileged (FU)" ("high income" & "female"), "unfavorable-privileged (UP)" ("low income" & "male"), & "unfavorable-unprivileged (UU)" ("low income" & "female") samples. Figure 7.1 shows the block diagram of the combined process of situation testing and sampling that generates perfectly balanced *initial fairly labeled set*.

To summarize, at first, we do *situation testing* to create the *fairly labeled set*. Then we create an *initial fairly labeled set* by choosing samples from the *fairly labeled set* in a way that based on target class and protected attribute combination, we have an equal proportion of examples. In §7.4, we will show how the performance of the model changes based on the size of the *initial fairly labeled set*.

7.3.2 Pseudo Labeling the Unlabeled data

We have the *initial fairly labeled set* from the previous step. We remove the labels of the rest of the training data and treat it as *unlabeled data*. In this step, we pseudo-label the unlabeled data using four different semi-supervised techniques [Zhu06]. This is called *pseudo labeling* because the generated labels are predicted labels, not actual labels. These four approaches differ in internal logic.

But to make the study comprehensible, we followed a single pattern while applying these techniques. Algorithm 5 shows our approach. We start with labeled dataset D_l (*initial fairly labeled set*), and unlabeled dataset D_u. Our goal is to get a new training dataset (pseudo-labels added). To do that, we select a baseline model (based on the technique). The baseline model is trained on the *initial fairly labeled set*. Then baseline model predicts labels of the unlabeled data. We consider prediction of only those data points as valid where the confidence of the predictor is very high. There are two kinds of selection criterion used - (a) select k_best data points based on prediction probability, or (b) select data points where prediction probability is above a certain threshold. Based on scikit-learn semi-supervised article [Skla], the ideal probability threshold is 0.7. We have used the same value. The data points being predicted with more than 70% confidence along with their predicted labels are added to the training data. This process is repeated until max_iteration is reached. Now we will describe four semi-supervised approaches in detail.

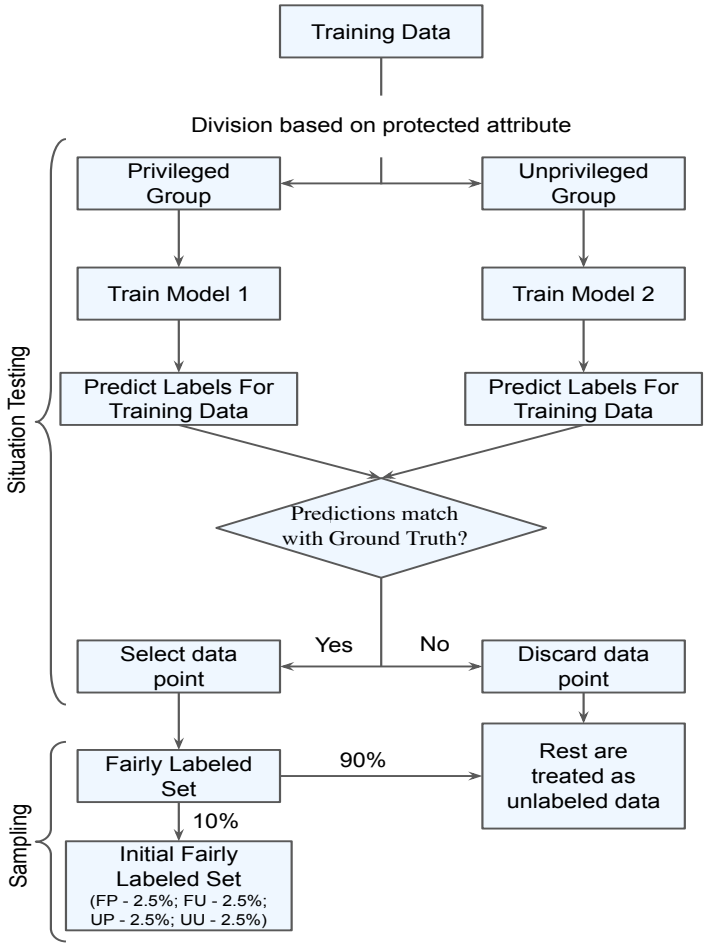

Figure 7.1 Algorithm (inspired from situation testing [Usa]) for selecting/sampling fairly labeled data points from the available data.

7.3.2.1 Self Training:

The self-training approach is based on Yarowsky et al.'s algorithm [Yar95]. The advantage of using self-training is that any supervised classifier with good calibration can be used as the baseline model. We have used logistic regression because it returns well calibrated predictions by default as it directly optimizes *log loss*. Prior works [Sklb; Log] found out logistic regression is a much better choice than random forest, naive bayes or svm as baseline model.

At first, a supervised classifier (here logistic regression) is trained on the *initial fairly labeled set* and then incrementally unlabeled data points are predicted. At each iteration, the data points having prediction probability more than "probability_threshold" (0.7) are selected and added to the training set with the predicted labels. This process continues until max_iteration is reached. Finally, as a result, we get a new training dataset which contains *initial fairly labeled set* and pseudo-labeled data points (by self-training).

7.3.2.2 Label Propagation:

Label propagation is a semi-supervised graph inference algorithm. The core algorithm was developed by Zhu et al [Zhu02]. It is a type of "community detection algorithm". The algorithm starts with building a graph from the available labeled and unlabeled data. Each data point is a node in the graph and edges are the similarity weights. The graph is represented in the form of a matrix. The algorithm has four main steps [Zhu02; Wu16].

- A unique label is assigned to each node in the network. At time t = 0, for a node x, let its label is $C_x(0) = x$
- The value of t is incremented. t += 1
- For each node x in the network, the most frequently occurring label among all the nodes with which x is connected is found. Ties are broken using uniform logic.

$$C_x(t) = f(C_{x1}(t-1), C_{x2}(t-2), ..., C_{xk}(t-k))$$

Algorithm 5: Pseudo-labeling Algorithm. We used probability_threshold = 0.7 and max_iteration = 100 in our implementation.

Input: Labeled dataset D_l, unlabeled dataset D_u, max_iteration, probability_threshold
Output: New training dataset (pseudolabels added)

1 Select a baseline model
2 Train baseline model on D_l
3 Predict on D_u
4 Select data points based on the selection criterion (probability_threshold)
5 Add selected data points with predicted class to the D_l
6 Re-train baseline model on D_l
7 Repeat steps 3-6 until max_iteration is reached

- Convergence condition is checked. If not met, we go to step 2, else stop.

It uses a 'kernel' function to project data into alternate dimensional spaces. We tried with 'rbf' and 'knn' kernels and found better performance with 'rbf'. The 'knn' kernel is memory-friendly because it creates a sparse matrix. As most of the elements in the sparse matrix are zero, all the matrix operations become faster. On the other hand, 'rbf' kernel generates a fully connected graph which is represented by a dense matrix. Thus, matrix operations become time consuming. So, 'rbf' kernel performs better but slower than 'knn' kernel. While reporting results, we reported 'rbf' kernel results for label propagation.

7.3.2.3 Label Spreading:

Label spreading is also a graph inference algorithm but has some differences from label propagation. The algorithm was invented by Zhou et al. [Zho04]. Label propagation uses the raw similarity matrix constructed from the data with no modifications. It believes that the original labels are perfect. On the contrary, label spreading does not blindly believe the original labels and make modifications to the ground truth. The method of changing the ground truth labels is called "clamping". Label propagation is a "hard clamping" approach because it does not change the original ground labels. In contrast, label spreading is a "soft clamping" approach and more robust to noise. The label spreading algorithm iterates on a modified version of the original graph and normalizes the edge weights by computing the normalized graph Laplacian matrix. Here also we tried with two different kernels ('rbf', 'knn') and got better results with 'rbf' kernel.

7.3.2.4 Co-Training:

Co-training is a very popular semi-supervised approach developed by Blum et al. [Blu98]. Here the feature set is divided into two mutually exclusive sets. Then two separate classifiers are trained on those two different feature sets (using the labeled data). Then both classifiers predict on the unlabeled data. For example, we take two classifiers *clf1* and *clf2*. The data points confidently predicted by *clf1* are used for *clf2* training and data points confidently predicted by *clf2* are used for *clf1* training. If clf1 and clf2 both are confident for a data point, that is added to the training-set with the predicted label. Co-training has had great success in the text mining [Nig00; Wan10] and the image domain [Bia16; Qia18]. The success of co-training depends on a very specific assumption. The assumption is "Original feature set can be divided into two mutually exclusive subsets which are conditionally independent given the class." Here we are using tabular (row-column) data and sub-diving features according to that assumption is not always possible. Thus, instead of using an off-the-shelf co-training approach, we developed a similar but slightly different *majority voting* technique. Our proposed majority voting algorithm is as follows:

- Build separate models from each attribute of *initial fairly labeled set*.
- Use each model to predict labels for the unlabeled data.

- For every data point, check predictions for all the models. Get the majority voting.
- Add the data point to the training set with the majority label.
- Incrementally repeat for all the unlabeled data points.

Here also, we use logistic regression model (one model per feature) for good calibration. The number of features in our datasets is not very high, thus this majority voting technique is feasible. In cases where the number of features is very high, instead of creating a new model for each feature, some kind of grouping of features (maybe choosing only important features based on information gain and then creating two or more subgroups with top K features) can be used. We keep this for our future work.

7.3.3 Synthetic Oversampling & Balancing

Our training data is now labeled. We can go for model training. But some prior works [Che18; Cha21] claimed that training data needs to be balanced in order to achieve fair prediction. Here data balancing means the number of examples based on protected attribute and class should be almost equal. We can undersample majority examples to obtain that data balance. But Yan et al. [Yan20b] said increasing the amount of training data instead of decreasing is likely to produce better trade-off between performance and fairness. Thus we decided to use synthetic generation of minority examples to balance the training data.

Traditional class balancing techniques such as SMOTE [Cha02] only balances the class by over-sampling the minority class. But, we want to balance based on two conditions - target class and protected attribute. For that, we use the Fair-SMOTE algorithm by Chakraborty et al. [Cha21]. At first, we divide the training data into four groups (Favorable & Privileged, Favorable & Unprivileged, Unfavorable & Privileged, Unfavorable & Unprivileged). Initially, these subgroups are of unequal sizes. Then we find out the group with maximum size. Our target is to create new data points of other groups to make them equal to the group with the maximum size. Fair-SMOTE synthetically generates new data points of minority groups. It starts with randomly selecting a data point (parent point p) from a minority sub-group. Then using K-nearest neighbor, two data points (c1, c2) are selected which are closest to p. Next, according to the algorithm described, a new data point is generated. There are two hyperparameters ("mutation amount" (f) and "crossover frequency" (cr)) that control the extrapolation. In our experiment, we used a grid search approach to find out the best values of cr & f. After synthetic oversampling, the training data becomes balanced based on target class and protected attribute i.e. above mentioned four groups become of equal sizes.

We have now described all the major components of Fair-SSL that generate pseudo-labeled, and balanced training data from a very small set of labeled examples. Figure 7.2 shows the Fair-SSL framework.

7.3.4 Experimental Setup

We conducted our experiments on ten datasets described on Table 2.1. We split the datasets using 5 fold cross-validation (train - 80%, test - 20%) and repeat 10 times with random seeds and finally report the median for every experiment. Standard data pre-processing techniques are used such as (a) ignoring rows with missing or corrupted values (b) continuous columns are converted to categorical (e.g., age<25: young, age>=25: old) (c) non-numerical features are converted to numerical (e.g., male: 1, female: 0) (d) finally, all the numerical feature values are normalized (converted between 0 to 1 using scikit–learn MinMaxScaler). Source code is written in Python and scikit–learn semi-supervised library [1] has been used. All experiments were conducted on a Windows laptop (x64) with a 2.7 GHz 8-Core Intel i7 CPU and 12GB of main memory. We run all the experiments for three different learners - a) logistic regression b) random forest & c) support vector machine (svm). We used scikit–learn version of these models along with off-the-shelf hyperparameters. Most of the prior works [Cha20b; Kam12c; Cal17b; Bis20; Cha19; Cha21] in fairness domain chose simple models like us instead of deep learning models. The main reason behind that is the fairness datasets are comparatively less complicated as they are small in size with respect to number of rows and columns. In future, we will use DL models if larger fairness datasets are available.

[1] https://scikit-learn.org/stable/modules/semi_supervised.html

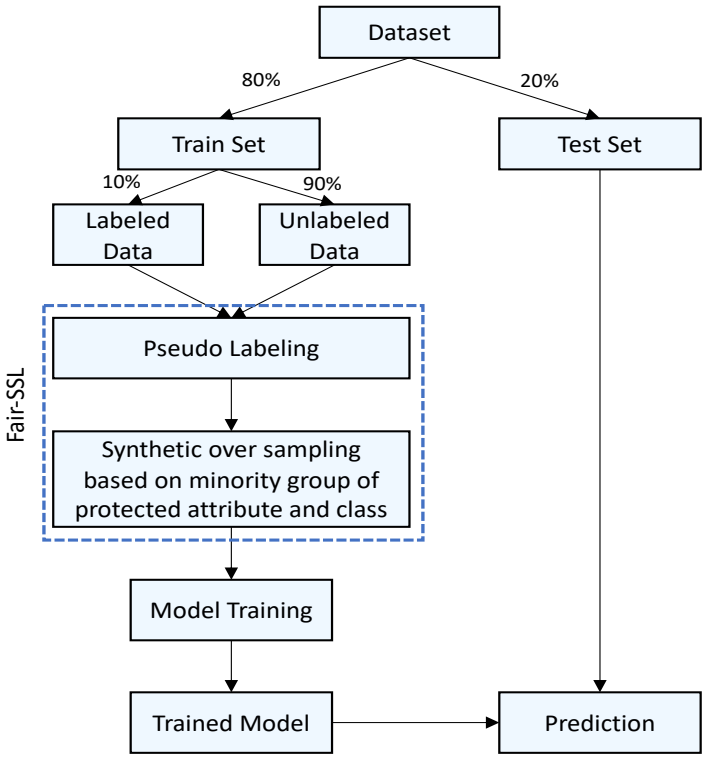

Figure 7.2 Framework of Fair-SSL

7.4 Results

Our results are structured around four research questions. For all the results, we repeat our experiments ten times with data shuffling and report the median. For answering RQ3 (Table 7.3), we vary the size of *initial fairly labeled set*. For all other cases (Table 7.1, 7.2, & Figure 7.3), size of *initial fairly labeled set* is 10% of the training data.

RQ1. Can Fair-SSL reduce bias?

The premise of this paper is finding out whether semi-supervised techniques can be helpful to achieve fairness or not. RQ1 directly asks that question. Table 7.1 answers the question. It contains the results for six datasets. "Default" rows signify when a logistic regression model is trained on the available labeled training data with no modification. We see for every dataset, the values of AOD, EOD, SPD, and DI are very high indicating bias in prediction. For every dataset, the last four rows show the results of Fair-SSL with four different algorithms used as pseudo-labeler inside. That means, 10% labeled training data is used and rest of the training data has been pseudo-labeled. After that combined training data is oversampled to be balanced based on protected attribute and class. Finally, the generated data has been used for model training. We see all four versions of Fair-SSL significantly reduce the values of four bias metrics (AOD, EOD, SPD, and DI) for all the datasets. That means Fair-SSL can successfully reduce bias. In Table 7.1, first five columns are showing the performance metrics. We see in some of the cases, recall and F1 are better than "Default". But there are some damage in case of false alarm, and accuracy. Prior fairness works [Cha20b; Kam18; Kam12a; Zha18; Cal17b] also damaged performance of the model while achieving fairness. This is called the "fairness-performance" trade-off in the literature. The answer for RQ1 is **"Yes, Fair-SSL can significantly reduce bias. It improves recall and F1 and sometimes damages false alarm, and accuracy."**

RQ2. How well does Fair-SSL perform compared to the state of the art bias mitigation algorithms?

Fair-SSL improves fairness of the prediction with a minor damage in performance. Here we are comparing Fair-SSL with three other state-of-the-art bias mitigation algorithms - Optimized Pre-processing [Cal17b], Fairway [Cha20b] and Fair-SMOTE [Cha20b]. If we look at Table 7.1, the last four columns are bias scores where Fair-SSL is performing as good as the others. In case of performance metrics (first five columns), we see it is losing sometimes but not by much. To get a clear picture, we take a look at Table 7.2. Here we show the summarized results for all ten datasets (Adult and Compas have two protected attributes each, i.e. $10 + 2 = 12$ cases) and three learners (logistic regression, random forest, and svm). We have implemented four different versions of Fair-SSL (ST, LP, LS, CT). For comparison purpose, for every dataset, we have created a validation set (20%) from the train set. On the validation set, we tried all four versions of Fair-SSL and chose the best one

to run on the test set. Thus, Table 7.2 contains comparison of the best version of Fair-SSL (best is selected based on validation results) with three other state-of-the-art bias mitigation algorithms. We see in bias scores, Fair-SSL is as good as other three. In 'recall' and 'F1 score', it performs better than Optimized Pre-processing and Fairway. One important point to mention here is that other three algorithms take advantage of full training data where Fair-SSL takes only 10% of labeled training data. The answer for RQ2 is **"Fair-SSL is as good as others in reducing bias, sometimes better in "recall" and "F1" than Optimized Pre-processing and Fairway."**

RQ3. How much labeled data is required to begin with?

Semi-supervised algorithms work with a small amount of labeled data and a large amount of unlabeled data. However, it is crucial to know how much labeled data is needed to start. Table 7.3 shows results for three datasets where size of the *initial fairly labeled set* has been varied from 1% to 20%. Here also we used the best version of Fair-SSL based on validation set results. We see the trend of increasing accuracy, F1 and decreasing AOD, EOD with increasing size of *initial fairly labeled set*. Even when using 5% labeled training data, we see Fair-SSL can significantly reduce bias. However, for accuracy and F1, Fair-SSL with 10% labeled training data performs much better than 5% version and very similar to 20% version. This gives a strong indication of semi-supervised learning being successful to achieve fairness. Hence, to answer RQ3, we say that **"Fair-SSL can significantly reduce bias even when a very small amount of labeled data points are available. Performance of Fair-SSL gets better with increasing size of initial labeled training set."**

RQ4. Which semi-supervised approach is the best to reduce bias?

The results so far can be summarized as follows: semi-supervised algorithms can be useful to augment bias mitigation process. That raises our next research question: which one of our four different semi-supervised techniques performs the best in this context?

Table 7.1 shows all four versions of Fair-SSL are reducing the bias scores. In case of performance metrics (the first five columns), all four of them are doing just the same (with minor differences). That means there is no way to choose one best method among the four methods. This reminds us of the popular "No free lunch theorem" for machine learning [Fed17]. We can certainly say semi-supervised pseudo-labeling improves fairness but which one is the best depends on the specific dataset. Thus it is wise to try all of them and find out the best one in case of real life applications.

When we are comparing the performance of the semi-supervised techniques, the execution time comparison also makes sense (as higher execution time means higher cost). Figure 7.3 shows the execution time (log scale) of four approaches for Adult and Compas dataset. We used 5-fold cross validation and 10 repeats i.e. every bar in the figure shows execution time for 5*10 = 50 runs. For both datasets, label propagation is the slowest and self-training is the fastest method. We used logistic regression as self-training baseline model. Choosing a different model may change the

Table 7.1 Results for RQ1, RQ2, and RQ4. In this table "Default" means off-the-shelf logistic regression; Optimized Pre-processing [Cal17b], Fairway [Cha20b], Fair-SMOTE [Cha21] are bias mitigation algorithms that use 100% training data labels. Fair-SSL uses 10% training data labels. Fair-SSL-ST is using self-training; Fair-SSL-LP is using label-propagation; Fair-SSL-LS is using label-spreading; Fair-SSL-CT is using co-training. Cells show medians for 10 runs. Here, the darkest pink cells show top rank (note: for the metrics with '+' more is better and for the metrics with '-' less is better). The lighter pink and lightest pink cells show rank two and rank three respectively; the white cells show the worst rank. Rankings were calculated via Scott-Knott test (§4.3.1).

