Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Miran Hadziomerovic

Individual Fairness in Machine Learning

Master's thesis in Informatics
Supervisor: Pinar Øzturk
July 2021

NTNU
Norwegian University of
Science and Technology

Miran Hadziomerovic

Individual Fairness in Machine Learning

Master's thesis in Informatics
Supervisor: Pinar Øzturk
July 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Declaration of Authorship

"Much has been written about AI's potential to reflect both the best and the worst of humanity. For example, we have seen AI providing conversation and comfort to the ones who are lonely while, at the same time, we have seen AI engaging in racial discrimination in different aspects of life. As leaders, it is incumbent on all of us to make sure we are building a world in which every individual has an opportunity to thrive."

Andrew Ng

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Abstract

Faculty of Information Technology and Electrical Engineering

Department of Computer Science

Master of Science in Informatics

Individual Fairness in Machine Learning

by Miran HADZIOMEROVIC

This thesis investigates methods of detecting dataset bias and using a case-based reasoning system constructed using automated weight calculations to ensure individual fairness.

Artificial intelligence (AI) systems are being introduced in new facets of society daily. These systems are created to make important decisions about peoples lives in different spheres of life such as court cases, hiring, credit scoring and many more. In the past, these decision-making systems have been shown to pass on human bias to their decision making. These results led to a new field of research: fair AI systems.

Before even considering fairness of an AI algorithm, one needs to consider if the data is unbiased. If this is not the case, even providing a fair algorithm, unfair results are possible. In order to be able to detect dataset bias, we propose a pre-processing method. It relies on feature relevance scoring methods capturing the relevance of each feature on distinguishing between different class labels. Using this relevance, we create an influence percentage for each feature in the dataset, and use this percentage to distinguish if a protected feature is too influential, marking the dataset biased and unusable if fairness is a concern.

After proposing a method for detecting dataset bias, we explore if a case-based reasoning system using automated weight calculations can be used to ensure individual fairness. We build on the case-based reasoning system presented in [Jaiswal and Bach, 2019] by providing improved global similarity metrics. In order to evaluate the fairness of the case-based reasoning system, we created a fairness verifier. The verifier is constructed as a simplified version of the verifier explained in [John, Vijaykeerthy, and Saha, 2020].

The main contribution of this thesis is twofold. Firstly, we successfully propose a pre-processing method for detecting dataset bias. Secondly, we showed that case-based reasoning using automated weight calculations can be used to ensure individual fairness of an AI system.

Preface

This thesis was written during autumn of 2020 and spring of 2021 to fulfill the degree of Master of Science in Informatics at the Department of Computer Science, Faculty of Information Technology and Electrical Engineering at the Norwegian University of Science and Technology. I would like to thank my supervisor Pinar Øzturk for her guidance and encouragement throughout the work with this thesis. Additionally, I would like to thank fellow student striving for a PhD Amar Jaiswal for helpful cooperation and discussions throughout this project. Finally, I would like to thank my family for their support throughout this entire master's degree.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

While more traditional forms of data analysis provide records or summary statistics, machine learning attempts to capture statistical relationships within a dataset. It does so by discovering useful patterns and correlations in the dataset that assist future decision making. The collection of correlations is commonly know as a "model", and it is used to automate the process of classification of individuals or groups, calculating the value of missing variables, or sometimes, even predicting future events. Some current applications that use these types of models are in assisting decision support for applications such as credit scoring of individuals, fraud detection and hiring. [Barocas and Selbst, 2016]

Initially, one might think that there wouldn't exist any unfairness in AI/ML systems since computers don't know how to be unfair. In an ideal scenario, this would be true but, one important point is being missed; who is creating/controlling these computers? - humans!. Humans, with all their biases (both subconsciously and consciously), are inherently passing on their mistakes down to the machines. There are several types of situations where this could be happening. Obviously, the model needs to be programmed unbiasedly in order to have fair results. The issue is that this is usually not enough. Before even getting to the programming stage, the system could be biased because of its initial dataset that is to be used to train the model.

When talking about dataset bias, there are two important terms that need to be focused on. These are protected features and a class variable. Protected features are values that might be considered sensitive information and shouldn't be used to distinguish between individuals. Some examples of protected features are age, gender and ethnicity. A class variable is the value that the system is supposed to predict, based on the selected features, for unseen data points. A class variable can (in classification problems) take only a finite number of values, which are mutually exclusive. These are denoted as class labels.