Dataset	Protected Attribute	Algorithms	Recall (+)	False alarm (-)	Precision (+)	Accuracy (+)	F1 Score (+)	AOD (-)	EOD (-)	SPD (-)	DI (-)
Adult Census Income	Sex	Default	0.46	0.07	0.69	0.83	0.54	0.12	0.24	0.21	0.56
		OP	0.42	0.09	0.61	0.76	0.51	0.04	0.03	0.04	0.14
		Fairway	0.25	0.04	0.71	0.72	0.42	0.02	0.03	0.04	0.11
		Fair-SMOTE	0.71	0.25	0.51	0.73	0.62	0.01	0.02	0.03	0.08
		Fair-SSL-ST	0.71	0.39	0.42	0.62	0.51	0.04	0.03	0.07	0.12
		Fair-SSL-LP	0.72	0.38	0.45	0.66	0.53	0.03	0.03	0.04	0.05
		Fair-SSL-LS	0.72	0.31	0.42	0.71	0.55	0.03	0.04	0.06	0.08
		Fair-SSL-CT	0.76	0.35	0.44	0.68	0.57	0.06	0.04	0.03	0.08
Adult Census Income	Race	Default	0.44	0.07	0.69	0.81	0.52	0.06	0.15	0.16	0.52
		OP	0.38	0.06	0.66	0.78	0.48	0.03	0.02	0.05	0.21
		Fairway	0.36	0.04	0.70	0.73	0.44	0.03	0.02	0.06	0.31
		Fair-SMOTE	0.7	0.22	0.51	0.72	0.62	0.04	0.04	0.05	0.16
		Fair-SSL-ST	0.7	0.26	0.51	0.74	0.6	0.05	0.08	0.05	0.18
		Fair-SSL-LP	0.72	0.31	0.49	0.72	0.59	0.02	0.03	0.02	0.16
		Fair-SSL-LS	0.7	0.33	0.51	0.71	0.58	0.02	0.02	0.05	0.19
		Fair-SSL-CT	0.72	0.31	0.48	0.71	0.58	0.03	0.05	0.05	0.21
Compas	Sex	Default	0.67	0.38	0.66	0.64	0.61	0.09	0.19	0.18	0.28
		OP	0.71	0.36	0.64	0.62	0.60	0.04	0.03	0.05	0.08
		Fairway	0.56	0.22	0.57	0.58	0.58	0.03	0.03	0.06	0.08
		Fair-SMOTE	0.62	0.32	0.56	0.55	0.65	0.02	0.05	0.08	0.09
		Fair-SSL-ST	0.42	0.21	0.65	0.58	0.54	0.02	0.09	0.12	0.21
		Fair-SSL-LP	0.52	0.34	0.62	0.54	0.58	0.02	0.06	0.09	0.19
		Fair-SSL-LS	0.52	0.33	0.61	0.62	0.62	0.03	0.03	0.05	0.09
		Fair-SSL-CT	0.49	0.28	0.62	0.61	0.57	0.03	0.02	0.05	0.07
Compas	Race	Default	0.69	0.39	0.65	0.64	0.68	0.05	0.11	0.12	0.21
		OP	0.68	0.33	0.63	0.62	0.67	0.03	0.06	0.06	0.12
		Fairway	0.55	0.21	0.58	0.58	0.56	0.02	0.04	0.03	0.11
		Fair-SMOTE	0.62	0.30	0.56	0.55	0.66	0.01	0.05	0.06	0.11
		Fair-SSL-ST	0.49	0.24	0.66	0.60	0.57	0.04	0.07	0.08	0.12
		Fair-SSL-LP	0.51	0.23	0.67	0.59	0.58	0.03	0.09	0.12	0.11
		Fair-SSL-LS	0.51	0.27	0.66	0.62	0.61	0.03	0.07	0.08	0.14
		Fair-SSL-CT	0.58	0.35	0.64	0.63	0.63	0.03	0.04	0.05	0.09
German Credit	Sex	Default	0.94	0.81	0.75	0.76	0.82	0.11	0.08	0.14	0.15
		OP	0.75	0.73	0.71	0.73	0.71	0.04	0.05	0.06	0.12
		Fairway	0.78	0.76	0.68	0.65	0.73	0.05	0.04	0.07	0.11
		Fair-SMOTE	0.62	0.36	0.71	0.64	0.71	0.05	0.05	0.05	0.13
		Fair-SSL-ST	0.61	0.32	0.72	0.61	0.69	0.02	0.05	0.06	0.1
		Fair-SSL-LP	0.57	0.41	0.72	0.56	0.66	0.06	0.02	0.03	0.08
		Fair-SSL-LS	0.42	0.17	0.75	0.54	0.56	0.06	0.03	0.07	0.12
		Fair-SSL-CT	0.52	0.26	0.75	0.58	0.64	0.06	0.03	0.04	0.09
Default Credit	Sex	Default	0.25	0.07	0.70	0.78	0.34	0.05	0.08	0.06	0.35
		OP	0.28	0.06	0.65	0.70	0.32	0.01	0.02	0.03	0.09
		Fairway	0.21	0.04	0.67	0.67	0.33	0.01	0.04	0.03	0.12
		Fair-SMOTE	0.58	0.26	0.39	0.68	0.44	0.02	0.03	0.05	0.03
		Fair-SSL-ST	0.48	0.12	0.53	0.79	0.51	0.03	0.04	0.05	0.12
		Fair-SSL-LP	0.51	0.31	0.37	0.67	0.44	0.05	0.03	0.03	0.08
		Fair-SSL-LS	0.59	0.36	0.34	0.64	0.42	0.05	0.04	0.03	0.09
		Fair-SSL-CT	0.59	0.35	0.36	0.65	0.44	0.03	0.03	0.05	0.08
Bank Marketing	Age	Default	0.73	0.21	0.76	0.77	0.77	0.08	0.22	0.24	0.31
		OP	0.72	0.20	0.74	0.75	0.75	0.04	0.05	0.02	0.04
		Fairway	0.71	0.17	0.73	0.71	0.71	0.04	0.03	0.05	0.06
		Fair-SMOTE	0.76	0.16	0.72	0.72	0.74	0.04	0.05	0.05	0.03
		Fair-SSL-ST	0.66	0.37	0.6	0.64	0.64	0.04	0.08	0.05	0.11
		Fair-SSL-LP	0.57	0.37	0.58	0.62	0.56	0.02	0.03	0.07	0.1
		Fair-SSL-LS	0.62	0.35	0.62	0.64	0.62	0.07	0.09	0.07	0.12
		Fair-SSL-CT	0.69	0.18	0.74	0.72	0.71	0.05	0.02	0.09	0.21
Student Performance	Sex	Default	0.81	0.06	0.85	0.88	0.83	0.06	0.05	0.06	0.12
		OP	0.79	0.06	0.83	0.83	0.82	0.02	0.04	0.04	0.06
		Fairway	0.76	0.05	0.81	0.84	0.84	0.03	0.02	0.04	0.07
		Fair-SMOTE	0.87	0.10	0.84	0.87	0.86	0.04	0.04	0.04	0.08
		Fair-SSL-ST	0.84	0.14	0.83	0.83	0.83	0.03	0.02	0.01	0.03
		Fair-SSL-LP	0.83	0.19	0.82	0.82	0.83	0.04	0.03	0.07	0.21
		Fair-SSL-LS	0.84	0.14	0.79	0.84	0.82	0.03	0.02	0.05	0.05
		Fair-SSL-CT	0.84	0.09	0.84	0.85	0.83	0.04	0.02	0.03	0.08

Table 7.2 RQ2 results: Summarized information of comparing Fair-SSL with Optimized Pre-processing [Cal17b], Fairway [Cha20b], & Fair-SMOTE [Cha21] based on results of 10 datasets and three learners (logistic regression, random forest, and svm). For every dataset, we have chosen the best performing method among the four methods in case of Fair-SSL. Number of wins, ties, and losses are calculated based on Scott-Knott ranks for each metric. Highlighted cells show Fair-SSL is performing similar/better than state-of-the-art bias mitigation algorithms.

		Recall	False alarm	Precision	Accuracy	F1 Score	AOD	EOD	SPD	DI	Total
					Optimized Pre-processing vs Fair-SSL						
1	**Win**	8	2	4	3	10	2	2	1	2	34
2	**Tie**	21	25	27	28	24	32	33	32	32	254
3	**Loss**	7	9	5	5	2	2	1	3	2	36
4	**Win + Tie**	29	27	31	31	34	34	35	33	34	288/324
					Fairway vs Fair-SSL						
5	**Win**	10	2	5	5	19	2	3	3	4	53
6	**Tie**	24	17	24	27	14	30	31	32	31	230
7	**Loss**	2	17	7	4	3	4	2	1	1	41
8	**Win + Tie**	34	19	29	33	33	32	34	35	35	283/324
					Fair-SMOTE vs Fair-SSL						
9	**Win**	1	3	2	3	3	1	2	2	2	19
10	**Tie**	28	30	30	28	29	34	33	32	31	275
11	**Loss**	7	3	4	5	4	1	1	2	3	30
12	**Win + Tie**	29	33	32	31	32	35	35	34	33	294/324

scenario. Nevertheless, it would seem that label propagation and label spreading can be slow where self-training can be very fast if a simple statistical model is chosen.

Hence, our answer for RQ4 is **"Semi-supervised techniques help but there is no winner based on performance. We should try all the approaches for a particular dataset to find out the best one. However, based on execution time, self-training with simple statistical model works the fastest."**

7.5 Discussion: Why Fair-SSL?

Here we discuss what makes Fair-SSL unique and more useful than prior works in the fairness domain.

- **Inexpensive** - Real world data comes as a mixture of labeled and unlabeled form. Available data labels can be of poor quality. Hiring Crowdsource workers for human labeling is not always an affordable solution. Tu et al. estimated that labeling GitHub issues as buggy/non-buggy for 500 projects would require $320K [Tu20b]. That said it is worth exploring alternative inexpensive ways of data labeling. Fair-SSL pseudo-labeler serves that purpose. It takes a very small amount of labeled data and then pseudo-labels the unlabeled data. We believe this will definitely help the domains where finding good quality labeled data is really challenging and expensive.

- **Performance** - Fair-SSL performs similarly to the state-of-the-art supervised bias mitigation algorithms. Our results based on 10 datasets and three learners show that Fair-SSL can significantly reduce bias, improves recall & F1 score and sometimes damages accuracy, false alarm. To the best of our knowledge, all the prior supervised bias mitigation algorithms (except

Table 7.3 RQ3 results: The change of accuracy, F1, AOD and EOD for Fair-SSL with increasing size of labeled training data (learner is logistic regression). Dark pink = 1st Rank; Light pink = 2nd Rank; White = Last Rank

Dataset	Protected Attribute	Size of Labeled Set	Accuracy (+)	F1 Score (+)	AOD (-)	EOD (-)
Adult	Sex	1%	**0.61**	**0.44**	**0.04**	**0.06**
		5%	**0.68**	**0.51**	**0.03**	**0.02**
		10%	**0.71**	**0.54**	**0.03**	**0.03**
		20%	**0.71**	**0.58**	**0.02**	**0.02**
Compas	Sex	1%	**0.47**	**0.44**	**0.07**	**0.12**
		5%	**0.57**	**0.54**	**0.06**	**0.06**
		10%	**0.58**	**0.61**	**0.03**	**0.07**
		20%	**0.61**	**0.64**	**0.01**	**0.05**
Student	Sex	1%	**0.67**	**0.65**	**0.18**	**0.12**
		5%	**0.78**	**0.75**	**0.05**	**0.07**
		10%	**0.83**	**0.82**	**0.02**	**0.02**
		20%	**0.85**	**0.83**	**0.02**	**0.01**

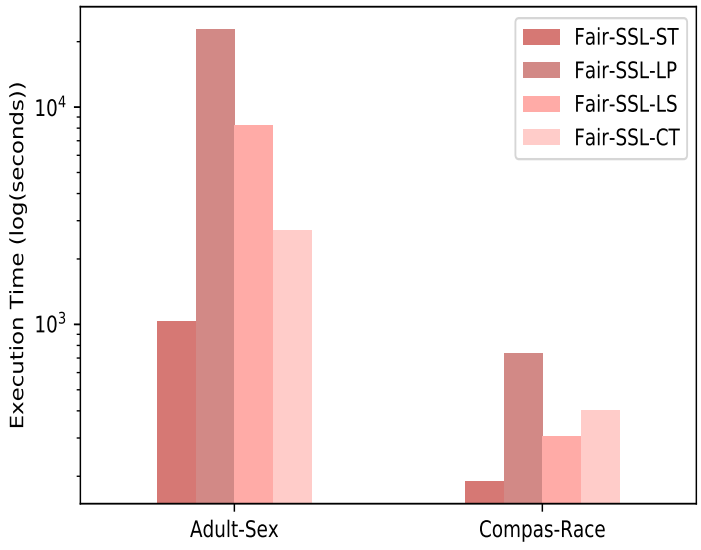

Figure 7.3 RQ4 results: Execution time comparison of the four semi-supervised methods for Adult (protected attribute - 'sex') and Compas (protected attribute - 'race') dataset.

Fair-SMOTE [Cha21]) damage performance of the model while making it fair [Bis20; Kle16; Tol19; Liu21]. Fair-SSL uses similar data balancing strategy as Fair-SMOTE and that is why it achieves similar results of higher recall & F1 but uses only 10% labeled data. However, the damage of accuracy, false alarm, and precision will remain a concern for future research. Since fairness always comes with a cost of affecting performance, it is the responsibility of the stakeholders to prioritize their objectives.

- **Model-agnostic** - Fair-SSL is a data pre-processor. That means after pseudo-labeling the data, any supervised model can be used for final prediction. Thus Fair-SSL is model agnostic.
- **Incremental Update** - In software industries, a common trend is incrementally updating the model or retraining the model with newer data. Most of the time this new data is unlabeled. Fair-SSL pseudo-labeler instead of supervised algorithms can be very handy in this situation as semi-supervised models are trained once but can be used for pseudo-labeling multiple times.

7.6 Threats to Validity

Sampling Bias - We have used ten well-known datasets and three classification models in our experiments. Most of the prior works [Cal17b; Gal17; Zha18; Kam18; Cha19] used one or two datasets where we used ten of them. In the future, we will explore more datasets and more learners. The conclusions may change a bit if other datasets and models are used.
Evaluation Bias - We used the four most popular fairness metrics in this study. Prior works [Cha20b; Har16; Kam12c] only used two or three metrics although IBM AIF360 contains more than 50 metrics. In the future work, we will use more evaluation criteria.
Conclusion Validity - One assumption of evaluating our experiments is the test data is unbiased. Prior fairness studies also made similar assumption [Cha21; Bis20; Cha19]. The only way to avoid this assumption will be use of domain experts that was beyond our scope.
Internal Validity - We used four semi-supervised algorithms with mostly off-the-shelf parameters (except a few cases such as kernel function selection). However, hyperparameters play a crucial role in the performance of ML models. In the future, it makes sense to do hyperparameter optimization for performance improvement [Tu21b; Tu18].
Construct validity - Semi-supervised learning can be helpful when bias comes from improper data labels or sampling. But there could be some other reasons causing bias as well such as objective function bias, homogenization bias [Das20], feature correlation bias [Zha16b]. Semi-supervised techniques may not be very helpful there.
External Validity - Our work is limited to binary classification and tabular data which are very common in AI software. However, all the methods used in this paper can easily be extended in case of multi-class classification, regression problems, and text mining. Our future work will be to experiment in other domains of SE and ML to see how semi-supervised methods fight against

ethical bias.

7.7 Conclusion

Fairness in machine learning software has become a serious concern in the software engineering community. Prior works mainly used supervised approaches to achieve fairness in ML models. Supervised approaches require data with ground truth labels. However, labeled data is not always available in real-life and even if available, human bias may be present in the ground truth. Keeping that in concern, this paper shows how semi-supervised techniques can be used to achieve fairness by using only 10% labeled training data. We have implemented and compared four most popular semi-supervised techniques by doing experiments on ten real-world datasets and three learners. Our results show that Fair-SSL is as good as reducing bias (with minor damage in performance) as three state-of-the-art bias mitigation algorithms. We have made our source code and datasets publicly available for future researchers. We believe our work will educate the software engineering community about machine learning fairness and also encourage more and more software researchers to work in the software fairness domain.

CHAPTER

8

FAIR ENOUGH: SEARCHING FOR SUFFICIENT MEASURES OF FAIRNESS

This work was done in collaboration with Suvodeep Majumder. Recent fairness research has proposed a plethora of new fairness metrics, for example, the dozens of fairness metrics in the IBM AIF360 toolkit. This raises the question: How can any fairness tool satisfy such a diverse range of goals? While we cannot completely simplify the task of fairness testing, we can certainly reduce the problem. This work shows that many of those fairness metrics effectively measure the same thing. This study was submitted on the Transactions on Software Engineering and Methodology, 2022

8.1 Introduction

A journal on software engineering methodologies needs to concern itself not just with one applications, but also general methods that hold across multiple applications. Recently the authors faced a methodological issue where reviewers challenged the validity of the metrics they used to assess that work. Prompted by that experience, we examined how the current SE research community selects metrics for assessing the *fairness of algorithmic decision making*.

On reading the literature, we found a general pattern: while the literature proposes a plethora of metrics[1], we could not find a principled argument (across a large space of known metrics) that it was necessary/unnecessary to report some metric X. This raises various methodological questions:

[1] E.g. the Fairlearn [Faib] tool lists 16 metrics; the Fairkit-learn tool [Joh20] comes with its own 16 metrics; IBM AIF360 toolkit [Bel18] offers 45 fairness metrics.

- Should we reject papers that "only" use (e.g.) five metrics? Or should researchers always use dozens of metrics?
- When we use automatic tools to optimize for fairness, should we optimize for dozens of goals? Or is optimizing for a smaller set sufficient?

To resolve these methodological concerns, we made the following conjecture. Given the large space of known metrics (such as the 30 studied in this paper), perhaps *many of these metrics are measuring the same thing*. As shown by the experiments of this paper, this is indeed the case, since we can cluster these 30 metrics into around half a dozen. While our results pertain a particular domain, there is nothing in principle stopping this methodology being applied to any domain where researchers keep proposing new metrics without first checking if the new metric is not just "old wine in new bottles"

As to the specifics of our domain, this paper concerns itself with measures of algorithmic fairness. Increasingly, software is being used for critical decision-making processes, such as patient release from hospitals [Str16; Med], credit card applications [DAV92], hiring [Sen19], and admissions [Aia]. According to guidelines from the European Union [Eu] and IEEE [Iee], a software cannot be used in real-life applications if it is found to be discriminatory toward an individual based on any sensitive attribute such as gender, race, or age. Hence "fairness testing" is now an open and pressing problem in software engineering.

As shown in Table 8.1, researchers have proposed a plethora of fairness metrics for measuring fairness, and that number is growing (e.g. see all the metrics proposed in [Faib; Joh20; Bel18]). Given that trend, is somewhat strange to report that researchers in this area only use a few metrics in their papers [Ple17; Gre19; Kal18; Wad18; Cel18; Fel15]. For example, in our literature review papers from the last three years, we see only a handful of papers (13 out of 60 to the best of our knowledge) using more than five fairness metrics to evaluate their method. This is surprising since all of them ignore more than half the known metrics. Is that wise?

The conjecture that is tested by this paper is that *there are too many spurious metrics* that *all measure very similar things*. If that were true, then it should be possible to simplify fairness assessment as follows:

> *Run metrics on real-world data. Find clusters of correlated metrics. Prune "insensitive clusters[2]". Only use one metric per surviving cluster.*

This paper experiments with seven datasets and finds that (a) 26 classification fairness metrics can be clustered into just seven groups; (b) four dataset metrics can be clustered into three groups and that (c) these clusters actually predict for different things. That is, it is no longer necessary (or even possible) to satisfy all these fairness metrics. Hence, to simplify fairness testing, we recommend (a) determining what type of fairness is desirable (and we offer a handful of such types); then (b) looking up those types in our clusters; then (c) testing for one item per cluster.

[2]Note: Here, by "insensitive" clusters, we mean those where changes to the data do not change the fairness scores.

Table 8.1 Mathematical definitions of the classification and dataset metrics used in this research. Definitions are collected from IBM AIF360 [Bel18], Fairkit-learn [Joh20] & Fairlearn [Faib]. For definitions of the terms used here, see Table 8.2.

(MID)	Metric Name	Description	Ideal Value	AIF360	Fairkit	Fairlearn
Classification Metrics						
C0	true_positive_rate_difference	$TPR_{D=unprivileged} - TPR_{D=privileged}$	0	✓	✓	✓
C1	false_positive_rate_difference	$FPR_{D=unprivileged} - FPR_{D=privileged}$	0	✓	✓	✓
C2	false_negative_rate_difference	$FNR_{D=unprivileged} - FNR_{D=privileged}$	0	✓	✓	✓
C3	false_omission_rate_difference	$FOR_{D=unprivileged} - FOR_{D=privileged}$	0	✓	✓	
C4	false_discovery_rate_difference	$FDR_{D=unprivileged} - FDR_{D=privileged}$	0	✓	✓	
C5	false_positive_rate_ratio	$FPR_{D=unprivileged} / FPR_{D=privileged}$	1	✓	✓	✓
C6	false_negative_rate_ratio	$FNR_{D=unprivileged} / FNR_{D=privileged}$	1	✓	✓	✓
C7	false_omission_rate_ratio	$FOR_{D=unprivileged} / FOR_{D=privileged}$	1	✓	✓	
C8	false_discovery_rate_ratio	$FDR_{D=unprivileged} / FDR_{D=privileged}$	1	✓	✓	
C9	average_odds_difference	$\frac{1}{2}$(*false_positive_rate_difference* + *true_positive_rate_difference*)	0	✓	✓	
C10	average_abs_odds_difference	$\frac{1}{2}$(|*false_positive_rate_difference*| + |*true_positive_rate_difference*|)	0	✓	✓	
C11	error_rate_difference	$ERR_{D=unprivileged} - ERR_{D=privileged}$	0	✓	✓	
C12	error_rate_ratio	$ERR_{D=unprivileged} / ERR_{D=privileged}$	1	✓	✓	
C13	selection_rate	$Pr(\hat{Y} = favorable)$	0	✓	✓	
C14	disparate_impact	$Pr(\hat{Y}=1\|D=unprivileged) / Pr(\hat{Y}=1\|D=privileged)$	1	✓	✓	✓
C15	statistical_parity_difference	$Pr(\hat{Y}=1\|D=unprivileged) - Pr(\hat{Y}=1\|D=privileged)$	0	✓	✓	✓
C16	generalized_entropy_index	$\frac{1}{n\alpha(\alpha-1)} \sum_{i=1}^{n} [(\frac{b_i}{\mu})^{\alpha} - 1]$ where $b_i = \hat{y}_i - y_i + 1$	0	✓		
C17	between_all_groups_generalized _entropy_index	generalized_entropy_index between all groups	0	✓		
C18	between_group_generalized _entropy_index	generalized_entropy_index between privileged and unprivileged groups	0	✓		
C19	theil_index	$\frac{1}{n} \sum_{i=1}^{n} \frac{b_i}{\mu} \ln \frac{b_i}{\mu}$	0	✓		
C20	coefficient_of_variation	$2 * \sqrt{generalized\_entropy\_index}$	0	✓		
C21	between_group_theil_index	theil_index between privileged and unprivileged groups	0	✓		
C22	between_group_coefficient _of_variation	coefficient_of_variation privileged and unprivileged groups	0	✓		
C23	between_all_groups_theil _index	theil_index between all groups	0	✓		
C24	between_all_groups_coefficient _of_variation	coefficient_of_variation between all groups	0	✓		
C25	differential_fairness_bias _amplification	Smoothed EDF between the classifier and the original dataset	0	✓		
Dataset Metrics						
D0	consistency	$1 - \frac{1}{n * n\_neighbors} \sum_{i-1}^{n} \|\hat{y}_i - \sum_{j \in N_{n_neighbors}(x_i)} \hat{j}_y\|$	1	✓		
D1	smoothed_empirical _differential_fairness	Smoothed EDF	0	✓		
D2	mean_difference	$Pr(\hat{Y}=1\|D=unprivileged) - Pr(\hat{Y}=1\|D=privileged)$	0	✓		
D3	disparate_impact	$Pr(\hat{Y}=1\|D=unprivileged) / Pr(\hat{Y}=1\|D=privileged)$	1	✓		

This paper is structured around these research questions.

RQ1: *Do current fairness metrics agree with each other?* Our experiments show that current fairness metrics often disagree, markedly.

RQ2: *Can we group (cluster) fairness metrics based on similarity?* We find sets of similar metrics using agglomerative clustering [Agg].

RQ3: *Are some fairness metrics more sensitive to change than others?* While most are sensitive, some are not.

RQ4: *Can we achieve fairness based on all the metrics at the same time?* It is challenging to do so since some of them are competing goals and some are contradictory based on definitions.

In terms of research contributions, this study is important since the art of software fairness testing is evolving rapidly. Studies like this one are essential to documenting what methods are "best" (as opposed to those that might distract from core issues). Accordingly:

- This paper proposes a novel metric assessment tactic that can clarify and simplify future research reports in this field (run metrics on real-world data; find clusters of correlated metrics; prune "insensitive clusters1"; only use one metric per surviving cluster).
- This paper tests that tactic in an *extensive case study* applying 30 fairness metrics and grouped them into clusters (RQ1 & RQ2). We say this study is extensive since it is far more detailed than prior work. All our empirical results were repeated 25 times. Our study explores multiple bias mitigation algorithms on seven datasets (than prior work [Gra19; Cel19; Cha21; Cha20b; Ces19] was tested on far fewer metrics and far fewer datasets).
- To the best of our knowledge, this study is the first one to perform such a *sensitivity meta-analysis* of fairness testing and to warn that some metrics are unresponsive to data changes (RQ3).
- This study also presents a *meta-analysis of metrics ability* to achieve fairness after application of bias mitigation technique (RQ4).

8.1.1 Preliminaries

Before beginning, we digress to make four points.

Firstly, mitigating the untoward effects of AI is a much broader problem than just exploring bias in algorithmic decision making (as done in this paper). The general problem of fairness is that influential groups in our society might mandate systems that (deliberately or unintentionally) disadvantage sub-groups within that society. An algorithm might satisfy all the metrics of Table 8.1 and still perpetuate social inequities. For example:

- Its license feeds might be so expensive that only a small monitory of organizations can boast they are "fair";
- The skills required to use a model's API might be so elaborate that only an elite group of programmers can use it even if the model is fair.

More generally, Gebru et al. [Geb; Buo18] argues that inequities arise from the core incentives that drive the organizations building an AI model, e.g., tools funded by the Defence Department have a tendency to support damage to property or life. She argues that "There needs to be regulation that specifically says that corporations need to show that their technologies are not harmful before deploying them". In terms of her work, this paper addresses the technical issue of how to measure "harm". As we show in Table 8.1 there are dozens of ways we might call software "biased" (and, hence, harmful). But we can also show that many of those measures are relatively uninformative. Hence, if some organization wishes to follow the recommendations of Gebru et al., then with the methods of this paper, they can make their case of "harmless" via a smaller and simpler report.

Secondly, Table 8.1 lists dozens of metrics currently seen in the SE fairness testing literature. This paper makes an *empirical argument* that this list is too long since many of these metrics offer

similar conclusions. One alternative to our *empirical argument* is an *analytical argument* that metric X (e.g.) is equivalent to metric Y. Later in this paper (see §8.5.1), we make the case that to reduce the space of metrics to be explored, that kind of analytical argument may actually be misleading.

Thirdly, to be clear, while we can reduce dozens of metrics down to ten, there will still be issues of how to trade-off within this reduced set. That said, we assert our work is valuable since debating the merits of, say, ten metrics is a far more straightforward task than trying to resolve all the conflicts between 30. Further, and more importantly, our methods could be used as a litmus test to prune away spurious new metrics that merely report old ideas but in a different way.

Fourthly, even after our mitigation algorithms, some fairness metrics still can contradict each other regarding the presence of bias. Hence, in §8.5.3, we offer an extensive discussion on what to do in that situation.

8.2 Background

8.2.1 The Problem of Algorithmic Fairness

As software developers, we cannot turn a blind eye to the detrimental social effects of our software. While no single paper can hope to fix all social inequities, this paper shows how to reduce the complexity involved in assessing one particular kind of unfairness (algorithmic decision making bias). There is much evidence of machine learning (ML) software showing biased behavior. For example, language processing tools are more accurate on English written by Anglo-Saxons than written by people of other races [Blo17]. An Amazon hiring tool was found to be biased against women [Amad]. YouTube makes more mistakes while generating closed captions for videos with female voices than males [Tat17; Koe20]. A popular risk-score predicting algorithm was found to be heavily biased against African Americans showing a higher error rate while predicting future criminals [Pro]. Gender bias is also prevalent in Google [Cal17a] and Bing [Joh20] translators.

Due to so many undesirable events, academic researchers and big industries have started giving immense importance to ML software fairness. Microsoft has launched ethical principles of AI where “fairness” has been given the topmost priority [Mic]. IBM has built a toolkit called AI Fairness 360 [Aifa] containing the most noted works in the fairness domain. In recent years, the software engineering research community has also started exploring this topic actively. ICSE’18 held a special workshop for “software fairness” [Faic]. ASE’19 held another workshop called EXPLAIN, where fairness and explainability of ML models were discussed [Exp]. Johnson et al. have created a public GitHub repository for data scientists to evaluate ML models based on quality and fairness metrics simultaneously [Joh20].

As to technology developed to detect and fix these issues of fairness, we can see three groups: *fairness testing, model-based mitigation, and fairness metrics,* .

Fairness Testing: The idea here is to generating discriminatory test cases and finding whether the model shows discrimination or not. The first work on this was THEMIS, done by Galhotra et al. [Gal17]. THEMIS generates test cases by randomly perturbing attributes. AEQUITAS [Ude18]

improves the way of test case generation to become more efficient. Aggarwal et al. combined local explanation and symbolic execution to generate a better black-box testing strategy [Agg19].

Model Bias Mitigation: There are three techniques used to remove bias from model behavior. The first one is "pre-processing" where before model training, bias is removed from training data. Some popular prior work includes optimized pre-processing [Cal17b], Fair-SMOTE [Cha21] and reweighing [Kam12a]. The second one is "in- processing" where after model training, the trained model is optimized for fairness. Popular prior work includes prejudice remover regularizer [Kam12c] and meta fair classifier [Cel20]. The last one is "post-processing" where while making predictions, model output is changed to remove discrimination. Some noted works include reject option classification [Kam18] and calibration [Ple17]. There is some work that combines two or more of these techniques, such as Fairway [Cha20b], a combination of "pre-processing" and "in-processing".

While the fairness testing and model bias mitigation are important areas, we note that *before* we can declare success in those two areas, we *first* need some way to measure that success.

Accordingly, this paper focuses on the third area called:

Fairness Metrics: Early work in this area was done by Verma et al. [Ver18a] who divided 20 fairness metrics into five groups based on the theoretical definitions. Hinnefeld et al. made a comparative analysis of four fairness metrics [Hin18]. Wang et al. did a user study to find a relation between fairness metrics and human judgments [Wan19]. There are also some papers coming from industry on the topic. LinkedIn has created a toolkit called LiFT for scalable computation of fairness metrics as part of large ML systems [Vas20]. Recently, Amazon internally published an empirical study based on 18 fairness metrics [al.20].

While all that research is certainly insightful, in some sense that work has been too successful. As mentioned in the introduction, the above work has now generated a plethora of metrics. Hence, for the rest of this paper, we check if we can simplify the current space of metrics.

8.2.2 Metrics Used in this Study

In our work, we collected all the metric definitions from the IBM AI Fairness 360 GitHub repository. Table 8.1 lists the metrics studied in this paper. The *Fairkit* and *Fairlearn* columns in Table 8.1 show the metrics that are common among the IBM AIF360 metrics and metrics from Fairkit [Joh20] (16 out of 16 available metrics) and Fairlearn [Faib] (7 out of 16 metrics) toolkit.

Before explaining fairness metrics, we need to understand some terminology. Table 2.1 contains ten binary classification datasets. The binary outcomes are *favorable* if it gives an advantage to the receiver (i.e., being hired for a job, getting credit card approval). Each of these datasets has at least one *protected attribute* that divides the population into two groups (*privileged* & *unprivileged*) that have differences in terms of benefits received. "sex", "race", "age" are examples of protected attributes. The goal of group fairness is, based on the protected attribute, privileged and unprivileged groups will be treated similarly. While individual fairness tries to provide similar outcomes to similar individuals.

A *fairness metric* is a quantification of unwanted bias in training data or models. Table 8.1 shows

a sample of such metrics. When selecting these particular metrics, we skipped over:

- Metrics for which we could not access precise definitions and implementations in IBM AIF360 toolkit [Bel18];
- Metrics for which we could not find publications to use as baselines in this paper.

These two selection rules resulted in the 30 metrics of Table 8.1, which divide as follows:

Classification Metrics: These measure fairness based on classification results and are labeled in Table 8.1 using a *Metric Id* beginning with *C*. Two inputs are needed to measure this: the first one is original dataset with true labels and the second one is predicted dataset. In case of binary classification, classification metrics can be calculated from confusion matrix. Table 8.2 shows a combined confusion matrix where every cell is divided based on the protected attribute.

Dataset Metrics: While classification metrics relate to predictions made from models, *dataset metrics* discuss learner-independent properties of the data. These are labeled in Table 8.1 using a *Metric Id* beginning with *D*. Only one input is needed to compute this: original dataset or transformed (by some bias mitigation algorithm) dataset. It can be applied for both *group and individual fairness*.

Distortion Metrics: For completeness, we note that AIF360 includes a third set of metrics called *distortion metrics*. While these metrics are not seen extensively in the current literature, they would be a worthy target for future research.

In Table 8.1, each metric has an *ideal value* representing the best-case scenario. This means at ideal value according to the metric privileged and unprivileged groups are treated equally. For most of the metrics, the ideal value is zero, while in some cases where the metric is a ratio, the ideal value is one. If the ideal value for a metric is zero, a positive value denotes an advantage for the unprivileged group, while a negative value denotes an advantage for the privileged group. On the other hand, if the ideal value for a metric is one, a value < one denotes an advantage for the privileged group and > one denotes an advantage for the unprivileged group.

To use these metrics, some threshold must be applied to report “fair” or “unfair”;

- For metrics with ideal value 0: the IBM AIF360 toolkit [Bel18] uses the following definition of “fair”: ranges between -0.1 to 0.1 as “fair” (so “unfair” means values outside that range).
- For metrics with ideal value 1: the IBM AIF360 toolkit [Bel18] uses the following definition of “fair”: ranges between 0.8 to 1.2 as “fair” (so “unfair” means values outside that range).

8.3 Methodology

8.3.1 Models

This paper analyzes the 30 fairness metrics in Table 8.1 using the seven datasets described in Table 2.1. In that work, we use one baseline model and two models tuned by pre-processing and in-processing algorithms to compare with:

Table 8.2 Mathematical definition of various confusion matrix based rates. These are used to calculate fairness metrics defined in Table 8.1.

	Actual Positive	Actual Negative
Predicted Positive	TP PPV = TP/(TP+FP) TPR = TP/(TP+FN)	FP FDR = FP/(TP+FP) FPR = FP/(FP+TN)
Predicted Negative	FN FOR = FN/(TN+FN) FNR = FN/(TP+FN)	TN NPV = TN/(TN+FN) TNR=TN/(TN+FP)

.

- **Baseline:** We used a logistic regression model for creating baseline results. Logistic regression is widely used in the fairness domain as baseline model [Cha20b; Kam12c; Cal17c; Cha19; Cha20c]. We used scikit-learn implementation with 'l2' regularization (which helps to prevent over-fitting), 'lbfgs' solver (which is a quasi-Newton optimization algorithm), and maximum iteration of 1000.

- **Reweighing:** A widely used [Ces19; Agr20; Bel19; Sat19; Jon20] pre-processing method proposed by Kamiran et al. [Kam12a]. Here, before model training, examples in each group, and label are given different weights to ensure fairness.

- **Meta Fair Classifier:** An in-processing method proposed by Celis et al. [Cel20], which is a widely used meta algorithm [Cel19; Gra19; Ber19; Pad20]. The optimization algorithm is developed to improve 11 fairness metrics with minimal loss in accuracy.

The last two bias mitigation algorithm implementations are taken from IBM AIF360 [**ibm**].

8.3.2 Agglomerative Clustering

Our metrics selection strategy requires a clustering algorithm. Two classes of such clustering algorithms are (a) partitioning clustering and (b) hierarchical clustering. Here we are grouping fairness metrics based on similarity, not on distance, and we have no prior idea about the number of clusters. Thus, in this case, the ideal choice is *hierarchical clustering*. Agglomerative clustering [Agg] is a hierarchical bottom-up clustering approach that is widely used in the ML community [Dic01; Zha16a; Rod08; Shi15; Do08; D'h05; Lin16; Fok11; Bav14]. In this approach, the closest pairs of items are grouped together. The closest of these groups are then grouped into a higher-level group. This repeats until everything falls into one group. We used the average pairwise dissimilarity between objects in two different clusters as a linkage method between groups. This process creates a dendrogram, a hierarchical structure of the groups/clusters obtained by between-cluster distance or dissimilarity. From this tree of groupings, we use the within-cluster similarity from the dendrogram, look for the largest distance that we can travel vertically without crossing any horizontal line [Tho53; Sta; Dan17], and extract the clusters at the largest change in dissimilarity (which is similar to SSE - Sum of Squared Error).

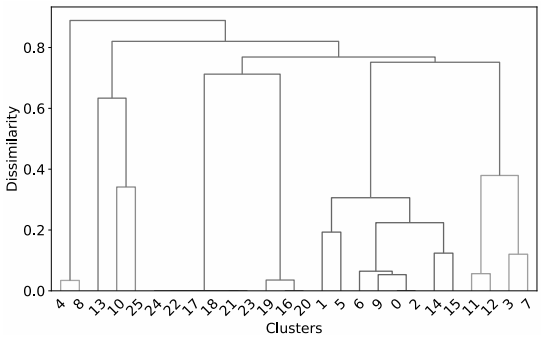

Figure 8.1 Agglomerative clustering of classification metrics (using Spearman rank correlation). Here x-axis shows the classification metric Ids from Table 8.1 and y-axis shows the dissimilarity measure between clusters.

8.3.3 Spearman Rank Correlation

Figure 8.1 shows the dendrogram created for the classification metrics using the above described method. Table 8.3 shows that we get seven clusters from 26 classification metrics. Following a similar process for dataset metrics we get three clusters as shown in Table 8.4.

To build these clusters and dendrograms, we measure the similarity of two metrics. In this paper, by "similarity" we mean, they are measuring the similar bias in the models/dataset. Such similar metrics will show a similar pattern of changes in bias when models are built using different parts of the data or different bias removal algorithms are used. To compute this similarity, we sample from our model training procedure (see §8.3.4.2) that computes our metrics 25 times, each time using different train/validation/test samples of the data. Next, for each dataset, for those 25 numbers, we use correlation to assess similarity.

Two widely used definitions of correlation [Dic01; Shi15; Ree15; Do08; Hos15; D'h05; Zha16a; Che17] are the (a) Pearson correlation (which evaluates the linear relationship between two continuous variables) and the (b) Spearman rank correlation (which is a non-parametric measure of rank correlation that evaluates the monotonic relationship between two continuous or ordinal variables). We choose Spearman rank correlation, as it measures the monotonic relationship between two variables and is less affected by outliers.

8.3.4 Experimental Setup

We summarize our experimental setup as follows.

8.3.4.1 Data Pre-processing:

Three different pre-processing steps are performed before using the data [Lah19; Fri19; Sch19] for model building. At first, each categorical value in the dataset is converted either using a label encoder or by one hot encoder, as most ML algorithms can't handle categorical values directly. Then the protected attributes are changed into ones and zeros from their original values. Here we denote

the privileged attribute as one and unprivileged as zero. Finally, we use min-max normalization in the datasets to normalize the data before building the models.

8.3.4.2 Model Training:

We used five fold cross-validation repeated five times with random seeds build training/ test sets (as recommended by [Ver18b; Sch19; Val19; Kam12b]). This step is to divide the data into multiple subsets of data with various degrees of bias. We train three models in each iteration (a) *baseline model:* here we use the training data to build a logistic regression model; (b) *Reweighing model:* here we first train the reweighing method, then use the learned model to transform the training data to achieve group fairness. Using the transformed data, we train a logistic regression from scikit-learn with 'l2' regularization, 'lbfgs' solver and maximum iteration of 1000; and (c) *Meta Fair Classifier model:* here to train the meta fair classifier model, we use the training data to build multiple meta fair classifier model with different values of τ (a hyperparameter for fairness penalty in the model) and measure the bias in the model using the validation set. Then to build the final model, we select the τ for which the model had the lowest bias in the validation set and build the final meta fair classifier model.