Which features should be taken as a class variable is not always obvious and should be decided by data miners. Their job is to translate some unstructured problems to more formal questions that can be parsed to computers easily. Data miners need to understand the project objectives and business requirements. Having this knowledge, they should be able to convert it into a data mining problem definition. While for now, systems need these types of subjective input, they sometimes lead to problems being constructed in such a way that induces unfair treatment of individuals based on protected features. When unfair bias is presented within the initial case base and its decision making for class variables, it is inevitable that the system as a whole will present biased outputs, even when the algorithms that are implemented (that use biased case bases) are unbiased and fair. This issue emphasises the importance of unbiased case bases to be able to start evaluating model fairness.

We can use an example of a system created for assisting in hiring decisions. In order to "teach" the system what a desirable employee for a company is, the system requires previous data of people who were considered, interviewed, hired etc. The issue arises from the fact that the person who was tasked with hiring decisions earlier (e.g. hiring manager), could have been biased either subconsciously or consciously, using protected and/or proxy attributes in his/hers decision making. Obviously, the data that he/she created is biased and shouldn't be used to steer future employment decisions.

A difficult part of mitigating bias is actually detecting it. This is the focus of the first experiment.

When considering fairness in AI/ML systems, there is a distinction between two types of fairness': group and individual fairness. Group fairness considers protected demographics groups (e.g. gender or racial groups) and requires parity of some statistical measure in all of these groups. Individual fairness, on the other hand, looks at constraints concerning specific similar pairs of individuals and their fair treatment in comparison. Most of fairness research in the ML field so far has been focused on group fairness. In this thesis, we attempt to bridge this gap by exploring individual fairness. Using definitions from [Dwork et al., 2012] as a foundation, we investigate ways of detecting bias which leads to unfairness before a model is created, as well as, experimenting with using case-based reasoning in order to ensure fair ML models.

1.1 Goals and Research Questions

This thesis has two overarching goals. The first is to:

Goal *Propose a method which detects bias in datasets.*

Having set this goal, we construct research question 1. Answering research question 1 helps us to achieve our goal.

Research Question 1 *How to detect dataset bias?*

After answering Research question 1, we use the method for detecting dataset bias in order to provide an unbiased dataset to the case-based reasoning model we construct. This is where our second goal is constructed:

Goal *Create a case based reasoning systems that ensures individual fairness*

Since scarce research is available on the topic, in order to be able to accomplish the set out goal, we need to answer Research question 2.

Research Question 2 *Can case-based reasoning systems using automated weight calculations be used for ensuring individual fairness?*

1.2 Thesis Structure

This thesis consists of five main parts. In chapter 2 we will present the background theory needed to understand the contents of this thesis. Chapter 3 will present the related work that covers the current research about individual fairness along with a verifier that detects individually unfair models, as well as a case-based reasoning approach that could be used for ensuring individual fairness. Chapter 4 will describe the methods and approaches we use, as well as describe the architectures used in this thesis. In chapter 5 we will describe the experiment plan and show and analyse the results of the experiments. Finally, in chapter 6 we will discuss the results, their affect on answering the research questions as well as conclude the thesis and present our vision for future work.

Chapter 2

Background Theory

While the initial artificial intelligence methods vary to those we use today, the concept of and the interest in artificial intelligence has been growing steadily over time. This has steered engineers towards attempting to solve more and more everyday tasks using artificial intelligence. Increasing hardware capabilities along with the emerging interest amongst individuals allow for inclusion of artificial intelligence in many aspects of society. These systems vary largely - from predicting severe weather and aviation turbulence, detecting various privacy breaches, providing personalized advertisements and, in some cases, making important life decisions [Rudin and Wagstaff, 2014]. This paper will mostly focus on artificial intelligence based decision-making systems.

2.1 AI in Decision-Making Systems

These decision-making systems make decisions that significantly affect peoples' lives in areas such as finances, college admission, employment, receiving medical treatment, or even judicial sentencing. The goal of using these types of systems was to remove human biases from decision making thus, making it more fair. Unfortunately, recent studies indicate this might not be the case. One of the more famous cases of biases in decision-making systems was the "ProPublica vs Northpointe" case regarding the predictive recidivism system "COMPAS" [Angwin et al., 2016].

Correctional Offender Management Profiling for Alternative Sanctions (COMPAS; Northpointe, which rebranded itself to "equivant" in January 2017), has been used to assess more than 1 million offenders since it was developed in 1998. COMPAS is used to predict a defendant's risk of committing a misdemeanor or felony within 2 years of assessment from 137 features about an individual and the individual's past criminal record. Even though none of COMPAS features directly identify an individuals race or ethnicity, a combination of different features could be used to indirectly determine ones' race and lead to racial disparities in predictions. [Dressel and Farid, 2018]

Angwin et al. analyzed the systems data with the predictions and argued that it is racially biased. While the total accuracy numbers were similar for both white and black people - 67% for whites, 63.8% for blacks, when mistakes did occur, they affected different races differently. What worried ProPublica's authors were the false positive and false negative rates [1]. The false positive rate for black defendants was 44.9% while the same rate for white defendants was almost twice as small at just 23.5%. The same pattern could be noticed in the false negative rates where white defendants who did reoffend were predicted not to do so 47.7% of the time, while the black counterparts' rate was almost twice as low at 28%.