8.3.4.3 Fairness Metric Calculation:

We collect the performance of each model based on 26 classification and four dataset metrics for each iteration of the cross-validation. So for each iteration, we use the test data for prediction and then use the predicted values along with the ground truth to calculate the 26 classification metrics. Similarly, we collect the four dataset metrics on the baseline and reweighing method. Meta fair classifier is not applicable in the case of dataset metrics.

8.3.4.4 Measure for Fairness:

Data Pre-processing, Model Training, and Fairness Metric Calculation steps are performed for each of seven datasets with five fold five repeat cross-validation. Then to measure if the model built on a dataset is fair or unfair according to a metric, we selected a threshold for each of the metrics. As mentioned in §8.2.2, that threshold is the *fair range*. If a metric value falls in that range, we say it "fair" otherwise "unfair".

8.3.4.5 Building Clusters:

One of the main goals of this study is to group a set of metrics together that perform similarly and measure similar kinds of bias. We use 26 classification metrics calculated on seven datasets with three different methods to calculate metric to metric correlation based on the Spearman rank correlation coefficient. We do the same for the four dataset metrics as well. This provides us two correlation matrices: one 26x26 and one 4x4. After that, to build the clusters using the agglomerative clustering, we convert the similarity matrix into a dissimilarity matrix [Hos15; D'h05] using equation 8.1. We

use this dissimilarity matrix to create the clusters. The agglomerative clustering process creates a dendrogram as shown in Figure 8.1. Now to select the number of clusters, we cut the dendrogram at a height, where the clusters will remain unchanged with the most increase/decrease of the cutting threshold. For classification metrics, we cut the dendrogram (Figure 8.1) at 0.57 as the clusters will remain unchanged between the cutoff value 0.49 and 0.64. Finally, we get the clusters containing classification metrics measuring similar kinds of bias. We perform the same process for dataset metrics and cut the dendrogram at a height of 0.4.

$$d(x,y) = 1 - |sim(x,y)| \tag{8.1}$$

8.3.4.6 Calculating Sensitivity:

Research question four asks about the consistency of the metric values for three cases: (a) raw data, b) after applying Reweighing (RW), (c) after applying Meta Fair Classifier (MFC). As we are using five cross fold five repeats for all the datasets, we get 25 results for each dataset and report for all seven datasets:

- the median value: the 50th percentile (or Q_2);
- the IQR: the (75-25)th percentile (or $Q_3 - Q_1$)

8.4 Results

Our results are organized based on four research questions.

RQ1: Do current fairness metrics agree with each other?

At first, we need to verify our motivation. In real life, do the fairness metrics contradict? Table 8.3 contains results for 26 classification metrics; Table 8.4 contains results for four dataset metrics. The learner here is logistic regression. The last row contains the percentage of metrics marking the specific dataset as unfair in both tables. If we combine last rows of Table 8.3 & 8.4 and sort them in ascending order, we get the following list:

$$\{\ 23, 34, 50, 50, 50, 54, 58, 65, 75, 75, 75, 75, 77, 100\ \}\%$$

The median value here is 62%; i.e., nearly half the time the metrics make *different* conclusions about the *same* data. This means that researchers and practitioners will be spending much effort trying to understand their systems using disagreeing oracles (a result that motivates this entire paper).

RQ2: Can we group (cluster) fairness metrics based on similarity?

Table 8.3 shows that 26 classification metrics can be divided into seven clusters. Table 8.4 shows that four dataset metrics can be divided into three clusters. More importantly, we note that:

Table 8.3 Cluster based results for 26 classification metrics on seven datasets. For a metric with an ideal value of zero, anything below -0.1 and above 0.1 is "unfair". For a metric with an ideal value of one, anything <0.8 or >1.2 is "unfair".

Cluster Id	MID	Metrics	Datasets							Metric Type
			Adult	Compas	German	Health	Bank	Student	Titanic	
0	C3	false_omission_rate_difference	Unfair	Fair	Fair	Unfair	Fair	Fair	Unfair	Mis-classification
0	C7	false_omission_rate_ratio	Unfair	Fair	Fair	Unfair	Fair	Unfair	Unfair	
0	C11	error_rate_difference	Unfair	Fair	Fair	Unfair	Fair	Fair	Fair	
0	C12	error_rate_ratio	Unfair	Fair	Fair	Unfair	Fair	Fair	Fair	
		Percentage of agreement	**100%**	**100%**	**100%**	**100%**	**100%**	**75%**	**50%**	
1	C10	average_abs_odds_difference	Unfair	Unfair	Unfair	Unfair	Unfair	Fair	Unfair	Differential Fairness
1	C25	differential_fairness_bias_amplification	Unfair	Unfair	Unfair	Unfair	Unfair	Fair	Unfair	
		Percentage of agreement	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	
2	C16	generalized_entropy_index	Fair	Unfair	Fair	Fair	Fair	Fair	Unfair	Individual Fairness
2	C19	theil_index	Unfair	Unfair	Fair	Unfair	Unfair	Fair	Unfair	
2	C20	coefficient_of_variation	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	
		Percentage of agreement	**67%**	**100%**	**67%**	**67%**	**67%**	**67%**	**100%**	
3	C4	false_discovery_rate_difference	Fair	Fair	Fair	Fair	Fair	Fair	Unfair	Mis-classification
3	C8	false_discovery_rate_ratio	Fair	Fair	Fair	Fair	Fair	Unfair	Unfair	
		Percentage of agreement	**100%**	**100%**	**100%**	**65%**	**100%**	**50%**	**100%**	
4	C0	true_positive_rate_difference	Unfair	Unfair	Fair	Unfair	Unfair	Fair	Unfair	Confusion Matrix Based Group Fairness
4	C1	false_positive_rate_difference	Fair	Unfair	Unfair	Unfair	Unfair	Fair	Unfair	
4	C2	false_negative_rate_difference	Unfair	Unfair	Unfair	Unfair	Unfair	Fair	Unfair	
4	C5	false_positive_rate_ratio	Fair	Unfair	Unfair	Unfair	Unfair	Fair	Unfair	
4	C6	false_negative_rate_ratio	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	
4	C9	average_odds_difference	Unfair	Unfair	Unfair	Unfair	Unfair	Fair	Unfair	
4	C14	disparate_impact	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	
4	C15	statistical_parity_difference	Unfair	Unfair	Unfair	Unfair	Unfair	Fair	Unfair	
		Percentage of agreement	**75%**	**100%**	**88%**	**100%**	**100%**	**75%**	**100%**	
5	C17	between_all_groups_generalized_entropy_index	Fair	Fair	Fair	Fair	Fair	Fair	Fair	Between Group Individual Fairness
5	C18	between_group_generalized_entropy_index	Fair	Fair	Fair	Fair	Fair	Fair	Fair	
5	C21	between_group_theil_index	Fair	Fair	Fair	Fair	Fair	Fair	Fair	
5	C22	between_group_coefficient_of_variation	Fair	Fair	Fair	Fair	Fair	Fair	Unfair	
5	C23	between_all_groups_theil_index	Fair	Fair	Fair	Fair	Fair	Fair	Fair	
5	C24	between_all_groups_coefficient_of_variation	Fair	Fair	Fair	Fair	Fair	Fair	Unfair	
		Percentage of agreement	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**67%**	
6	C13	selection_rate	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	Intermediate Metric
		Percentage of agreement	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	
Percentage of metrics marking dataset as unfair			**58%**	**54%**	**34%**	**65%**	**50%**	**23%**	**77%**	

Table 8.4 Cluster based results for four dataset metrics on seven datasets. For a metric with ideal value of zero, anything below -0.1 and above 0.1 is "unfair". For a metric with ideal value of one, anything <0.8 or >1.2 is "unfair".

Cluster Id	MID	Metrics	Datasets							Metric Type
			Adult	Compas	German	Health	Bank	Student	Titanic	
0	D0	consistency	Fair	Unfair	Fair	Unfair	Fair	Unfair	Fair	Individual Fairness
1	D1	smoothed_empirical_differential_fairness	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	Unfair	Differential Fairness
2	D2	mean_difference	Unfair	Unfair	Unfair	Fair	Unfair	Fair	Unfair	Confusion Matrix Based Group Fairness
2	D3	disparate_impact	Unfair	Unfair	Unfair	Fair	Unfair	Fair	Unfair	
Percentage of metrics marking dataset as unfair			**75%**	**100%**	**75%**	**50%**	**75%**	**50%**	**75%**	

- RQ1 reported intra-project disagreement on “fair“-vs-“unfair”;
- We note that there is much intra-cluster agreement for each data set in Table 8.3 and Table 8.4.

As evidence, we note that the majority fairness decision is always the same within the clusters for each dataset. In Table 8.3, the row *Percentage of agreement* comments on the uniformity of decisions within each cluster (for each dataset). Note that uniformity is very high (often 100%). That means metrics inside each cluster agree with each other for every dataset. Among the seven clusters, we see six clusters (except cluster two) show 100% agreement considering median value across seven datasets. For example, in case of cluster zero, percentage of agreement is 100% for five datasets; 75% for one; 50% for one. Majority is 100%. That is true for clusters 1,3,4,5,6 & 7. We see a similar agreement pattern inside clusters in Table 8.4 also.

For reference purposes, the last column of Table 8.3 and Table 8.4 offers names for those clusters:

- **Misclassification (cluster 0, 3)**: these metrics try to measure the difference or ratio of misclassification errors between groups;
- **Differential fairness (cluster 1)**: these metrics try to measure if probabilities of the outcomes are similar regardless of the combination of protected attributes [Fou19];
- **Individual Fairness (cluster 2)**: It measures if two similar individuals with respect to the classification task receive the same outcome or not;
- **Confusion matrix based group fairness (cluster 4)**: these metrics measure difference or ratio between groups based on confusion matrix;
- **Between group individual fairness (cluster 5)**: measures the difference or ratio of individual fairness between groups;
- **Intermediate metrics (cluster 6)**: these are intermediate metrics.

From a practitioner viewpoint, this clustering is useful because:

- The clustering reduces the confusion of having too many metrics and not knowing their similarity.
- As the metrics inside the same cluster measure same kind of bias and behave in the same manner; we can choose just one metric from each cluster. Thus we measure a few metrics but can cover a much more comprehensive range of fairness notions.
- If we see agreement among all the metrics inside a cluster for a particular dataset, then one metric can be chosen as representative of the whole cluster.
- In case of intra-cluster conflicts, choosing only one metric can be risky. In these cases, practitioners need to do a proper risk assessment before selecting metrics. That said, if there is intra-cluster conflict among metrics, we can choose one from the ‘fair’ group and one from the ‘unfair’ group to mitigate that risk.

As part of this study, we further analyzed each cluster mathematically to verify if our cluster of metrics and their mathematical definitions coincide. A detailed analysis of these clusters and their mathematical analysis has been discussed in §8.5.1.

RQ3: Are some fairness metrics more sensitive to change than others?

An ideal metric is responsive to the dataset it examines. An "insensitive" metric is one that delivers the same conclusions, no matter what data is being examined. An "insensitive" cluster is one containing mostly insensitive metrics. Such insensitive clusters could be ignored since they are not informative.

We measure sensitivity by looking at the variability of our metrics scores using the intra-quartile range (IQR=$Q_3 - Q1$). For each data set, we found the IQR across all clusters. Next, we highlight the sensitive results; i.e. those with an IQR greater that d*standard deviation. The remaining, unhighlighted results are the insensitive metrics.

As to what value of d to use in this analysis, we take the advice of a widely cited paper by Sawilowsky [**sawilowsky2009new**] (this 2009 paper has 1100 citations). That paper asserts that "small" and "medium" effects can be measured using $d = 0.2$ and $d = 0.5$ (respectively). We will analyze this data by splitting the difference looking for differences larger than $d = (0.5+0.2)/2 = 0.35$.

Turning now to Table 8.5 and Table 8.6 we see that most clusters have highlight IQR results.

Table 8.5 This table shows sensitivity of the classification metrics on the three different models used in this study (a) Baseline; (b) Reweighing(RW); and (c) Meta Fair Classifier(MFC). The table shows the median and IQR values of three datasets. Here the cells in IQR columns are marked with "red" that change by more than a small amount (35th percentile of the standard deviation of the IQR values). The insensitive metrics are those that usually have white IQR values.

MID	Compas						Health						German					
	Baseline		RW		MFC		Baseline		RW		MFC		Baseline		RW		MFC	
	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR
C3	-0.067	0.079	-0.113	0.015	-0.061	0.035	-0.14	0.032	-0.211	0.068	-0.151	0.137	0	0.5	-0.5	0.667	0	0.592
C7	0.817	0.211	0.691	0.029	0.824	0.099	0.357	0.091	0.158	0.333	0.408	0.356	2.3	0.72	0	0.5	1	0.53
C11	-0.033	0.043	-0.016	0.039	-0.042	0.026	-0.108	0.01	-0.133	0.013	-0.098	0.111	0.059	0.074	0.059	0.114	0.049	0.062
C12	0.912	0.112	0.958	0.112	0.887	0.071	0.508	0.174	0.339	0.026	0.494	0.51	1.18	0.274	1.18	0.418	1.166	0.215
C10	0.252	0.058	0.029	0.02	0.181	0.035	0.162	0.103	0.106	0.087	0.161	0.064	0.221	0.167	0.043	0.048	0.031	0.121
C25	0.531	0.354	-0.22	0.141	0.359	0.148	0.193	0.249	-0.094	0.422	0.113	0.428	2.399	3.29	1.162	0.49	1.578	2.087
C16	0.193	0.001	0.189	0.012	0.192	0.007	0.091	0.004	0.087	0.017	0.091	0.025	0.076	0.011	0.071	0.011	0.066	0.011
C19	0.268	0.003	0.263	0.017	0.269	0.009	0.132	0.021	0.14	0.051	0.139	0.033	0.083	0.02	0.073	0.017	0.064	0.02
C20	0.878	0.003	0.87	0.027	0.876	0.016	0.602	0.015	0.589	0.058	0.602	0.082	0.553	0.041	0.532	0.041	0.513	0.043
C4	0.04	0.034	0.138	0.051	0.037	0.058	-0.091	0.133	-0.009	0.202	-0.016	0.152	0.059	0.135	0.059	0.114	0.045	0.064
C8	1.109	0.089	1.376	0.143	1.098	0.156	0	0.944	0.964	1.571	0.925	1.292	2.6	0.542	1.18	0.459	1.156	0.233
C0	-0.273	0.087	-0.004	0.052	-0.212	0.048	-0.106	0.139	0.13	0.227	-0.117	0.405	-0.077	0.092	0	0.038	-0.017	0.062
C1	-0.186	0.052	-0.014	0.02	-0.17	0.031	-0.214	0.053	-0.13	0.174	-0.109	0.114	-0.3	0.233	0	0.029	-0.053	0.176
C2	0.273	0.087	0.004	0.052	0.212	0.048	0.106	0.139	-0.13	0.227	0.117	0.405	0.077	0.092	0	0.038	0.017	0.062
C5	0.408	0.037	0.956	0.069	0.471	0.068	0	0.219	0.25	0.654	0.191	0.347	0.7	0.228	1	0.029	0.947	0.176
C6	1.71	0.242	1.009	0.124	1.514	0.133	1.467	0.5	0.429	1.222	1.453	2.056	3.4	0.56	0	5.459	11.532	3.4
C9	-0.252	0.058	-0.018	0.059	-0.181	0.035	-0.162	0.103	-0.06	0.158	-0.139	0.165	-0.221	0.167	0	0.043	-0.031	0.121
C14	0.432	0.073	0.89	0.133	0.541	0.062	0.314	0.14	0.435	0.322	0.387	0.274	0.836	0.12	1	0.045	0.971	0.105
C15	-0.264	0.05	-0.049	0.059	-0.205	0.03	-0.356	0.079	-0.289	0.179	-0.298	0.18	-0.164	0.122	0	0.043	-0.029	0.104
C17	0.002	0.002	0.001	0.002	0.001	0.001	0.001	0.003	0.002	0.001	0.001	0.001	0.001	0.002	0.002	0.001	0.001	0.001
C18	0.002	0.002	0.001	0.002	0.001	0.001	0.001	0.001	0.002	0.001	0.001	0.001	0.001	0.002	0.002	0.001	0.001	0.001
C21	0.002	0.002	0.001	0.003	0.001	0.001	0.003	0.001	0.002	0.001	0.001	0.001	0.003	0.002	0.003	0.001	0.002	0.001
C22	0.088	0.049	0.052	0.006	0.057	0.019	0.036	0.037	0.017	0.054	0.045	0.03	0.029	0.063	0.03	0.05	0.036	0.038
C23	0.002	0.002	0.001	0	0.001	0.001	0.003	0.003	0.006	0.001	0.001	0.001	0.001	0.002	0.001	0.001	0.004	0.001
C24	0.088	0.049	0.052	0.006	0.057	0.019	0.036	0.037	0.017	0.054	0.045	0.03	0.029	0.063	0.03	0.05	0.036	0.038
C13	0.405	0.025	0.436	0.016	0.41	0.017	0.407	0.05	0.39	0.133	0.411	0.056	0.945	0.015	0.975	0.03	0.99	0.047

Table 8.6 This table is similar to Table 8.5, showing the sensitivity of the dataset metrics on (a) Baseline; (b) Reweighing (RW).

MID	Compas						Health						German					
	Baseline		RW		MFC		Baseline		RW		MFC		Baseline		RW		MFC	
	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR	Med	IQR
D1	0.568	0.021	0.568	0.021	-	-	0.804	0.02	0.804	0.02	-	-	0.632	0.008	0.632	0.008	-	-
D2	0.252	0.043	0	0	-	-	0.868	0.322	0.001	0	-	-	0.298	0.105	0.002	0	-	-
D3	-0.105	0.016	0	0	-	-	-0.313	0.067	0	0	-	-	-0.097	0.033	0	0	-	-
D4	0.777	0.033	1	0	-	-	0.411	0.137	1	0	-	-	0.865	0.043	1	0	-	-

However, in Table 8.5, we see the clusters formed by metrics C16, C18, C20 (individual fairness) and C17, C18, C21, C22, C23, C24 (between group individual fairness) are insensitive. This, in turn, means that we should *not* criticize a fairness analysis that ignores these metrics.

RQ4: Can we achieve fairness based on all the metrics at the same time?

Different fairness metrics measure different kinds of bias. If any of the metrics complain about the fairness of the test results, then we can not trust the model blindly, and it should go through further scrutiny and improvement. Bias mitigation algorithms try to make unfair models fairer. Here we are verifying even after applying bias mitigation algorithms; can we achieve fairness based on all the metrics or not? We have chosen two highly cited algorithms from IBM AIF360: Reweighing (RW) by Kamiran et al. [Kam12a] and Meta Fair Classifier (MFR) by Celis et al [Cel20].

Table 8.7 shows those results collected for seven datasets after using RW and MFC algorithms. For every dataset (row-wise), we show the number of metrics changed towards or away from its ideal value. In that table:

- FU denotes the metrics that changed towards ideal value;
- UF denotes the metrics that moved away from the ideal value,
- NC means the metrics which did not change.

Table 8.7 This table shows the number of classification metrics that move towards or away from the ideal value when either Reweighing or Meta Fair Classifier is used to remove bias in the models. Here "UF" shows the number of metrics that moved towards the ideal metric value, while "FU" shows the opposite. Finally "NC" shows the number of metrics that did not change at all.

Dataset	Reweighing (RW)			Meta Fair Classifier (MFC)		
	UF	FU	NC	UF	FU	NC
Adult	13	13	0	11	15	0
Compas	15	7	4	16	6	4
Health	17	5	4	17	7	1
German	19	6	1	19	7	0
bank	16	6	4	15	7	4
Titanic	11	15	0	17	9	0
Student	15	7	4	12	10	4

Note that majority of the metrics move towards "fair", but there are some metrics that move towards "unfair". For Reweighing, some metrics show "no change", but we have verified they always remain in the *fair range*.

The main takeaway here is no longer necessary (or even possible) to satisfy all these fairness metrics. While our analysis can reduce dozens of metrics down to ten, there will still be issues of how to trade-off within this reduced set. Even after applying bias mitigation approaches, some metrics still conflict with others. This finding is similar to the claim made by others:

- Berk et al. [Ber17b] offer an "Impossibility Theorem" that says there is no way to satisfy all kinds of fairness together.
- As Yuriy Brun said at his keynote at ICSSP'2020 "*we need to work the system in a biased way sometimes*" [Bru20].

8.5 Discussion

We have described all of our results. Here we are summarizing the results in a comprehensible way to reach a stable conclusion. The main idea of this work is to reduce the complexity of measuring fairness. That said, it is imperative we narrate our conclusions in a very easy way. We discuss here three major concerns that arise from §8.4 and try to simplify fairness measurement to our best.

8.5.1 Why Not Group Metrics via their Analytical Structure?

This paper has offered an empirical analysis that many of the metrics in Table 8.3 are synonymous since, when clustered, they fell together into just a few similar groups. In this section, we check if the same conclusions can be achieved from a more analytical analysis that looked at the structure of the equations for the fairness metrics.

Sometimes, a group generated by formula's analytical structure is similar to the clusters we generated above. For example:

- In cluster three (from Table 8.3), all metrics are based on *FDR*, which suggests that both from an empirical and analytical point of view, they should be similar.
- Also, In cluster zero, we see that all those metrics are based on *FOR* and error rate. Intuitively, this seems sensible since here metrics try to measure amount of misclassification.

That said, as shown by the following three examples, there are many examples where an equation's analytical structure does *not* predict for its empirical cluster.

- ***EXAMPLE #1:*** If we look at cluster five, all six metrics inside this cluster are related to "between group individual fairness". This metric is based on the same benefit function:

$$y = \hat{y} - y + 1 \tag{8.2}$$

(For more details on that. see Table 8.1 metric id C16). We note that cluster two is also based on Equation 8.2, but the metrics inside this cluster represent individual fairness for each group separately. That means

> *Although all metrics inside cluster two and cluster five are based on the same benefit function, they measure different definitions of fairness.*

That is, a formal analysis of the analysis might combine these clusters, whereas a data-oriented empirical analysis would argue for their separation.

- ***EXAMPLE #2:*** In cluster four from Table 8.1, the metrics C0, C1, C2, C5, C6 and C9 depend on *TPR*, *FPR* and *FNR*. Recall that *FPR* and *FNR* report type one and type two errors (misclassification on fairness); Now *TPR* can be expressed as *1 - FPR*, which means the change in *TPR* will mirror changes in *FPR*. In contrast, in this cluster, the other two metrics C14 and C15 are based on selection rate (ratio of number of predicted positive and number of instances). Although there is not much similarity in the formula between these two and other metrics in this cluster, we can see they perform similarly when measuring fairness. That is:

  > *An analytical analysis does not always reflect the measurement of fairness in the real world scenario.*

  Verma et al. [Ver18] in their paper notice a similar phenomenon where they observe that: *Equal Predictive parity (a measure they explore) should also have equal* FDR *... but when measured from an empirical point of view, they showed they are not the same.*

- ***EXAMPLE #3:*** In cluster one, metrics C10 and C25 have very different mathematical formulas. C10 is based on *FPR* while C25 is based on smoothed EDF– the Empirical differential fairness. *EDF* is calculated based on Dirichlet smoothed base rates for each intersecting group in the dataset, which is based on count of predicted positive. Here as well, we see that

  > *Two formulas with a different analytical structure can have a similar performance w.r.t. fairness.*

To summarize the above, we quote Alfred Korzybski, who warned:

> *A map is not the territory.*

While the analytical structure of the formula offers intuitions about the nature of fairness, those intuitions had better be checked via empirical analysis.

8.5.2 Is our Empirical Analysis Useful?

We have established the requirement of empirical analysis and we have also done that analysis. We need to find out whether this analysis would be helpful in real-life applications or not. Here we describe various scenarios of fairness contradiction and how our study helps to remove that.

Imagine a college admission decision scenario, where the system might be seen as biased against group B if applicants from group A are accepted more than group B. Here group A and group B are divided based on different values of a protected attribute. The college applies a bias mitigation approach to solve this problem using a group fairness metric by changing group A's or B's scoring threshold. Now, if a member of group A is rejected, while a member of group B has been accepted with an equal or lower score, then the system might be seen as biased against that individual. The main takeaway from this story is that there is a conflict between "individual fairness" and "group fairness" [Bin19].

The concept of fairness is very much application-specific and choosing the appropriate metric is the sole responsibility of the policymaker. An ideal scenario will be building a machine learning model which does not show any kind of bias. However, that is too good to be true. Brun et al. found out that if a model is adjusted to be fair based on one protected attribute (e.g., sex), in some cases model becomes more biased based on another protected attribute (e.g., race) [Faic]. Kleinberg and other researchers argue that different notions of fairness are incompatible with each other and hence it is impossible to satisfy all kinds of fairness simultaneously [Kle16]. Here one thing to remember while doing prediction is that fairness is not the only concern. Prediction performance is the most important goal. Berk et al. found out that accuracy and fairness are competing goals [Ber17c]. This trade-off makes the job even more complicated since damaging model performance while making it fair may be unacceptable.

As researchers, we know that satisfying all kinds of fairness together is not possible. A policymaker has to choose which fairness definitions are most important for the particular domain and ignore the rest. Our work of dividing fairness tries to make the choice easier, as choosing metrics from a group of 10 options is much simpler than choosing from 30 choices. Using our results of Table 8.3 and Table 8.4, in a specific domain, if group fairness is more important than individual fairness, then cluster four will be given more priority than clusters two and five (Table 8.3). Once a cluster is given priority, one or two metrics can be chosen to represent the whole cluster. That means our whole work boils down to minimizing the number of metrics to look at and covering a wide range of fairness. We believe future researchers and industry practitioners will use our work as a guide and that will be the fulfillment of this study.

8.5.3 What to do when the metrics contradict each other?

We have seen that there are scenarios where fairness metrics contradict each other. According to some metrics, the prediction is fair, where some other metrics disagree. Fairness metrics find out how critical the errors of a prediction model are. It is the decision of the policymaker or the domain

expert to choose appropriate fairness metrics based on what kind of bias is more important for the specific domain. For example, consider the following two scenarios:

- Suppose we are predicting if a patient has cancer or not, depending on the symptoms. Here predicting a benign case as malignant is not very dangerous but predicting a malignant case as benign is extremely dangerous. A wrong diagnosis for an actual cancer patient will delay the treatment, and the patient may die. That means *false negative* is more important here.
- Suppose we are predicting if future performance of a student based on previous records. Here if we predict a good student as bad, that is not that fatal. However, if a student who really needs special attention and help from teachers, is given a good rating then it will be misery for that student. That means *false positive* is more important here.

If we know which metrics look at what kind of error, it will be easier for the decision-maker to choose. That said, based on the guidance we have provided, in case of contradiction among metrics, one metric over another will be given priority.

8.6 Threats to Validity & Future Work

This paper explores machine learning methods for software engineering. One issue with any paper like this is a few selection and evaluation biases along with construct and external validity based on the choice of models, datasets, and methods. In the future, we plan to address the apparent threats to validity that this paper has not fully addressed.

Construct Validity: Here, we have used popular *hierarchical* clustering called *agglomerative* approach, as the number of clusters was not known beforehand. In future, we need to experiment with other clustering techniques to check for conclusion stability. This analysis used logistic regression (LR), as much prior work on fairness has also used LR [Bel18; Cha20b]. Nevertheless, in future work, we need to explore some other classification models including DL models. Also, the metric clusters found in Table 8.3 and Table 8.4 are created using the results of our choice ML models, dissimilarity measures, and cutting point in the dendrogram. Thus, choosing one metric from each cluster may contain some risk, and researchers need to be careful while making informed choices about metric selection.

Evaluation Bias: We have used 30 metrics taken from IBM AIF360 [Bel18]. We have also covered most of the metrics from Fairkit-learn [Joh20] and Fairlearn [Faib]. There are other metrics and definitions of fairness, thus the results of this study may not generalize to all available metrics. But the 30 metrics covered in this study are widely used in the fairness domain [Kea18; Fri19; Zaf17; Bis20; Cot19].

External Validity: We have used seven datasets. In the fairness domain, one big challenge is the availability of adequate datasets. It would be insightful to re-run this study on new datasets and also on other domains.

Sampling Bias: In this work we used thresholds recommended by IBM AIF360 ("fair" means -0.1,0.1 or 0.8,1.2 for different kinds of metric). Future work should explore the sensitivity of our conclusions to changes in those thresholds.

Another issue with sampling bias is that our analysis is based on the data of Table 2.1. We recommend that when new data becomes available, we test the conclusions of this paper against that new data. That would not be an arduous task (and to simplify that task, we have placed all our scripts online in order).

8.7 Conclusion

Fairness is a rapidly evolving domain and the number of fairness metrics is increasing exponentially. While performing our literature review we saw the current practice in this domain is to rely on a handful of metrics and ignoring the rest. But which metrics can be ignored? Which are essential?

To answer these questions, this paper has experimented with the following *metrics selection tactic*: When applied, the paper reported that this tactic could reduce dozens of metrics to just a handful. We found:

- RQ1 showed that all the metrics do not agree with each other when labeling a model as fair or unfair.
- RQ2 showed that metrics can be clustered together based on how they measure bias. Each of the resultant clusters measures different types of bias and selecting one metric from each cluster should be representative enough to measure increase or decrease in bias in other metrics in the same cluster.
- RQ3 showed that we could ignore at least two of those clusters, since they were not "sensitive". Recall that by "insensitive" clusters, we mean those where changes to the data did not change the fairness scores.
- RQ4 showed that this reduced set actually predicts for different things. That said, it is no longer necessary (or even possible) to satisfy all these fairness metrics.

From these results, we argue that:

- There are many spurious fairness metrics; i.e. metrics that measure very similar things.
- To simplify fairness testing, just (a) determine what type of fairness is desirable (for a list of types, see Table 8.3 and Table 8.4); then (b) look up those types in our clusters; then (c) just test for one item per cluster.
- While this approach does not completely remove all issues with fairness testing, it does reduce a very complex problem of (say) 30 metrics to a much smaller and manageable set.
- Also, the methods of this paper could be used as a litmus test to prune away spurious new metrics that merely report the same thing as existing metrics.

Part 3 - Future Directions

CHAPTER

9

CAUSALITY-BASED TESTING FOR SOFTWARE FAIRNESS

This section extends our work to include causality. Causality is a vast domain and we focused on a very limited part. Here, we take an algorithmic causal approach that states: (1) It is unfair if final decisions make an unnecessarily causal connection between decisions and attributes that identify specific social groups (e.g. race, gender, etc); (2) decisions can be made fairer by reducing the number of those causal connections. This study was submitted to ESEC/FSE 2022. We believe, future researchers will try to cover broader spectrum of causality in software fairness domain.

9.1 Introduction

As software engineering evolves, so do our notions of software testing. For example, until the age of smartphones, software engineers rarely tested the energy requirements of their software. Nowadays, such energy testing is standard in GUI design [LV15] and in software configuration studies [Nai18]. Similarly, once upon a time, software engineers were not required to test for bias against different social groups within their systems. Now, that has changed given all the recent high-profile examples of software-induced bias. Hence, testing software for bias and mitigating bias is now an important step in the software development life cycle. But how to test for bias? And, if detected, how can we mitigate that problem?

Answering these questions has become increasingly complex. For example, Majumder et al. [Maj21] reported that there are over two dozen widely-used measures of fairness. And as to methods for reducing that bias, the IBM AI360 toolkit contains dozens of algorithms [Bel18], each of which is

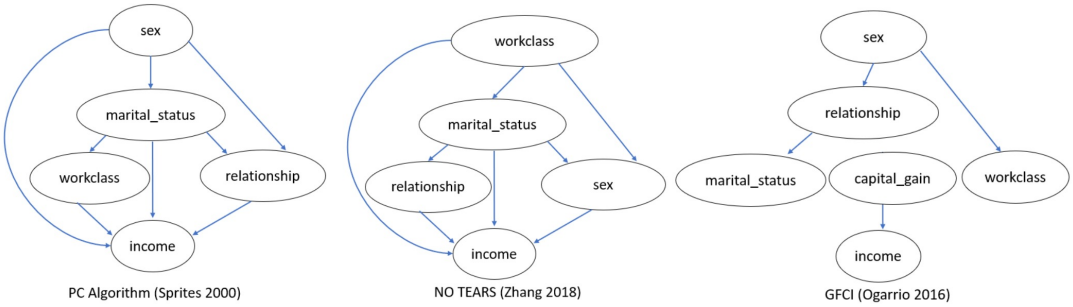

Figure 9.1 Three algorithms generating different causal interactions between attributes. It is difficult to choose a single one.

useful in different circumstances. Faced with this complex labyrinth of fairness assessment measures and mitigation algorithms, we seek some underlying principle(s) that fundamentally simplifies bias measurement and mitigation.

Potentially, causal reasoning is one way to clarify testing for software fairness. In that approach:

- Bias is *caused by* decisions being unnecessarily connected to attributes that identify social groups (e.g. race, gender, etc).
- Bias can be mitigated by *reducing those causal connections*.

One reason to take this causal approach is that causality has a long-standing theoretical clarity. In 1973, David Lewis [Lew73] defined it as follows "causation is something that makes a difference, and the difference it makes must be a difference from what would have happened without it". That is to say, T causes Y iff changing T leads to a change in Y, keeping everything else constant. Also, there is an abundance of off-the-shelf causal graph generators including those we study here such as PC, NO TEARS, GFCI, and (more recently) DoWhy [Spi93; Zhe18; Oga16; Sha20a; Sha19; Sha20b].

But there is a problem: these graphs are complex to generate. Numerous heuristics are used to speed up that process. Different graph generators used different heuristics and so produce different causal graphs (for examples of this, see later in this paper). This makes it difficult to offer definitive mitigation for decision bias.

This paper tests the following conjecture:

> *When* one *causal graph is uninformative, perhaps we can do better by sampling the causal inference from multiple graphs.*

That is, instead of tackling problems of bias mitigation using a single causal graph (which may have a somewhat variable structure, depending on how you generated it), it is better to report unfairness propensity as a probability P computed using multi-graph causal inference. Our tool for this work is the DoWhy causal graph generator [Sha20a] which, internally, generates multiple approximate causal graphs which are then summarised as a "propensity" $P(T,Y)$ reporting the probability that changing T leads to a change in Y, keeping everything else constant.

- Recidivism models are more likely to falsely label black defendants as future criminals at twice the rate as white defendants [Mac].
- Widely-used facial recognition software that predicts characteristics, such as gender and age, has a much higher error rate for dark-skinned women compared to light-skinned men [Ski].
- Amazon's ML software to offer same-day delivery to prime users became biased against black neighborhoods [Amab].
- Google Translate, the most popular translation engine in the world, shows gender bias. "She is an engineer, He is a nurse" is translated into Turkish and then again into English becomes "He is an engineer, She is a nurse" [Cal17a].
- For more examples, see Rudin [Rud19], Nobel [Nob18], Gebru [Geb].

Table 9.1 Biases from decision-based software.

To the best of our knowledge, this is the first SE work where causal inference is used to assess ethical bias in SE models. More specifically, while DoWhy has been previously applied to a wide range of other problems [Sha20a], this paper is the first to apply this multi-causal-graph approach to the problem of software fairness. Hence the task of this paper is to:

> *Assess the value of DoWhy's multi-causal-graph propensity metric for bias measurement and mitigation.*

The rest of this paper shows that DoWhy's $P(T,Y)$ probabilistic propensity measure satisfies the required properties of an unfairness metric. Specifically, for decisions Y, when we set T to fairness-related attributes such as age, race, gender, etc then:

- P decreases as we apply the standard bias mitigation algorithms;
- When data is perturbed, P changes in a manner predicted by theory (in this paper, perturbations include random common causes, placebo treatments, and explorations of random subsets)

Hence, we will conclude that this probabilistic propensity P is a useful tool for causality-based testing for software fairness.

9.1.1 Motivating Example

This section reviews some problems and benefits of using causal graphs to test fairness. To begin, we say that some decisions from a model have *favorable labels*; i.e. they give an advantage to the receiver such as receiving a loan, being hired for a job. A *sensitive/protected attribute* is an attribute

that divides the whole population into two groups (privileged & unprivileged) that have differences in terms of receiving benefits. For example:

- In case of credit card application data sets, based on protected attribute “sex”, “male” are privileged and “female” are “unprivileged”;
- In the case of health data sets, based on protected attribute “age”, “young” are privileged and “old” are unprivileged.

When studying bias in decision-making software, we look at data used to generate decisions about favorable labels. Causal graphs report the connection between attributes and decisions. For example, in the “Adult” data set [Adu], we might use such causal connections to explore the effect of sex on income. To that end. we might use some algorithms like PC, NO TEARS, or GFCI [Spi93; Zhe18; Oga16] to build the causal graphs of Figure 9.1. The Adult data set has 14 attributes of which many do not appear in the graphs (since the focus is here on the protected attribute and the outcome only).

The starting point of this paper was the observation that different causal graph generators produce different graphs. For example:

- In two of these graphs, there is a direct edge (path length = 1) between sex and income. If these graphs were being used to assess loan applications, then we might become nervous about an undue influence of gender or the ability to raise capital (for the purposes of, say starting a business).
- In the third graph (on the right-hand-side) no such path exists between sex and income (though some other income-related node has appeared called capital_gain).

This non-determinacy in the generated causal graphs raises questions about the work of Zhang et al. [Zha16b] who used the PC causal graph generator [Spi93] to assess fairness within a data set. Since they explored a single graph, and the structure of those graphs can vary, we were skeptical about using their methods to make conclusions on how to measure and mitigate bias.

That said, we note in Figure 9.1 that a majority (two out of three) of the graphs have a direct edge from sex to income. Perhaps we can find stable conclusions if we reported the average case structure of multiple causal graphs. For that purpose, we turn to the DoWhy algorithm [Sha20a; Sha19; Sha20b]. After building one graph, this algorithm acts like a scientist exploring a theory. Numerous robustness tests are applied to the graph (described below) that report what effects are stable across (e.g.) multiple subsets of the data.

9.1.2 Fairness Metrics

One complaint we have with the current fairness research is that it is growing too complex, perhaps exploring in too many metrics, some of which might be spurious. As evidence of that, consider the toolkits generated by fairness researchers. Between them, the IBM AIF360 [Bel18], Fairkit-learn [Joh20], and Fairlearn [Faib] toolkits contain more than fifty different fairness metrics.

Table 9.2 Performance measures

	Actual Positive	Actual Negative
Predicted Positive	TP; (and TPR = TP/(TP+FN))	FP (and FPR = FP/(FP+TN))
Predicted Negative	FN (and FNR = FN/(TP+FN))	TN (and TNR=TN/(TN+FP))

Table 9.2 shows the performance scores that might be collected from a binary classifier. From these scores we can comment on two widely-used fairness metrics [Cha20b; Cha21; Bis20; Cha19]:

- **Equal Opportunity Difference (EOD)**: Difference of True Positive Rates(TPR) for the different groups in a population [Bel18].
- **Average Odds Difference (AOD)**: Average of difference in False Positive Rates(FPR) and True Positive Rates(TPR) for the different groups in a population [Bel18].

But note the problem here: these scores are not fundamental to the domain. Instead, they are a report of what happens when that data is processed via a particular learner. Ideally, we seek fairness measures that are not some quirk of which learner is applied to the data. We say that this is ideal since, if that can be achieved, then any mitigation applied to the data *apply to a wide range of learners applied to that data*.

In practice this ideal is hard to achieve. We saw in Figure 9.1 that if we reason at a more fundamental level about the causal connections in the data, then it is still possible to generate different models. The way out of this conundrum is *not* to rely on a single learner or a single causal graph. Rather, we should report propensities computed across multiple causal models (which is the approach taken for the rest of this paper).

9.1.3 Causality

Most of the prior fairness works tested the performance of their algorithms using a handful of fairness metrics as mentioned in the earlier section. We, in this work, decided to test software for fairness using causal inference.

Reichenbach's Common Cause Principle states that ***"if two things, A and B are dependent; Then either A causes B or B causes A or something else causes both A and B"*** [**Hitchcock**]. **Causal Inference** is the procedure of finding the right one. We now need some mathematical foundations to proceed further. There is more than one mathematical model of causality. We, in this work, decided to use a widely used causal modeling framework that is Structural Causal Models (SCM) from Pearl et al. [Hal05] (this shares much in common with the approaches of Robins al. [Rob86] and Sprites at al. [Spi93] and the other causal graph generators used in this paper). A ***Structural Causal Model (SCM)*** [Gal98] M is a triplet $< U, V, F >$ where

- V is a set of observed random variables that form the causal system of our interest;

- U is a set of latent (or unobservable) background variables that represent all possible causes of V and which jointly follow some distribution $P(U)$;
- F is a set of functions $\{f_1, f_2, ..., f_n\}$, one for each $V_i \in V$, such that $V_i = f_i(pa_i, U_{pa_i}), pa_i \subset V_i, U_{pa_i} \subset U$. Such equations are also known as structural equations [Bol89].