Northpointe argued that their system was indeed fair and not bias when using other, more standard, fairness metrics that they considered during development [Dieterich, Mendoza, and Brennan, 2016]. They dissected all the main accusations from ProPublica and argued they are not evidence of bias providing their own technical analysis using different fairness metrics. This back and forth has raised a greater issue in the world of ethical artificial intelligence and automated decision-making systems - how to decide which metric is the "most" fair one and how to satisfy both accuracy and fairness metrics?

2.2 Fairness Metrics

Even though the notion of fairness has been talked about in philosophy for thousands of years, a single definition for fairness has not been agreed upon. This is caused by the terms fairness and justice being difficult to define precisely. Using common sense, one might define fairness as the absence of biases and prejudice towards a group or an individual. The presence of said biases are referred to as discrimination. Discrimination is often used to describe actions towards a person/group because of some inherent or developed traits such as race, gender, age, sexual orientation etc. These types of traits are usually referred to as *protected features* (or attributes) and should not be used while evaluating a person/group. Discrimination of a person/group is possible even when protected features are absent. This is due to *proxy features*. Proxy features encode a certain protected feature with a substantial degree of accuracy. For example, zip code is a widely accepted proxy that can often reveal race, ethnicity, or age. For this reason, proxy features should also be considered when observing possible sources of discrimination.

Since human moral alone is not enough to stop discriminatory practices, governments have decided to enforce laws which prevent different kinds of discrimination. One downside to this approach is that definitions vary from country to country which raises the issue of same systems being categorized differently depending on the country where they are evaluated. In order to evaluate AI systems, a unified mathematical definition of fairness is needed. There are currently many definitions that are attempting to tackle this issue but opinions on the best and most straight forward one vary vastly.

[1]A false positive is an error in binary classification in which a test result incorrectly indicates the presence of a condition such as a disease when the disease is not present, while a false negative is the opposite error where the test result incorrectly fails to indicate the presence of a condition when it is present.

Frequently, fairness definitions are split into two categories: individual fairness and group fairness. Individual fairness definitions have the main goal of treating individuals equally, regardless of group affiliations. Group fairness definitions are focused on group fairness and equality, usually measured by various metrics (e.g. equal false positive or false negative rates). While not all fairness definitions are easily measurable, there are enough possibilities in both categories to be able to choose adequate metrics. [Dwork et al., 2012]

2.2.1 Group Fairness Definitions

The majority of present day literature focus on group (statistical) definition of fairness. Group definitions of fairness solves a small number of protected demographic groups (e.g. racial or gender groups) and requires a parity of some statistical measure in all of these groups. Some of the more popular group metrics include false positive rates, false negative rates, raw positive classification rate, positive predictive value etc (see [Berk et al., 2021] for more examples). This type of fairness is commonly used since it is simple and the definitions can be achieved without any assumptions of on the data. One downside, however, presents itself in the fact that there is no guarantee for individual or subgroup fairness. There is also a possibility that two different group fairness metrics will be in a disagreement on the final result (see [Kleinberg, Mullainathan, and Raghavan, 2016] for proof). [Chouldechova and Roth, 2018]

The following metrics are commonly used when considering group fairness:

- **Statistical Parity** was introduced early in AI literature considering fairness of models. At those times, it was used interchangeably with the term group fairness [Zemel et al., 2013]. The idea of this metric is to ensure that the proportion of members in a protected group receiving positive classification should be equal to the proportion of the full population. The statistical parity metric is defined as the difference between the percentage of people receiving positive classification in the protected and the unprotected group.
- **Disparate Impact** is a metric similar to statistical parity. It also considers positive outcomes of protected and unprotected groups. The difference relies in the calculation. Instead of looking at the difference between percentages of people classified positively in both groups, disparate impact provides the ratio between the probabilities for the two groups [Feldman et al., 2015].
- **Equalized Odds** was presented as an alternate approach to statistical parity. Instead of considering the probabilities of positive outcomes, equalized odds considers the true and false positive rates. Under this metric, an algorithm is regarded to as fair if both protected and unprotected groups have equal true and false positive rates, respectively[Hardt et al., 2016].

2.2.2 Individual Fairness Definitions

Individual fairness definitions are constructed to ask constraints that bind on specific pairs of individuals in contrast to binding on a quantity averaged over groups. [Joseph et al., 2016] propose a definition that states that less qualified individuals should not be favored over more qualified individuals, where quality is defined with respect to the actual underlying label which is unknown to the algorithm.

Another example is presented in [Dwork et al., 2012]. The basic premise of their definition is that similar individuals should be treated similarly where similarity is defined with respect to a task-specific metric that should be determined for each case individually.

Even though these types of definitions can look to be more meaningful than group fairness definitions, there is one drawback. Individual fairness definitions require making assumptions. The approach in [Joseph et al., 2016] requires strong assumptions on the functional form of the relationship between labels and features, while the approach in [Dwork et al., 2012] requires an existing and agreed upon similarity metric which is usually non-trivial to acquire.

Although this drawback makes it more difficult to measure individual fairness in practice, which is evident in the lack of papers covering this topic, exploring individual fairness is an important research agenda that this paper will mostly focus on.

Even though individual fairness is not always easily quantifiable, there is one commonly used individual fairness metric; The Theil Index.

The Theil Index builds on the theory of general entropy indices and calculates if similar individuals are treated in a similar way. A special case of entropy indices (where $\alpha = 1$) for a problem with n observations is defined as follows [Speicher et al., 2018]:

$$TheilIndex = \frac{1}{n}\sum_{i=1}^{n}\frac{b_i}{\mu}\log\frac{b_i}{\mu}$$

where $b_i = \hat{y}_i - y_i + 1$ and $\mu = \frac{\sum_i b_i}{n}$. For more on the Theil Index, see [Speicher et al., 2018].

2.3 Bias Types

When considering fairness in machine learning, it is important to understand the different sources of unfairness. More precisely, where in the machine learning pipeline bias can be introduced, hindering the system and producing unfair results. In order to dissect the different sources of bias, we divide the machine learning pipeline into meaningful steps. In figure 2.1, we present a simplified representation of the machine learning pipeline.

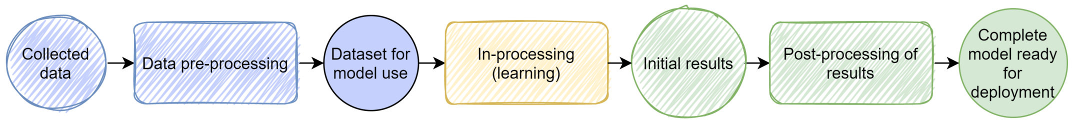

FIGURE 2.1: Machine learning pipeline steps

2.3.1 Data Pre-Processing Bias

The first, and most common, type of bias in a ML system is found in the dataset used for the construction of the ML model. In this case, bias can be present in the data points which are provided by the dataset, or can be induced by faulty handling of features values or labels.

Aggregation bias relates to the different groups of data points (e.g. white people, women) that a ML model should predict decisions for. Aggregation bias occurs when different groups have vital differences that need to be taken into account in the decision making process. If not handled properly, these differences will cause the model to not be able to recognise patterns for these different groups, resulting in bad performance of the decision making system. This type of bias should also be considered and prevented in the in-processing stage of the ML pipeline. One example would be to train and use different models depending on the group that is being provided, in order to more accurately recognise patterns in the data points.

Under-coverage bias occurs when a certain group of data points is under-represented in the dataset used for modeling. This will cause the ML model to perform worse for that group, since it has not had enough data points of that type to be able to learn from.

Feature selection bias relates to the process of feature selection. Feature selection is a vital part of the ML pipeline. The selection should ensure that only the most important features that correlate to the class labels are used. While feature selection affects the accuracy of a ML model, it can also be used to induce unfair decisions. One important consideration that should be discussed is whether a protected feature should be used or left out of the decision making process.

Label bias is one of the commonly mentioned problems appearing in ML models. This type of bias arises from choosing the label or labeling itself. In real-world ML approaches, the label is usually partly or fully unobservable. This property induces the need for a proxy label. A proxy label is used to represent the label as close to the ground truth as possible. Knowing this, bias in label choosing can be observed as a measurement mistake between the ground truth and the proxy label. This type of bias occurs because of bias or misunderstanding from the person who assigned the label.

Data measurement bias occurs in feature values in the dataset. This type of bias is introduced when the values of features are incorrect. This could be because of incorrect measurements or technology limitations. For example, bias in image processing could be introduced by images in the dataset that can not represent the real-world accurately as they are limited by the camera sensors they were captured by.

2.3.2 In-processing bias

Often, in ML literature, it is claimed that the in-processing step (learning) step of the ML pipeline is impossible of creating bias but that its bias is representing unhandled dataset bias. However, it has been shown that bias can still be induced during the in-processing step. This could be down to a variety of reasons including using the wrong learning approach and parameters [Danks and London, 2017].