Every SCM M includes a **causal diagram G**. It is represented as a **Directed Acyclic Graph (DAG)**, where:

- Each $V_i \in V$ is a node;
- There is an edge $V_i \implies V_j$ if $V_i \in Pa_j$

Here are some key intuitions about the causal graph.

- Assumptions are encoded by missing edge and direction of edges. For example, an arrow is drawn from node A to node B. That means, when the value of A changes, the value of B is going to change weight in some way. But that weight could be zero also. So, drawing an edge between two nodes is not declaring anything. It assumes that there might be a relationship between node A and node B. But if that edge is removed, then the weight is explicitly set to zero. In that case, no matter how A changes, the value of B is not going to change.
- The relationships in the causal graph represent stable and independent mechanisms. For example, we have a large causal graph that represents a bunch of different stuff in some system or environment. If we reach in and change some of the edges, we can expect that the rest of the causal graph is not going to be affected.
- Graphs are a tool to help us reason about a specific problem. We can write these graphs at different layers of abstraction. At a very detailed level, we can capture major causal relationships or the most minor ones. The right way will depend on the questions we are trying to answer, basically, on the context of the problem.

There are many ways to generate causal graphs from the observed data. Initially, we tried to use some of those to generate causal graphs on our data sets. The first one is the PC algorithm by Sprites et al. [Spi93]; the second one is NOTEARS by Zhang et al. [Zhe18]; and the third one is GFCI by Ogarrio et al. [Oga16]. There are two kinds of graphical models - undirected ones (Markov model) and directed ones (Bayesian model). These graphs can be generated in two ways. The first one is a constraint-based learning method where a bunch of hypothesis tests are done to figure out the independent sets between rows and then the graph is constructed out of the independent sets. The second type of method is score based learning where a score function is defined and the graph is optimized by optimizing the score function. PC algorithm [Spi93] is constraint based, NOTEARS [Zhe18] is score based, and GFCI [Oga16] is a combination of constraint based and score based.

The **PC algorithm** [Spi93] starts with making a complete undirected graph where number of nodes is equal to the number of attributes and all the nodes are connected. The second step is removal of some particular edges. Conditional independence tests are done to decide if an edge is removed or retained. For each edge, the PC algorithm tests if the pair of variables, X and Y, connected by the edge are independent; conditioning on a subset Z of all neighbors of X and Y. Finally, after removal of several edges, one DAG is generated as output. The only shortcoming of PC algorithm is very high running time. In the worst-case, its run time is exponential to the number of nodes (variables), and thus it is inefficient when being applied to high dimensional data.

Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning (NO TEARS) is a continuous optimization algorithm [Zhe18]. This algorithm tries to find the DAG that best fits the data by continuously optimizing the score function. NOTEARS converts a combinatorial optimization into a continuous one which is much easier to solve. The latter one could be solved by any numerical solver.

Greedy Fast Causal Inference (GFCI) [Oga16] has two main phases. In the first phase, the algorithm creates a bunch of *Causal Bayesian Networks* (CBN) and greedily search over them to find the highest scoring network. The only assumption is that there is no unmeasured confounder. A *confounder* is a variable that influences both the dependent variable and the independent variable. The output of the first phase goes to the second phase. In the second phase, Fast Causal Inference (FCI) algorithm [Spi02] is applied to post-processes the output to produce a representation of a set of models that may include unmeasured confounders.

As mentioned above, these three algorithms have a problem: they do not generate a definitive single canonical causal graph. Therefore, we stopped using them and searched for alternatives. We found the **DoWhy** causal inference library introduced by Microsoft [Sha20a]. We chose DoWhy for three reasons:

- DoWhy acts like a scientist carefully probing a theory. The result of that probing is a list of results showing how much one attribute is causally connected to another.
- It is widely used (in GitHub, the repository has 576 forks and 3.7k stars as of March, 2022).
- In their presentation *Causal inference for machine learning: Generalization, Explanation, and Fairness*[1], the primary author of DoWhy raises the possibility that, potentially, DoWhy could be used in the fairness domain. That said, they do not check that possibility. As near as we can tell, this paper is the first extensive study of DoWhy as a bias measurement and mitigation tool.

DoWhy runs in four steps:

- **Modeling:** Here, we manually craft an initial causal graph (use the modeling rules shown below). One advantage of using DoWhy is that, since it aggressively tests each model, there is no requirement here that these initial models have to be fully accurate.

[1] `http://www.amitsharma.in/files/causality-for-machine-learning-econdatascience-seminar.pdf`

- **Identification:** Next, DoWhy reflects over the data set looking for what subset of the edges (created in the modeling step) connect inputs to output. Note that this is a nondeterministic process so, in this step, DoWhy makes many experiments to produce many candidate models.
- **Estimation:** Here, DoWhy selects "good" models; i.e. those that balance bias and variances (and "bias", in this context measures the difference between model estimates learned from all data versus estimates learned from some subset used to train a specific model).
- **Refutation:** Finally, those good models are further pruned via numerous sensitivity analyses that check if estimates can be refuted. For details on this refutation process, see below.

The result of the above is a lot of predictions from the good models that survived refutation. By measuring frequency counts across that list, DoWhy computes probabilistic propensity scores.

It has not escaped our attention that this four step cycle makes many assumptions. Hence, it is an open issue of the probabilistic propensity scores computed in this manner are realistic measures of anything at all. The rest of this paper describes experiments that give confidence that the DoWhy procedure is actually useful for measuring and mitigating bias.

9.2 Methodology

9.2.1 Causal modeling with DoWhy

This section offers more details on how we use DoWhy. Note that step1 requires some manual work while the rest are all automatic.

Step 1 - Model a causal problem

DoWhy requires some initial causal graph as an input form. Accordingly, we created an initial causal graph using the following modeling rules:

- We spend half an hour reasoning about each data set.
- In that time, we selected 2 or 3 attributes at random from our data sets and then applied domain knowledge to postulate zero or more edges between these attributes.
- For each protected attribute, we included one direct edge (i.e. path length=1) from protected attribute toward the outcome variable (dependent variable).
- There is at least one indirect path (length > 1) from protected attribute toward the outcome variable (dependent variable).
- There are no cycles in the generated graph
- The maximum path length is less than five
- All other attributes (not present in the graph) are treated as unobserved confounders; i.e. they were added into the graph in-between the protected attributes and the decisions.

For example, from the Adult dataset, here is the causal graph generated by our manual modeling. Protected attribute is “sex” and the outcome variable is “income”.

sex -> income;
sex -> marital_status; marital_status -> income;
workclass -> occupation;
sex -> occupation; occupation -> income;
marital_status -> relationship; relationship -> income;
workclass -> income;
U->sex; U->race; U->income;

In the above causal graph, we consider two kinds of edges from “sex” to “income”. The first one is direct edge (sex -> income); the second one is through one or more than one attribute (for example, sex -> marital_status; marital_status -> income). The first one denotes ***direct discrimination*** and the second one denotes ***indirect discrimination***. “U” denotes unobserved confounders. This causal graph contains our assumptions about all kinds of interventions possible among “sex” and “income”. We use causal inference to test whether that is valid or not.

Step 2 - Identify a target estimated under the model

Based on the causal graph, DoWhy finds all possible ways of identifying a desired causal effect based on the graphical model. It uses graph-based criteria and do-calculus to find potential ways to find expressions that can identify the causal effects.

Step 3 - Estimate causal effects

DoWhy supports various estimation methods– most of them are directly implemented in DoWhy; some of them are part of EconML [KB19] which is a Python package for ML-based heterogeneous treatment effects estimation. Some highly cited estimation methods are propensity score based matching [ROS83], simple conditioning matching, outcome based estimation (T-learner, X-learner) [Kü19], loss-based estimation (R-learner) [Nie20], and threshold based estimation. We used only propensity score based estimation in this work. Future extension could cover other methods.

Step 4 - Try to refute the obtained estimate

Finally, DoWhy tries to validate the models it has generated. Here, it does various tests to see whether the edges in the learned models are useful. To that end, various tests are performed:

- Add Random Common Cause: Does the estimation method change its estimate after we add an independent random variable as a common cause to the data set? (Hint: It should not)
- Placebo Treatment: What happens to the estimated causal effect when we replace the true treatment variable with an independent random variable? (Hint: the effect should go to zero)
- Data Subsets Validation: Does the estimated effect change significantly when we replace the given data set with a randomly selected subset? (Hint: It should not)

9.2.2 Experimental Setup

All our experiments were repeated ten times and we report medians over that sample. We used the ten data sets described in Table 2.1. This data was selected since it is the standard test suite used to test fairness algorithms by many authors [Cha20b; Cha21; Bis20; Cha19; Cha22; Cha20c]. The last two columns of that table show the binary classification decision made by this data.

Before applying DoWhy, we pre-processed the data. The rows containing missing values are ignored, continuous features are converted to categorical (e.g., age<25: young, age>=25: old), non-numerical features are converted to numerical (e.g., male: 1, female: 0), finally, all the feature values are normalized (converted between 0 to 1). Then we apply DoWhy causal inference as described in the above section.

9.2.3 Evaluation

For causal inference, the most popular evaluation metric is *propensity score* [Ros83]. As defined in our introduction, our propensity score $P(T, Y)$ is the probability that changing T leads to a change in Y, keeping everything else constant. When Y is some decision (like "reject their loan application") and T are attributes related to different social groups (e.g. age, sex,race, etc) then

- **Higher** *propensity score* signifies **higher** probability of causal model assumptions being true i.e. **discrimination is there**.
- **Lower** *propensity score* signifies **lower** probability of causal model assumptions being true i.e. **discrimination is not there**.

This work assesses causal graph reasoning via changes to the propensity score. What we do not do is report prediction performance scores like accuracy, recall, etc. As described above in §**??**, in 2021, Chakraborty et al. [Cha21] showed that many of the operators applied to fix fairness are also the sampling operators widely used in software analytics to improve prediction. Hence, in studies with the same data sets explored here, Chakraborty et al. showed that their Fair-SMOTE system could improve not only fairness measures, but also predictive performance. Since this paper also uses Fair-SMOTE on the same data sets as seen in that 2021 paper, we will explore here new issues such as the efficacy of probabilistic propensities calculated across multiple causal graphs (rather than older, now resolved, issues such as the effect of bias mitigation on prediction).

As to comparing treatments, we apply a simple "less than" operator between the medians seen in ten runs of our experimental rig. This is not our normal practice (usually we report results using some Scott-Knott [Gho15; Mit13] variant with a non-parametric bootstrap significance tests and effect size tests). But in this case, there was little added value in a more complex statistical analysis. For example, the propensity scores for "Default" and "Fair-SMOTE" from our causal models are:

$$\begin{aligned} \textit{Default} &= (.17,.04,.06,\mathbf{.01},.07,.03,.31,.08,.55,.06,.04,.04) \\ \textit{FairSMOTE} &= (.02,.03,.03,\mathbf{.05},.05,.03,.08,.04,.29,.02,.01,.01) \end{aligned}$$

Table 9.3 Change of propensity scores for four state-of-the-art algorithms for different treatments. Treatment 1 = "add random common cause"; Treatment 2 = "replace treatment with placebo"; Treatment 3 = "remove random subset of data"; Darker pink cells contain the best scores (lowest bias); Lighter pink = second best; White = worst. This table contains results for seven datasets. For ten datasets, please see our paper.

Dataset	Protected Attribute	Treatments	Default	Fairway	LFR	Reweighing	Fair-SMOTE
Adult Census Income	Sex	Causal Model	0.17	0.11	0.07	0.08	0.02
		Treatment 1	0.16	0.11	0.17	008	0.01
		Treatment 2	0.01	0	0.01	0	0
		Treatment 3	0.16	0.11	0.16	0.08	0.02
	Race	Causal Model	0.04	0.03	0	0.02	0.03
		Treatment 1	0.03	0.04	0	0.03	0.04
		Treatment 2	0.02	0	0	0	0
		Treatment 3	0.04	0.03	0	0.03	0.04
Compas	Sex	Causal Model	0.06	0.04	0.05	0.05	0.03
		Treatment 1	0.07	0.05	0.12	0.05	0.03
		Treatment 2	0	0	0	0	0
		Treatment 3	0.07	0.05	0.05	0.05	0.02
	Race	Causal Model	0.01	0	0.01	0.01	0.05
		Treatment 1	0.02	0	0.03	0.02	0.04
		Treatment 2	0	0	0	0	0
		Treatment 3	0.01	0	0.04	0.01	0.06
German	Sex	Causal Model	0.07	0.02	0.02	0.03	0.05
		Treatment 1	0.06	0.02	0.02	0.03	0.02
		Treatment 2	0	0	0	0	0
		Treatment 3	0.07	0.04	0.11	0.03	0.04
Default Credit	Sex	Causal Model	0.03	0.03	0.03	0.05	0.03
		Treatment 1	0.03	0.03	0.05	0.05	0.03
		Treatment 2	0	0	0	0	0
		Treatment 3	0.03	0.03	0.05	0.05	0.03
Heart Health	Sex	Causal Model	0.31	0.34	0.61	0.19	0.08
		Treatment 1	0.29	0.35	0.37	0.18	0.04
		Treatment 2	0	0	0	0	0
		Treatment 3	0.31	0.34	0.51	0.19	0.06
MEPS15	Race	Causal Model	0.04	0.03	0.07	0.01	0.01
		Treatment 1	0.03	0.03	0.05	0.01	0.01
		Treatment 2	0	0	0	0	0
		Treatment 3	0.04	0.04	0.05	0.01	0.01
MEPS16	Race	Causal Model	0.04	0.03	0.11	0.01	0.01
		Treatment 1	0.03	0.03	0.11	0.01	0.01
		Treatment 2	0	0	0	0	0
		Treatment 3	0.04	0.03	0.11	0.01	0.01

With one exception, the overall trend of *default[i] - FairSmote[i]* is very clear and supportive of our conclusion that bias mitigation operators change propensity scores in the expected manner (i.e. they decrease it). That exception was $i = 4$ (where propensity increased from 0.01 to 0.05).

(Aside: another justification of using just "less than" comes from our **RQ3**, below, where we report that the variability in the results across our ten runs was very small (under 20% of the median).)

9.3 Results

Table 9.3 shows results for four treatments:

- **Treatment 0/Causal Model** - Original causal model is used and *propensity score* is measured.
- **Treatment 1** - What is the change of *propensity score* when an independent random variable is added as a common cause?
- **Treatment 2** - What is the change of *propensity score* when the true treatment is replaced with an independent random variable?
- **Treatment 3** - What is the change of *propensity score* when we use a random subset of the data rather than the full data?

The last five columns of Table 9.3 show results from five experiments

- The first experiment is "Default" i.e. raw data is used. A causal model is built containing the assumption of direct and indirect discrimination (protected attribute has direct and indirect path towards the outcome). Then we do the identification, estimation and refutation respectively as described earlier.
- The remaining four columns show results for each of the four state-of-the-art fairness algorithms.

9.3.1 Research Questions

Our results are structured around three research questions.

RQ1. Are we measuring causality?

The DoWhy's refutation step comes with various tests that check if a modeling system actually measures causality. **RQ1** reports the results of those tests.

In machine learning, the normal approach of checking robustness of any algorithm is cross-validation of data. But in case of causality, the main idea is to less depend on the data and rely on the causal structure of the graph. That is why the best practice for checking robustness of causal approaches are using refutation tests. To be certain, not just use one refutation test, use more. There are two kinds of refutation tests possible.

- Unit testing: In a software development process, unit testing is just checking one part of the analysis pipeline. In our case, we can just test whether the modeling is correct using a conditional independence test or similarly we can test identification step using D-separation test. Some other unit tests could be "bootstrap refuter", and "data subset refuter".

- Integration testing: In software development, integration testing means testing the whole software pipeline. In our case, this means not just testing one step, rather testing all steps of causal inference (modeling, identification, estimation) together. Some popular integration refutation tests are - "placebo treatment refuter", "dummy outcome refuter", "random common cause refuter", and "sensitivity analysis".

We, in this work, chose to do one unit test and two integration tests. In Table 9.3, "Treatment 1" means "adding random common cause" According to the notes above (in Step 4 of §9.2.1), this should not alter the estimates of a causal graph and, as seen in Table 9.3, this is indeed the case (evidence: all the "Treatment 1" results are nearly identical to the "Causal model" results).

"Treatment 2" means "replace treatment with placebo". According to the notes above, this should drive the propensities to zero (why? there should be no causal connection between placebos and output). In Table 9.3, this is indeed the case (evidence: observe all the "0" values in the Treatment 2 rows).

"Treatment 3" means "remove random subset of data". In our case, we removed 20% of the data. According to the notes above, a general effect should also hold in a majority subset of the data. This is indeed the case : all the "Treatment 3" values are nearly identical to "Causal Model".

Thus, the answer for RQ1 is **"As measured by the refutation tests of DoWhy, our modeling system does measure causality"**.

RQ2. Is DoWhy's propensity measure also a measure for bias?

As stated above, the four modeling steps of DoWhy makes many assumptions. Hence, it is an open issue whether the probabilistic propensity scores computed in this manner are realistic measures of anything at all. **RQ2** asks if that propensity measure is actually useful for measuring bias.

Our reading of the literature is that, in terms of being able to reduce bias, Fair-SMOTE is arguably the current state-of-the-art[2] and the others are ordered as follows:

$$\textit{do nothing} < \textit{Fairway} < \textit{LFR} < \textit{Reweighing} < \textit{FairSMOTE}$$

Hence we say that if DoWhy is an effective measure of fairness, as seen in Table 9.3, its propensity scores (connecting protected attributes to decisions) *decrease* as we work left to right across this ordering. On the causal Model row, the cells with least scores are highlighted in dark color and second least cells are highlighted in light color. If we aggregate across all the data sets, Fair-SMOTE

[2]For notes on why that is so, please see our discussion section.

wins 8/12 times; LFR wins 4/12 times; Reweighing and Fairway wins 3/12 times (based on number of dark cells).

Thus, the answer for RQ2 is **"Yes, probabilistic propensity score is a measure of bias"**.

RQ3. Can multi-causal-graph reasoning stabilize causal reasoning?

Finally, we return to the motivating example shown at the start of this paper. In that section, we noted that individual causal models may not be canonical. If that instability persists over all our experiments, then we would be unable to offer definitive recommendations about bias reduction.

What we can say from our experiments is that our results are stable. Despite all the perturbations made by DoWhy against our models, the variability around our median scores are very small. Specifically:

- The interquartile range (IQR) is a non-parametric measure of variance in a distribution. It is measured as the difference between the 75the and 25th percentile.
- Across all our results, the IQR was less than 20% of the median value.

Thus, the answer for RQ3 is **"Yes, we can find stable causal properties using multi-causal graph reasoning ".**

9.3.2 Discussion: Notable Features of Table 9.3

The darker colors in Table 9.3 indicate where we observed the largest bias reduction. If we calculate, Fair-SMOTE wins 8/12 times; LFR wins 4/12 times; Reweighing and Fairway wins 3/12 times (based on number of dark cells). Why does Fair-SMOTE work so well?

We believe the main reason behind is that Fair-SMOTE is an *oversampling* algorithm that introduces new data points by mutation process. For example, say a data point is defined as

$$< X_1, X_2, X_3, X_4, P, X_6, X_7, C >$$

where X_i are feature values, P is the protected attribute value and C is the class value. Fair-SMOTE creates new data points such as

$$< X_1', X_2', X_3', X_4', P, X_6', X_7', C >$$

by extrapolating the values of X_i and keeping the same value of P and C. This process continues until the data becomes balanced based on class and protected attribute i.e. data has equal number of privileged & favorable (for Adult data set, it is "male" sex and "high" income), privileged & unfavorable ("male" sex and "low" income), unprivileged & favorable ("female" sex and "high" income), and unprivileged & unfavorable ("female" sex and "low" income) samples.