2.3.3 Post-processing bias

Even when the outputted results of a ML system are not biased, this does not guarantee fairness of the system. The final type of bias occurring in the ML pipeline is the presentation bias. When ML systems are used for decision making, they often require an accompanying user interface to simplify visualizing the results. This user interface is often created by a different person than the one creating the ML model and could be used to portray the results in a different way than what was intended. Regardless of the person creating the user interface, it is important to mention that bias can be introduced by the sole representation of the ML system [Mehrabi et al., 2019].

2.4 Case-Based Reasoning

As mentioned in section 2.1, decision-making systems are used in a variety of different aspects of life. One of the most popular implementations of a decision-making system is by using case-based reasoning.

The premise of CBR is that similar problems/cases have similar results/outcomes. With this in mind, the process of solving new problems in CBR is self-explanatory. In CBR, a new problem is solved by searching through previous cases and retrieving the most similar one. To be able to state that cases are similar to each other a *similarity metric* has to be established. This will be further discussed in section 2.4.2. In other words, in CBR, a new problem is solved by adapting or using old problems.

This type of approach has many benefits. It is very effective in solving problems where there isn't much information about the problem or the problem domain is complex to understand. In figure 2.2 an abstract case-based reasoning system is depicted.

In general, every CBR system can be dissected into four main processes:

1. ***Retrieve***; retrieve the most similar case(s)
2. ***Reuse***; reuse the knowledge of previous case(s) to solve new problem

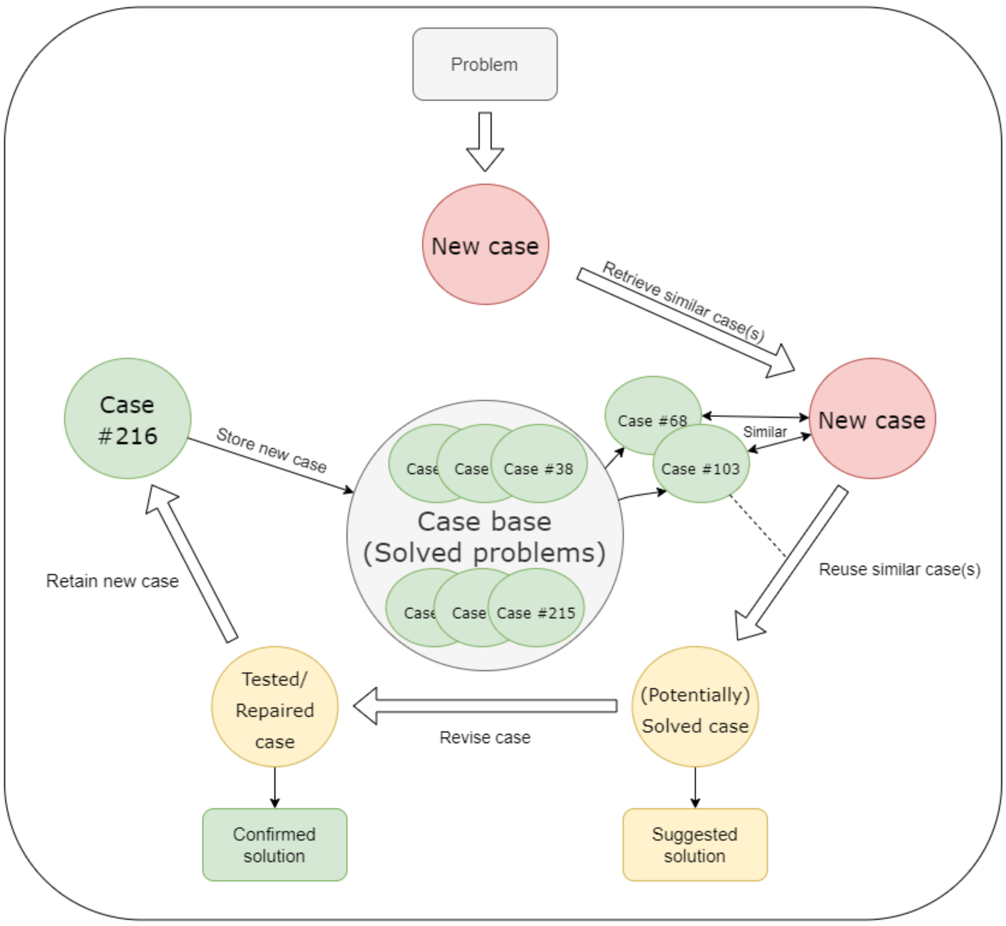

FIGURE 2.2: Case-based reasoning system

3. ***Revise***; revise or test the proposed solution so it fits better
4. ***Retain***; retain the newly solved case and store it for solving of future problems

At the beginning of the process, a problem description moulds a new case. Using this new case, other similar cases are *retrieved* from the previous knowledge (case base), according to some similarity metric. *Reusing* the retrieved case(s) combined with the new case, a problem solution is proposed. The next step is testing and *revising* the proposed solution in order to fit it properly. At the end, a confirmed solution is provided and the newly solved case is *retained* and stored in the previous knowledge section in order to help solve future problems.