What is the effect of this style of over-sampling? In short, the more imbalanced the initial data is, the more number of samples are generated by Fair-SMOTE. In our view, this leads to better causal

fairness because causal model gets more and diverse number of examples to learn from.

9.4 Threats to Validity

Sampling Bias - We have used ten well-known data sets in our experiments. Most of the prior works [Cal17b; Gal17; Zha18; Kam18; Cha19] used one or two data sets where we used ten of them. In the future, we will explore more data sets. The conclusions may change a bit if other data sets are used.
Evaluation Bias - In this study we did not use any fairness metric. We used propensity score which is vastly used in statistics and machine learning domain. In future, we will explore more evaluation criteria.
Conclusion Validity - We used DoWhy causal inference library here. There are other causal inference algorithms available too. We chose DoWhy because of its recent popularity and there are many other causal algorithms under the hood of DoWhy. However, using other causal inference libraries/algorithms may change the conclusions.
Construct Validity - DoWhy requires an initial graph to start with. We have generated that input graph based on our own rules as described in section 9.2.1. We would like to see future researchers refute/improve our results by using different rules of graph generation.
External validity - This work is based on binary classification and tabular data which are very common in AI software. Causal inference can be extended to regression algorithms. Causal fairness can be very helpful in other domains such as text mining and image processing.

9.5 Conclusion

This paper has assessed the value of causality-based assessment of bias. Motivated by an overwhelming number of fairness metrics [Maj21] and bias mitigation algorithms [Bel18; Joh20; Faib], we sought a more fundamental view of the whole issue of bias. Previous work suggested that causality might be such a useful framework for reflecting on software bias [Sha20a]. In that approach, we say that:

- It is unfair if final decisions make an unnecessarily causal connection between decisions and attributes that identify specific social groups (e.g. race, gender, etc);
- Decisions can be made fairer by reducing the number of those causal connection

Prior work that attempted this causal approach [Zha16b] was incomplete in that it only explored a single graph (thereby overlooking the variable nature of causal graphs we can generate from data, see Figure 9.1).

We conjectured that

> *When* one *causal graph is uninformative, perhaps we can do better by sampling the causal inference from multiple graphs.*

Our pre-experimental concern with this approach was that it would introduce so much variance that it would be ineffective for measuring bias. However, based on the experiments of this paper, we are now far more optimistic about using causality to address bias. We find that, while the structure of one causal graph might be unstable, by sampling multiple graphs, we can report causal structures (**RQ1**) that measure bias (**RQ2**) and which are stable across a wide range of perturbations (**RQ3**).

CHAPTER

10

CONCLUSION & FUTURE WORK

This section covers future direction of software fairness research from our point of view.

10.1 Evolution of Software Fairness

Fairness became a matter of concern in the ML community over the last decade. Software engineering research community took really long time to accept that fairness is also a concern in software development life cycle. In 2018, Yuriy Brun and Alexandra Meliou from University of Massachusetts Amherst published a paper on ESEC/FSE 2018 [Bru18b]. That is the first work that introduces fairness in software. After that, we see a gradual increase of fairness works in software engineering venues. Our work Fair-SMOTE has received **ACM SIGSOFT Distinguished Paper Award** in ESEC/FSE 2021. That explains the relevance of fairness in software engineering research. We believe more and more works focusing on software fairness will be published in the future. ICSE 2022 is going to organize the second workshop (Fairware 2022 [1]) dedicated to software fairness only. We see the prospect of more works on software fairness in the future.

10.2 Fairness in the real world

Researchers are trying to develop various algorithms or optimization techniques to achieve fairness in ML models. But these algorithms are not yet used in the industries. A Google paper states the issues of using fairness algorithms in the real-world [Beu19]. Microsoft did a survey study to find out the challenges software engineers face to develop a fair ML software. The study consisted thirty-five

[1] https://fairwares.github.io/

semi-structured interviews and an anonymous survey of 267 ML practitioners[Hol19]. The main purpose of the study was to find the gap between academic solutions and real world implementation challenges. One of the reasons of this gap is “Bias” and “Fairness” in real-life applications are very subjective, it depends on the type of application. So far, there are three kinds of definitions of fairness exists in the literature [CD18].

- Anti-classification - Protected attributes (E.g.- gender, race) are not explicitly used to make decisions.
- Classification Parity - Common measures of predictive performance (false positive rate and false negative rate) are equal or almost same across groups defined by the protected attributes.
- Calibration - Conditional on risk estimates, outcomes are independent of protected attributes.

Which definition we use depends on the domain. we say that before jumping to measure bias or mitigate bias, practitioners need to be careful about the context. For example: There are certain areas where we want discrimination to make things fair. Imagine a scenario, where a company is looking for hiring top fifty people from an exam. If there are more than fifty male candidates obtaining higher marks than the maximum score obtained by any female, then the unbiased selection process will select only male candidates. But this unbiased selection process may not be fair, because it is showing discrimination against females.

To bolster the arguments we are making here, we take examples from the sports domain. In the Olympics or any International Sports Championship, “Male” and “Female” participants participate in different events. So, every event is divided into two sections - “Male” and “Female”. Thus the evaluation process is not biased on “Gender”. We may consider a similar approach in case of hiring process also. Another type of application is - prediction systems. When it comes to bias in a prediction system, there can be two scenarios -

- Where actual ground truth is present, so prediction can be verified. Every year, the New York City Police Department releases the “Stop, Question and Frisk Data” [Nyp]. This dataset contains all the records where New York police suspect a person criminal based on certain features and investigate that person. So, there are two steps - first is to suspect a person criminal, that is prediction and then examining the person to verify the prediction. If any kind of bias is there in the first prediction step (Gender - Male/Female, Race - Black/White), that can be evaluated in the second step. Because we have the information whether the suspected criminal was actual criminal or not.
- Here actual truth is not present, so if class labels are biased, there is no way to verify the prediction. But we can apply Fair-SMOTE that can find out ambiguous data points showing bias and remove those to reduce bias in model prediction.

So, before testing a software system for bias or before starting to mitigate bias, it is essential to understand the context first.

10.3 Future Direction

Section 2.1 describes prior works in machine learning fairness. There are many fairness testing frameworks available and also algorithms for bias mitigation. Improved algorithms will continue coming in the future. But will that be sufficient? May be not. We need algorithms that are actionable and real-life applicable.

The above mentioned Microsoft study found out a common pipeline structure of ML software projects. Figure 10.1 shows the nine stages of that pipeline. We see there is no stage dedicated to fairness testing and mitigation. That means it is still not a standard industry practice to do fairness testing and mitigation before production release. We all know how important that step is, still it is missing. We can not continue like that. Section 2.1 shows many examples of ML software making unethical decisions in real-life and generating serious public objection. In most of the cases, the particular organization had to discontinue their production server because of legal accusations. These kinds of events not only cause revenue deficit but also affect the reputation of an organization in long term. That said, it is time to make fairness testing and mitigation a standard ML software practice. We propose a vision to achieve that.

Our plan is to do incremental hyperparameter optimization. An industry standard machine learner is updated/re-trained multiple times with a large amount of training samples. When a model is updated, the parameters of the model also need an update. In this study, we have shown how our FAIR_FLASH optimizer can find optimal parameters of a ML model by optimizing 4 goals simultaneously (2 fairness goals and 2 performance goals). Our idea is to include an extended version of FAIR_FLASH in the pipeline mentioned in Figure 10.1. We have already established the reason behind optimization for both fairness and performance in chapter 4. But choosing the appropriate fairness metrics is the key factor that will ensure the desired kind of bias getting reduced. In chapter 8, we have shown that there are more than fifty different fairness metrics. But that huge number can be reduced to a very small set of clusters. We have given mathematical/formal validation of that clustering results. We believe practitioners/stake-holders can choose a few metrics from those clusters depending on the domain. The choice of metrics can vary from time to time. Before model deployment, ML model needs to be optimized based on the chosen fairness and performance metrics. If

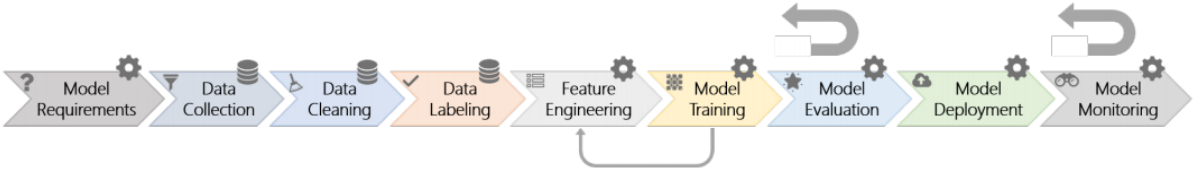

Figure 10.1 Amershi et al. conducted a large study on AI-based applications inside Microsoft Corporation [Ame19]. They found out most of the ML software projects follow a pipeline containing nine steps. First three stages are data-oriented - 1) data collection, 2) cleaning, & 3) labelling and others are model-oriented - 4) model requirements, 5) feature engineering, 6) model training, 7) evaluation, 8) deployment, & 9) monitoring.

that is doable then we do not need separate fairness testing and mitigation stages. Hyperparameter optimization will select the set of optimal parameters to achieve fair prediction results.

We are very fortunate to be one of the pioneers of software fairness research. We hope this study will bolster the significance of fairness testing and mitigation in real life ML software applications. We have shared our vision that fairness will become an integral part of the ML software pipeline. We hope more and more software engineering researchers will work on the fairness domain and improve on our frameworks.

BIBLIOGRAPHY

[Agg19] Aggarwal, A. et al. "Black Box Fairness Testing of Machine Learning Models". *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2019. Tallinn, Estonia: ACM, 2019, pp. 625–635.

[Agr20] Agrawal, A. et al. "Debiasing classifiers: is reality at variance with expectation?" *Available at SSRN 3711681* (2020).

[Aifa] "AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias" (2018).

[al.20] al., S. D. et. "Fairness Measures for Machine Learning in Finance". *AWS Cloud* (2020).

[Alo11] Alonso, O. & Baeza-Yates, R. "Design and Implementation of Relevance Assessments Using Crowdsourcing". *Proceedings of the 33rd European Conference on Advances in Information Retrieval*. ECIR'11. Dublin, Ireland: Springer-Verlag, 2011, 153–164.

[Amaa] "Amazon just showed us that 'unbiased' algorithms can be inadvertently racist" ().

[Amab] "Amazon just showed us that 'unbiased' algorithms can be inadvertently racist" (2016).

[Amac] "Amazon Mechanical Turk" (2021).

[Amad] "Amazon scraps secret AI recruiting tool that showed bias against women" (2018).

[Ame19] Amershi, S. et al. "Software Engineering for Machine Learning: A Case Study". *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 2019, pp. 291–300.

[Ang] Angell, R. et al. "Themis: Automatically Testing Software for Discrimination". ESEC/FSE 18. Lake Buena Vista, FL, USA.

[Ban] "Bank Marketing UCI" (2017).

[Bav14] Bavota, G. et al. "Recommending refactorings based on team co-maintenance patterns". *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*. 2014, pp. 337–342.

[Bel18] Bellamy, R. et al. "AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias" (2018).

[Bel19] Bellamy, R. K. et al. "AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias". *IBM Journal of Research and Development* **63**.4/5 (2019), pp. 4–1.

[Ber19] Bera, S. K. et al. "Fair algorithms for clustering". *arXiv preprint arXiv:1901.02393* (2019).

[Ber17a] Berk, R. et al. *A Convex Framework for Fair Regression*. 2017. arXiv: 1706.02409 [cs.LG].

[Ber17b] Berk, R. et al. *Fairness in Criminal Justice Risk Assessments: The State of the Art*. 2017. arXiv: 1703.09207 [stat.ML].

[Ber17c] Berk, R. et al. "Fairness in Criminal Justice Risk Assessments: The State of the Art". *Sociological Methods & Research* (2017).

[Beu19] Beutel, A. et al. *Putting Fairness Principles into Practice: Challenges, Metrics, and Improvements*. 2019. arXiv: 1901.04562 [cs.LG].

[Bia16] Bianco, S. et al. "CURL: Image Classification using co-training and Unsupervised Representation Learning". *Computer Vision and Image Understanding* **145** (2016). Light Field for Computer Vision, pp. 15–29.

[Bin19] Binns, R. *On the Apparent Conflict Between Individual and Group Fairness*. 2019. arXiv: 1912.06883 [cs.LG].

[Bis20] Biswas, S. & Rajan, H. "Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness". *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (2020).

[Blo17] Blodgett, S. L. & O'Connor, B. *Racial Disparity in Natural Language Processing: A Case Study of Social Media African-American English*. 2017. arXiv: 1707.00061 [cs.CY].

[Blu98] Blum, A. & Mitchell, T. "Combining Labeled and Unlabeled Data with Co-Training". *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. COLT' 98. Madison, Wisconsin, USA: Association for Computing Machinery, 1998, 92–100.

[Bol89] Bollen, K. A. "Structural Equation Models with Observed Variables". *Structural Equations with Latent Variables*. John Wiley Sons, Ltd, 1989. Chap. Four, pp. 80–150. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118619179.ch4.

[Bru20] Brun, Y. "Preventing Undesirable Behavior of Intelligent Machines (ICSSP and ICGSE 2020 Keynote)" (2020).

[Bru18a] Brun, Y. & Meliou, A. "Software fairness". *ESEC/FSE 2018*. 2018.

[Bru18b] Brun, Y. & Meliou, A. "Software Fairness". *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2018. Lake Buena Vista, FL, USA: Association for Computing Machinery, 2018, 754–759.

[Buj18] Bujang, M. A. et al. "Sample Size Guidelines for Logistic Regression from Observational Studies with Large Population: Emphasis on the Accuracy Between Statistics and Parameters Based on Real Life Clinical Data". *The Malaysian Journal of Medical Sciences : MJMS* **25** (2018), pp. 122 –130.

[Buo18] Buolamwini, J. & Gebru, T. “Gender shades: Intersectional accuracy disparities in commercial gender classification”. *Conference on fairness, accountability and transparency*. PMLR. 2018, pp. 77–91.

[Cal10] Calders, T. & Verwer, S. “Three Naive Bayes Approaches for Discrimination-Free Classification”. *Data Min. Knowl. Discov.* **21**.2 (2010), 277–292.

[Cal17a] Caliskan, A. et al. “Semantics derived automatically from language corpora contain human-like biases”. *Science* **356**.6334 (2017), pp. 183–186. eprint: `https://science.sciencemag.org/content/356/6334/183.full.pdf`.

[Cal17b] Calmon, F. et al. “Optimized Pre-Processing for Discrimination Prevention”. *Advances in Neural Information Processing Systems 30*. Ed. by Guyon, I. et al. Curran Associates, Inc., 2017, pp. 3992–4001.

[Cal17c] Calmon, F. P. et al. “Optimized pre-processing for discrimination prevention”. *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 3995–4004.

[Cel19] Celis, L. E. & Keswani, V. “Improved adversarial learning for fair classification”. *arXiv preprint arXiv:1901.10443* (2019).

[Cel18] Celis, L. E. et al. *Ranking with Fairness Constraints*. 2018. arXiv: `1704.06840 [cs.DS]`.

[Cel20] Celis, L. E. et al. *Classification with Fairness Constraints: A Meta-Algorithm with Provable Guarantees*. 2020. arXiv: `1806.06055 [cs.LG]`.

[Ces19] Cesaro, J. & Cozman, F. G. “Measuring Unfairness Through Game-Theoretic Interpretability”. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019, pp. 253–264.

[Cha20a] Chakraborty, J. “Fairway” (2020).

[Cha] Chakraborty, J. & Menzies, T. “Software Engineering for Fairness”. *ACM/IEEE International Conference on Automated Software Engineering - ASE 2019* ().

[Cha19] Chakraborty, J. et al. *Software Engineering for Fairness: A Case Study with Hyperparameter Optimization*. 2019. arXiv: `1905.05786 [cs.SE]`.

[Cha20b] Chakraborty, J. et al. “Fairway: A Way to Build Fair ML Software”. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020. Virtual Event, USA: Association for Computing Machinery, 2020, 654–665.

[Cha20c] Chakraborty, J. et al. “Making Fair ML Software Using Trustworthy Explanation”. *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. ASE ’20. Virtual Event, Australia: Association for Computing Machinery, 2020, 1229–1233.

[Cha21] Chakraborty, J. et al. “Bias in Machine Learning Software: Why? How? What to Do?” *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2021. Athens, Greece: Association for Computing Machinery, 2021, 429–440.

[Cha22] Chakraborty, J. et al. *Fair-SSL: Building fair ML Software with less data*. 2022. arXiv: 2111.02038 [cs.SE].

[Cha02] Chawla, N. V. et al. “SMOTE: Synthetic Minority Over-sampling Technique”. *Journal of Artificial Intelligence Research* **16** (2002), 321–357.

[Che19a] Chen, D. et al. “Replication can improve prior results: A github study of pull request acceptance”. *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. IEEE. 2019, pp. 179–190.

[Che18] Chen, I. et al. *Why Is My Classifier Discriminatory?* 2018. arXiv: 1805.12002 [stat.ML].

[Che19b] Chen, J. et al. “Predicting Breakdowns in Cloud Services (with SPIKE)”. *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2019. Tallinn, Estonia: Association for Computing Machinery, 2019, 916–924.

[Che17] Chen, T.-H. et al. “Analytics-driven load testing: An industrial experience report on load testing of large-scale systems”. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. IEEE. 2017, pp. 243–252.

[CD18] Corbett-Davies, S. & Goel, S. *The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning*. 2018. arXiv: 1808.00023 [cs.CY].

[Cot19] Cotter, A. et al. “Optimization with Non-Differentiable Constraints with Applications to Fairness, Recall, Churn, and Other Goals.” *Journal of Machine Learning Research* **20**.172 (2019), pp. 1–59.

[Dan17] Dann, D. et al. “Reconstructing the giant: Automating the categorization of scientific articles with deep learning techniques”. *Proceedings der 13. Internationalen Tagung Wirtschaftsinformatik* (2017), pp. 1538–1549.

[Das20] Das, S. & Donini, M. *Fairness Measures for Machine Learning in Finance*. 2020. eprint: https://pages.awscloud.com/rs/112-TZM-766/images/Fairness.Measures.for.Machine.Learning.in.Finance.pdf.

[DAV92] DAVIS, R. H. et al. “Machine-learning algorithms for credit-card applications”. *IMA Journal of Management Mathematics* **4**.1 (1992), pp. 43–51. eprint: https://academic.oup.com/imaman/article-pdf/4/1/43/6764690/4-1-43.pdf.

[Deb02] Deb, K. et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. *IEEE Transactions on Evolutionary Computation* **6**.2 (2002), pp. 182–197.

[D'h05] D'haeseleer, P. "How does gene expression clustering work?" *Nature biotechnology* **23**.12 (2005), pp. 1499–1501.

[Dic01] Dickinson, W. et al. "Finding failures by cluster analysis of execution profiles". *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*. IEEE. 2001, pp. 339–348.

[Do08] Do, J. H., Choi, D, et al. "Clustering approaches to identifying gene expression patterns from DNA microarray data". *Molecules and cells* **25**.2 (2008), p. 279.

[Dou18] Douzas, G. et al. "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE". *Information Sciences* **465** (2018), 1–20.

[Skla] "Effect of varying threshold for self-training" (2021).

[Iee] *Ethically-Aligned Design: A Vision for Prioritizing Human Well-Begin with Autonomous and Intelligence Systems*. 2019.

[Eu] *Ethics Guidelines for Trustworthy Artificial Intelligence*. 2018.

[Exp] "EXPLAIN 2019" (2019).

[Faia] "Facebook says it has a tool to detect bias in its artificial intelligence" (2018).

[Faib] *Fairlearn*. 2021.

[Aifb] "Fairness Metrics" (2020).

[Faic] "FAIRWARE 2018:International Workshop on Software Fairness" (2018).

[Fat] "FATE: Fairness, Accountability, Transparency, and Ethics in AI" (2018).

[Fed17] Fedden, L. *The No Free Lunch Theorem (or why you can't have your cake and eat it)*. 2017.

[Fel15] Feldman, M. et al. *Certifying and removing disparate impact*. 2015. arXiv: `1412.3756 [stat.ML]`.

[Fok11] Fokaefs, M. et al. "JDeodorant: identification and application of extract class refactorings". *2011 33rd International Conference on Software Engineering (ICSE)*. IEEE. 2011, pp. 1037–1039.

[Fou19] Foulds, J. et al. "Differential Fairness". 2019.