In many ways, case-based reasoning (CBR) is fundamentally different from other AI approaches. Rather then relying on general knowledge of a problem domain or learning associations between problem descriptors and conclusions in order to solve a problem, CBR is capable of using specific knowledge acquired by previous experiences and past problems commonly referred to as *cases*. One more difference is that CBR is based on incremental learning since every new solved case can be used with future retrievals. [Aamodt and Plaza, 1994]

In order to construct a CBR model, one needs to decide which features need to be used to distinguish between different labels. This is often not straightforward as raw data contains many features that might be useless. Another issue could be that some features should not be highly important as they are considered protected features. A human can potentially identify protected features incorrectly, resulting in unfair decision being made. These problems introduce the need for an automated approach to feature selection. This could remove any bias that could occur through feature selection. One approach could be by using feature relevance scoring methods.

2.4.1 Feature Relevance Scoring Methods

In recent years, the increasing interest in collecting data has resulted in an exponential growth in provided data both in respect dimensionality and sample rate. Datasets have more features and samples than ever before. While this increase in available data inevitably provides more useful data, the challenge of separating useful data from noise is becoming more challenging. Manual filtering has become unfeasible in most use cases, so more automated approaches using data mining and machine learning techniques is necessary.

Dimensionality reduction is one of the most popular methods in order to remove irrelevant and redundant features. It is usually split into two different subgroups: feature extraction and feature selection. In this paper, focus will be on feature selection. Feature selection methods have a goal of selecting a small subset of features that minimize redundancy and maximize relevance to the target (e.g. class label in classification). [Tang, Alelyani, and Liu, 2014]

In classification problems, feature selection has a goal of selecting highly discriminant[2] features. The relevance of features is assessed by the ability of using a particular feature in distinguishing different classes. If a specific feature f is highly correlated to a specific class c, then the feature f is considered relevant. We argue, this property could be used to detect biased datasets as highly discriminant protected features could be a signal of unfair past classifications.

After a selection of features that are used to construct a model have been selected, the final step of creating a CBR model relates to defining similarity metrics.

2.4.2 Similarity Metrics

The concept of similarity is well known and used in many fields of pure and applied science. The notion of a distance $d(x, y)$ between objects x and y has long been used in various contexts as a quantitative measure of similarity or lack there of. [Finnie and Sun, 2002]

[2]Features that have a high relevance to the outcome.

Similarity is a core concept in CBR since it is used in case retrieval, case adaption and case retaining and storing into the case base [Sun, Finnie, and Weber, 2004]. Similarity, in general, is derived from the similarity of two triangles and two matrices in mathematics.

Definition 2.4.1. *A binary relation S on a non-empty set X is called a similarity relation provided it satisfies the following conditions:*

1. $\forall x, xSx$
2. *If* xSy, *then* ySx ($xSy \implies ySx$)
3. *If* xSy, ySz, *then* xSz ($xSy \wedge ySz \implies xSz$)

The conditions (1), (2) and (3) are the reflexive, symmetric and transitive properties. If xSy, we say that x and y are similar. [Ross and Wright, 1992]

Let's provide an example [Finnie and Sun, 2002]: Let f be a function with domain A and codomain B ($f : A \to B$) and define xSy if $f(x) = f(y)$. Then it implies that S is a similarity relation on A.

From the above mentioned example, we can derive a clear conclusion; If x and y are similar in the sense of S, then x and y have the same solution ergo $f(x) = f(y)$. In other words, it is implied that **similar problems have the same solutions**. This translates well into CBR systems since this property is the premise that CBR systems are built upon. In a CBR system, there are two types of similarities that need to be considered. These are local and global similarities.

Local similarities are used to define the relationships between different values of features i.e. a local similarity states how similar two values of a certain feature are. Defining local similarities is useful in cases where different values of a feature should be similar to a degree and are not completely exclusive. For example, bachelor education might need to be somewhat similar to a masters degree when evaluating a person. However, intuitively, peoples different genders should not have any similarity between the different options and the gender should be considered similar between two feature values only if the gender values are the same. There are several types of local similarities that are used, depending on the type of values a certain feature has. For numeric values, common types local similarity types include using a constant, a polynomial function or similar. For symbolic types, local similarities are usually represented using similarity tables.

While local similarities provide a way of defining relationships between different values for a certain feature, so far there are no defined relationships between different data points i.e. collections of features. This is where global similarities are used.

Global similarities are used to define the relationship between different data points. A global similarity combines local similarities for all the features that are selected in a dataset, and provides a similarity score between different data points. This score is used during the retrieval phase of a CBR model in order to recognize which cases from the case base are the most similar to the case that is being queried. There are several types of global similarities that can be used depending on the type of data points in the dataset. One of the most popular approaches is using a weighted sum.

The **weighted sum** method combines all the selected features by weighing the results of their local similarities. Each of the features that should be used to compare similarity of data points must be assigned a weight value. The weighted sum approach is designed so more important features should be assigned higher weights in order for their local similarity to affect the decision more compared to less important features. The weighted sum method, used as a global similarity function, is depicted in figure 2.1.

$$global(X,Y) = \sum_{i=1}^{n} w_i \cdot local_i(x_i, y_i) \tag{2.1}$$

where X and Y are two data points that are being compared, x_i and y_i are values of feature i, w_i is the defined weight value for feature i, n is the total number of features that are weighted, and $local_i$ is the defined local similarity function for feature i.

For each of the features, a weight needs to be defined in order to calculate the weighted sum. The decision of weights values is not a straightforward one. One approach would be for domain experts to define a weight value for each feature in the dataset. The drawback of using this approach is that it is often difficult to manually quantify importance of features, while it is also time consuming. Another approach is by creating an automated weight calculation method. The latter approach is investigated in this thesis.

Chapter 3

Related Work

In this chapter, we cover two papers that study the field of individual fairness in both a theoretical and practical approach. Finally, we present a framework for a case-based reasoning system which is based on automated weight learning.

3.1 Fairness Through Awareness

Fairness Through Awareness[Dwork et al., 2012] is one of the earliest published papers covering the topic of individual fairness. Dwork et al. defined individual fairness by the principle that two similar individuals should be treated similarly. "We capture fairness by the principle that any two individuals who are similar with respect to a particular task should be classified similarly" [Dwork et al., 2012, p. 214]. This definition is widely considered to be the foundation of any individual fairness discussions. This approach is called similar treatment and stems from ethics and philosophy of law, which require that like cases be treated alike [Binns, 2020].

Dwork et al. suggest that for defining individual fairness for a system two distance metrics need to be established. The first one is a similarity metric. A similarity metric is a distance metric that computes the degree of similarity between two individuals. The second defined distance metric is a distance metric that calculates the difference in the chances two individuals face of obtaining the decision outcomes. In order for an individually fair approach, the difference between the outcomes of the individuals must not be greater than their distance in regard to the similarity metric. The different metrics are visualized in figure 3.1.

In order to more precisely define similar treatment and individual fairness, Dwork et al. propose a function based on Lipschitz mapping. A mapping $M : V \to \Delta(A)$ satisfies the (D, d)-Lipschitz property if $\forall(x, y) \in V$, we have:

$$D(Mx, My) \leq d(x, y) \tag{3.1}$$

where x and y are individuals, V is a set of individuals, M is a function that assigns a probability distribution over the outcomes A. D is a distance metric

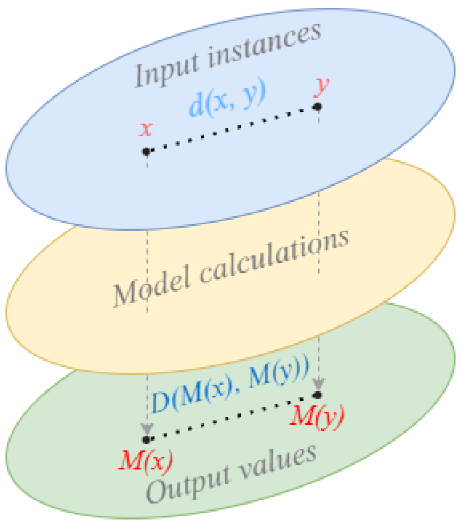

FIGURE 3.1: Visualization of the distance metrics for similarity of individuals and similarity in outcomes

which measures the difference between the outcomes, while d represents the similarity metric i.e. degree of similarity of individuals x and y.