[Fri19] Friedler, S. A. et al. "A comparative study of fairness-enhancing interventions in machine learning". *Proceedings of the conference on fairness, accountability, and transparency*. 2019, pp. 329–338.

[Gal17] Galhotra, S. et al. "Fairness testing: testing software for discrimination". *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017* (2017).

[Gal98] Galles, D. & Pearl, J. “An Axiomatic Characterization of Causal Counterfactuals”. *Foundations of Science* **3** (1998), pp. 151–182.

[Geb] Gebru, T. *“Race and Gender*.

[Ger18] German, D. M. et al. ““Was My Contribution Fairly Reviewed?”: A Framework to Study the Perception of Fairness in Modern Code Reviews”. *Proceedings of the 40th International Conference on Software Engineering*. ICSE ’18. Gothenburg, Sweden: Association for Computing Machinery, 2018, 523–534.

[Gho15] Ghotra, B. et al. “Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models”. *2015 IEEE/ACM 37th IEEE ICSE*. Vol. 1. 2015.

[Goo] *Google Cloud Pricing*. 2021.

[Gra19] Grari, V. et al. “Fair adversarial gradient tree boosting”. *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2019, pp. 1060–1065.

[Gre19] Green, B. & Chen, Y. “Disparate Interactions: An Algorithm-in-the-Loop Analysis of Fairness in Risk Assessments”. *Proceedings of the Conference on Fairness, Accountability, and Transparency*. FAT* ’19. Atlanta, GA, USA: Association for Computing Machinery, 2019, 90–99.

[Hal05] Halpern, J. Y. & Pearl, J. “Causes and Explanations: A Structural-Model Approach. Part I: Causes”. *The British Journal for the Philosophy of Science* **56** (2005), pp. 843 –887.

[Har16] Hardt, M. et al. *Equality of Opportunity in Supervised Learning*. 2016. arXiv: 1610.02413 [cs.LG].

[Med] “Health care start-up says A.I. can diagnose patients better than humans can, doctors call that ‘dubious’”. *CNBC* (2018).

[Hin18] Hinnefeld, J. H. et al. *Evaluating Fairness Metrics in the Presence of Dataset Bias*. 2018. arXiv: 1809.09245 [cs.LG].

[Hir] *HireVue Assessment Tools*. 2020.

[Hol19] Holstein, K. et al. “Improving Fairness in Machine Learning Systems”. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI ’19* (2019).

[Hom] “Home Credit Default Risk” (2017).

[Hos15] Hossen, B. et al. “Methods for evaluating agglomerative hierarchical clustering for gene expression data: A comparative study”. *Computational Biology and Bioinformatics* **3**.6 (2015), pp. 88–94.

[Hyu20] Hyun, M. et al. *Class-Imbalanced Semi-Supervised Learning*. 2020. arXiv: 2002.06815 [cs.LG].

[Aia] *Improving the enrollment process through machine learning*. 2020.

[Imt19] Imtiaz, N. et al. “Investigating the Effects of Gender Bias on GitHub”. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019, pp. 700–711.

[JM19] James Manyika, J. S. & Presten, B. *What Do We Do About the Biases in AI?* 2019.

[Jia19] Jiang, H. & Nachum, O. *Identifying and Correcting Label Bias in Machine Learning*. 2019. arXiv: 1901.04966 [cs.LG].

[Joh20] Johnson, B. et al. *Fairkit, Fairkit, on the Wall, Who's the Fairest of Them All? Supporting Data Scientists in Training Fair Models*. 2020. arXiv: 2012.09951 [cs.LG].

[Jon20] Jones, G. P. et al. “Metrics and methods for a systematic comparison of fairness-aware machine learning algorithms”. *arXiv preprint arXiv:2010.03986* (2020).

[Kal18] Kallus, N. & Zhou, A. *Residual Unfairness in Fair Machine Learning from Prejudiced Data*. 2018. arXiv: 1806.02887 [stat.ML].

[Kam12a] Kamiran, F. & Calders, T. “Data preprocessing techniques for classification without discrimination”. *Knowledge and Information Systems* **33**.1 (2012), pp. 1–33.

[Kam12b] Kamiran, F. & Calders, T. “Data preprocessing techniques for classification without discrimination”. *Knowledge and Information Systems* **33**.1 (2012), pp. 1–33.

[Kam18] Kamiran, F. et al. “Exploiting Reject Option in Classification for Social Discrimination Control”. *Inf. Sci.* (2018).

[Kam12c] Kamishima, T. et al. “Fairness-Aware Classifier with Prejudice Remover Regularizer”. *Machine Learning and Knowledge Discovery in Databases*. Ed. by Flach, P. A. et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 35–50.

[Kea18] Kearns, M. et al. “Preventing fairness gerrymandering: Auditing and learning for subgroup fairness”. *International Conference on Machine Learning*. PMLR. 2018, pp. 2564–2572.

[KB19] Keith Battocchi Eleanor Dillon, M. H. G. L. P. O. M. O. V. S. *EconML: A Python Package for ML-Based Heterogeneous Treatment Effects Estimation*. https://github.com/microsoft/EconML. Version 0.x. 2019.

[Kle16] Kleinberg, J. et al. *Inherent Trade-Offs in the Fair Determination of Risk Scores*. 2016. arXiv: 1609.05807 [cs.LG].

[Koe20] Koenecke, A. et al. “Racial disparities in automated speech recognition”. *Proceedings of the National Academy of Sciences* **117**.14 (2020), pp. 7684–7689. eprint: https://www.pnas.org/content/117/14/7684.full.pdf.

[Kü19] Künzel, S. R. et al. “Metalearners for estimating heterogeneous treatment effects using machine learning”. *Proceedings of the National Academy of Sciences* **116**.10 (2019), 4156–4165.

[Lah19] Lahoti, P. et al. “ifair: Learning individually fair data representations for algorithmic decision making”. *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE. 2019, pp. 1334–1345.

[Lew73] Lewis, D. “Causation”. *Journal of Philosophy* **70**.17 (1973), pp. 556–567.

[Li11] Li, S. et al. “Semi-Supervised Learning for Imbalanced Sentiment Classification”. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*. IJCAI’11. Barcelona, Catalonia, Spain: AAAI Press, 2011, 1826–1831.

[Lin16] Lin, Q. et al. “Log clustering based problem identification for online service systems”. *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. IEEE. 2016, pp. 102–111.

[LV15] Linares-Vásquez, M. et al. “Optimizing energy consumption of guis in android apps: A multi-objective approach”. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 2015, pp. 143–154.

[Liu21] Liu, S. & Vicente, L. N. *Accuracy and Fairness Trade-offs in Machine Learning: A Stochastic Multi-Objective Approach*. 2021. arXiv: 2008.01132 [cs.LG].

[Lof18] Loftus, J. R. et al. *Causal Reasoning for Algorithmic Fairness*. 2018. arXiv: 1805.05859 [cs.AI].

[Log] “Log-linear Models and Conditional Random Fields” (2021).

[Low21] Lowy, A. et al. “FERMI: Fair Empirical Risk Minimization via Exponential Renyi Mutual Information”. *arXiv preprint arXiv:2102.12586* (2021).

[Luo11] Luong, B. T. et al. “K-NN as an Implementation of Situation Testing for Discrimination Discovery and Prevention”. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’11. San Diego, California, USA: Association for Computing Machinery, 2011, 502–510.

[Mac] “Machine Bias”. *www.propublica.org* (2016).

[Pro] “Machine Bias: There’s software used across the country to predict future criminals. And it’s biased against blacks” (2016).

[Maj21] Majumder, S. et al. “Fair Enough: Searching for Sufficient Measures of Fairness”. *CoRR* **abs/2110.13029** (2021). arXiv: 2110.13029.

[Mep] “Medical Expenditure Panel Survey” (2015).

[Mic] *Microsoft AI principles. 2019.* 2019.

[Mit13] Mittas, N. & Angelis, L. “Ranking and Clustering Software Cost Estimation Models through a Multiple Comparisons Algorithm”. *IEEE TSE* **39**.4 (2013).

[Nai18] Nair, V. et al. “Finding Faster Configurations using FLASH”. *TSE* (2018), pp. 1–1.

[Nai17] Nair, V. et al. “Using bad learners to find good configurations”. *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 2017, pp. 257–267.

[Nie20] Nie, X & Wager, S. “Quasi-oracle estimation of heterogeneous treatment effects”. *Biometrika* **108**.2 (2020), pp. 299–319. eprint: https://academic.oup.com/biomet/article-pdf/108/2/299/37938939/asaa076.pdf.

[Nig00] Nigam, K. & Ghani, R. “Understanding the Behavior of Co-training”. *KDD-2000 Workshop on Text Mining* (2000).

[Nob18] Noble, S. U. *Algorithms of oppression*. New York University Press, 2018.

[Nyp] “NYPD:Stop, Question and Frisk Data” (2018).

[Oga16] Ogarrio, J. M. et al. “A Hybrid Causal Search Algorithm for Latent Variable Models”. *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*. Ed. by Antonucci, A. et al. Vol. 52. Proceedings of Machine Learning Research. Lugano, Switzerland: PMLR, 2016, pp. 368–379.

[Wal] “On Orbitz, Mac Users Steered to Pricier Hotels” (2012).

[Pad20] Padh, K. et al. “Addressing Fairness in Classification with a Model-Agnostic Multi-Objective Algorithm”. *arXiv preprint arXiv:2009.04441* (2020).

[Ple17] Pleiss, G. et al. *On Fairness and Calibration*. 2017. arXiv: 1709.02012 [cs.LG].

[Sklb] “Probability calibration” (2021).

[Com] “propublica/compas-analysis” (2015).

[Eus] *PROVING DISCRIMINATION CASES – THE ROLE OF SITUATION TESTING*. 2009.

[Qia18] Qiao, S. et al. *Deep Co-Training for Semi-Supervised Image Recognition*. 2018. arXiv: 1803.05984 [cs.CV].

[Ree15] Reeb, P. D. et al. “Assessing dissimilarity measures for sample-based hierarchical clustering of RNA sequencing data using plasmode datasets”. *PLoS One* **10**.7 (2015), e0132310.

[Rob86] Robins, J. “A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect”. *Mathematical Modelling* **7**.9 (1986), pp. 1393–1512.

[Rod08] Rodrigues, P. P. et al. “Hierarchical clustering of time-series data streams”. *IEEE transactions on knowledge and data engineering* **20**.5 (2008), pp. 615–627.

[ROS83] ROSENBAUM, P. R. & RUBIN, D. B. “The central role of the propensity score in observational studies for causal effects”. *Biometrika* **70**.1 (1983), pp. 41–55. eprint: https://academic.oup.com/biomet/article-pdf/70/1/41/662954/70-1-41.pdf.

[Ros83] Rosenbaum, P. R. & Rubin, D. B. “The central role of the propensity score in observational studies for causal effects”. *Biometrika* **70**.1 (1983), pp. 41–55.

[Rud19] Rudin, C. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. *Nature Machine Intelligence* **1**.5 (2019), pp. 206–215.

[Sar12] Sarasua, C. et al. “CrowdMap: Crowdsourcing Ontology Alignment with Microtasks”. *The Semantic Web – ISWC 2012*. Ed. by Cudré-Mauroux, P. et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 525–541.

[Sat19] Sattigeri, P. et al. “Fairness GAN: Generating datasets with fairness properties using a generative adversarial network”. *IBM Journal of Research and Development* **63**.4/5 (2019), pp. 3–1.

[Sch19] Schelter, S. et al. “Fairprep: Promoting data to a first-class citizen in studies on fairness-enhancing interventions”. *arXiv preprint arXiv:1911.12587* (2019).

[Sen19] Sennaar, K. *Machine Learning for Recruiting and Hiring – 6 Current Applications*. 2019.

[Sha20a] Sharma, A. & Kiciman, E. “DoWhy: An End-to-End Library for Causal Inference”. Causal Data Science Meeting (https://causalscience.org/). 2020.

[Sha20b] Sharma, A. & Kiciman, E. “DoWhy: An End-to-End Library for Causal Inference”. *arXiv preprint arXiv:2011.04216* (2020).

[Sha19] Sharma, A., Kiciman, E., et al. *DoWhy: A Python package for causal inference*. https://github.com/micros
2019.

[Shi15] Shirkhorshidi, A. S. et al. “A comparison study on similarity and dissimilarity measures in clustering continuous data”. *PloS one* **10**.12 (2015), e0144059.

[Sil18] Silberman, M. S. et al. “Responsible research with crowds: pay crowdworkers at least minimum wage”. *Communications of the ACM* **61**.3 (2018), pp. 39–41.

[Sim20] Simons, T. *Addressing issues of fairness and bias in AI*. 2020.

[Usa] *Situation Testing for Employment Discrimination in the United States of America*. 2007.

[Agg] *sklearn.cluster.AgglomerativeClustering*. 2011.

[Spi93] Spirtes, P. et al. *Causation, Prediction, and Search*. Vol. 81. 1993.

[Spi02] Spirtes, P. et al. “Constructing Bayesian Network Models of Gene Expression Networks from Microarray Data”. *Proc. of the Atlantic Symposium on Computational Biology, Genome Information Systems & Technology* (2002).

[Sta14] Stanescu, A. & Caragea, D. “Semi-Supervised Self-training Approaches for Imbalanced Splice Site Datasets”. 2014.

[Sta] “Stanford Hlab” (1953).

[Sto97a] Storn, R. & Price, K. “Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces”. *Journal of Global Optimization* **11** (1997), pp. 341–359.

[Sto97b] Storn, R. & Price, K. V. “Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces”. *Journal of Global Optimization* **11** (1997), pp. 341–359.

[Str16] Strickland, E. “Doc bot preps for the O.R.” *IEEE Spectrum* **53**.6 (2016), pp. 32–60.

[Stu] “Student Performance Data Set” (2014).

[Ski] “Study finds gender and skin-type bias in commercial artificial-intelligence systems” (2018).

[Tat17] Tatman, R. “Gender and Dialect Bias in YouTube’s Automatic Captions”. *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 53–59.

[Fora] “The Algorithm That Beats Your Bank Manager” (2011).

[3ds] *The most costly thing in Machine Learning Algorithms is labeling*. 2021.

[Tho53] Thorndike, R. L. “Who belongs in the family?” *Psychometrika* **18**.4 (1953), pp. 267–276.

[Tol19] Tolan, S. et al. “Why Machine Learning May Lead to Unfairness: Evidence from Risk Assessment for Juvenile Justice in Catalonia”. *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law*. ICAIL ’19. Montreal, QC, Canada: Association for Computing Machinery, 2019, 83–92.

[Tu21a] Tu, H. & Menzies, T. “FRUGAL: Unlocking SSL for Software Analytics”. *arXiv: Machine Learning* (2021).

[Tu18] Tu, H. & Nair, V. “While Tuning is Good, No Tuner is Best”. *FSE SWAN*. 2018.

[Tu20a] Tu, H. et al. *Better Data Labelling with EMBLEM (and how that Impacts Defect Prediction)*. 2020. arXiv: `1905.01719 [cs.SE]`.

[Tu20b] Tu, H. et al. “Better Data Labelling with EMBLEM (and how that Impacts Defect Prediction)”. *IEEE Transactions on Software Engineering* (2020), pp. 1–1.

[Tu21b] Tu, H. et al. “Mining Workflows for Anomalous Data Transfers”. *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 2021, pp. 1–12.

[Adu] “UCI:Adult Data Set” (1994).

[Def] “UCI:Default of credit card clients Data Set” (2016).

[Hea] “UCI:Heart Disease Data Set” (2001).

[Ger] “UCI:Statlog (German Credit Data) Data Set” (2000).

[Ude18] Udeshi, S. et al. “Automated directed fairness testing”. *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE 2018* (2018).

[Val19] Valentim, I. et al. “The Impact of Data Preparation on the Fairness of Software Systems”. *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE. 2019, pp. 391–401.

[Vas20] Vasudevan, S. & Kenthapadi, K. “LiFT”. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020).

[Ver18] Verma, S. & Rubin, J. “Fairness Definitions Explained”. *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*. 2018, pp. 1–7.

[Ver18a] Verma, S. & Rubin, J. “Fairness Definitions Explained”. *Proceedings of the International Workshop on Software Fairness*. FairWare ’18. Gothenburg, Sweden: Association for Computing Machinery, 2018, 1–7.

[Ver18b] Verma, S. & Rubin, J. “Fairness definitions explained”. *2018 ieee/acm international workshop on software fairness (fairware)*. IEEE. 2018, pp. 1–7.

[Wad18] Wadsworth, C. et al. *Achieving Fairness through Adversarial Learning: an Application to Recidivism Prediction*. 2018. arXiv: 1807.00199 [cs.LG].

[Wan19] Wang, H. et al. *An Empirical Study on Learning Fairness Metrics for COMPAS Data with Human Supervision*. 2019. arXiv: 1910.10255 [cs.CY].

[Wan10] Wang, W. & Zhou, Z.-H. “A New Analysis of Co-Training”. *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Haifa, Israel: Omnipress, 2010, 1135–1142.

[Forb] “White Men Account for 72% of Corporate Leadership at 16 of the Fortune 500 Companies” (2017).

[Sev] *Why split data in the ratio 70:30?* 2012.

[Wu16] Wu, D. O. *Network Science and Applications*. Tech. rep. 2016.

[Xia19] Xiang, M. *Human Bias in Machine Learning*. 2019.

[Yan20a] Yan, S. et al. “Fair Class Balancing: Enhancing Model Fairness without Observing Sensitive Attributes”. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 1715–1724.

[Yan20b] Yan, S. et al. “Fair Class Balancing: Enhancing Model Fairness without Observing Sensitive Attributes”. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM ’20. Virtual Event, Ireland: Association for Computing Machinery, 2020, 1715–1724.

[Yar95] Yarowsky, D. “Unsupervised Word Sense Disambiguation Rivaling Supervised Methods”. *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*.

ACL '95. Cambridge, Massachusetts: Association for Computational Linguistics, 1995, 189–196.

[Yu19] Yu, Z. et al. "Improving Vulnerability Inspection Efficiency Using Active Learning". *IEEE Transactions on Software Engineering* (2019), pp. 1–1.

[Yu18] Yu, Z. et al. "Finding better active learners for faster literature reviews". *Empirical Software Engineering* **23**.6 (2018), 3161–3186.

[Yu19] Yu, Z. et al. "TERMINATOR: Better Automated UI Test Case Prioritization". *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2019. Tallinn, Estonia: Association for Computing Machinery, 2019, 883–894.

[Zaf17] Zafar, M. B. et al. "Fairness constraints: Mechanisms for fair classification". *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 962–970.

[Zha18] Zhang, B. H. et al. *Mitigating Unwanted Biases with Adversarial Learning*. 2018. arXiv: 1801.07593 [cs.LG].

[Zha16a] Zhang, F. et al. "Cross-project defect prediction using a connectivity-based unsupervised classifier". *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE. 2016, pp. 309–320.

[Zha16b] Zhang, L. et al. *A causal framework for discovering and removing direct and indirect discrimination*. 2016. arXiv: 1611.07509 [cs.LG].

[Zha16c] Zhang, L. et al. "Situation Testing-Based Discrimination Discovery: A Causal Inference Approach". *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI'16. New York, New York, USA: AAAI Press, 2016, 2718–2724.

[Zha20a] Zhang, P. et al. "White-Box Fairness Testing through Adversarial Sampling". *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. ICSE '20. Seoul, South Korea: Association for Computing Machinery, 2020, 949–960.

[Zha20b] Zhang, T. et al. "Fairness in Semi-supervised Learning: Unlabeled Data Help to Reduce Discrimination". *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.

[Zha17] Zhang, Z.-W. et al. "Label Propagation Based Semi-Supervised Learning for Software Defect Prediction". *Automated Software Engg.* **24**.1 (2017), 47–69.

[Zhe18] Zheng, X. et al. "DAGs with NO TEARS: Continuous Optimization for Structure Learning". *NeurIPS*. 2018.

[Zho04] Zhou, D. et al. "Learning with Local and Global Consistency". *Advances in Neural Information Processing Systems*. Ed. by Thrun, S. et al. Vol. 16. MIT Press, 2004.

[Zhu06] Zhu, X. *Semi-Supervised Learning Literature Survey*. 2006.

[Zhu02] Zhu, X. & Ghahramani, Z. *Learning from Labeled and Unlabeled Data with Label Propagation*. Tech. rep. 2002.