In essence, individual fairness is calculated by measuring similarity of individuals and similarity of outcomes using similarly scaled distance metrics and considering the difference of these two metrics. For example, let us consider algorithm M is a ML model for credit scoring. When M is applied to an individual, it produces a probability an individual should be approved a credit. Suppose two individuals x and y where they are very similar i.e. they have similar traits such as , type of employment, yearly income etc, but x is a male, while y is a female. It is important to understand that gender is not considered relevant when considering credit approval and this difference is not reflected in the similarity metric. For these two individuals, we calculate a small similarity distance $d(x, y) = .02$. When evaluating the results, a discrepancy was noticed. M assigns x a probability of 0.9 of getting approved, while y receives a probability of 0.75. This difference could be caused by the model being trained on historical data where females were discriminated. When we look at the difference between distance of outcomes and the similarity metric $D(Mx, My) = 0.15$, we notice that the distance outcomes are larger. Model M fails to respect the Lipstich constraint, and according to Dwork et al., it is individually unfair. The two individuals are treated dissimilarly, despite being similar.

This example illustrates that irrelevant differences between people should not lead to significant differences in their chance of the desired outcome. One of the difficulties of individual fairness that Dwork et al. mention is defining the appropriate similarity metric since it is based on moral judgement. In the example above, we argued that gender is not relevant when consider credit scoring. This stems from a moral decision that it would be unfair to treat gender as relevant in credit scoring.

While Dwork et al. used the definition of individual fairness to force a fair system, in this thesis, we use it as a basis for investigating fairness of ML models. Since [Dwork et al., 2012] present a more theoretical approach to considering individual fairness, they do not consider specific similarity metrics. One paper that proposes a similarity metric to be used for testing ML models is Verifying Individual Fairness in Machine Learning Models by [John, Vijaykeerthy, and Saha, 2020].

3.2 Verifying Individual Fairness in Machine Learning Models

The goal of this paper [John, Vijaykeerthy, and Saha, 2020] is to construct verifiers that evaluate individual fairness of a given model. In order to achieve this, John, Vijaykeerthy, and Saha leverage on the individual fairness definition from [Dwork et al., 2012]. The paper focuses on tests on linear and kernelized polynomial/radial basis function classifiers and uses publicly available datasets (german credit, adult income, fraud detection, credit islr). This paper considers models for which they have white-box[1] access and which take structured data as input.

The definition of individual fairness, in this paper, is based on the abstract definition from [Dwork et al., 2012]. In [Dwork et al., 2012] a model f is considered fair if for any pair of inputs x, x' which are sufficiently similar - in line with a specified similarity metric ($d(x, x')$ is sufficiently small), model f outputs $f(x), f(x')$ which are also sufficiently similar - in line with a different similarity metric ($D(f(x), f(x'))$ is sufficiently small).

In other words, if two inputs are similar and the outputs are not, this is considered a bias instance and the model f is considered individually unfair. While thorough, [Dwork et al., 2012] is mostly theoretical and hard to implement as is for structured data. To solve this problem, the paper decides to split the features of a dataset into subgroups of features. Each subgroup has a threshold (determined by domain experts) which indicates closeness i.e. if we have two feature values x_i, x_i' , they are considered similar if their absolute difference is less than the threshold ε_j of the subgroup S_j which they are a part of ($|x_i - x_i'| \leq \varepsilon_j$, $x_i, x_i' \in S_j$). Features are separated in subgroups based by humans on domain knowledge. This is also beneficial when considering protected features. The other definition used in this paper proves this. It states that any two valid inputs which differ only on the protected feature(s) must always be put in the same class. This is achieved by assigning the subgroups containing protected features thresholds to ∞ while the non - protected subgroups have 0 as a threshold. All protected features can be put into a single subgroup where the threshold is ∞ (all the protected features are considered equal and irrelevant when considering the similarity of

[1]A white box is a subsystem whose internals can be viewed but usually not altered.

instances) when evaluating fairness. Knowing this, the model is considered fair if a pair of similar inputs x, x' (based on a defined similarity metric) have the same solution in classification models ($f(x) = f(x')$) and close enough solutions in regression models where the absolute difference is considered ($|f(x) - f(x')|$ needs to be sufficiently small).

For the actual verifiers, the focus was on creating sound verifiers but not complete since this is much more cumbersome. A verifier V that is considered sound, for a given model f, outputs NO BIAS, if, and only if, the model f is actually unbiased. The second trait of a verifier is completeness. A verifier V is considered complete will, for a given model f, output a bias instance if, and only if, the model is actually bias. V will also *always* terminate outputting a bias instance or stating that the model has NO BIAS. The steps from the verifiers are shown in figure 3.2.

In this paper, in order to develop the algorithms, the individual bias verification problem was formulated as a (non-convex) optimization problem, and provably-correct global optimization approaches (such as mixed integer linear programming) were used to perform the verification. This was done since the closeness constraints for x, x' ($|x_i - x'_i| \leq \varepsilon_j$, $x_i, x'_i \in S_j$, $\forall i \in S_j$, $\forall j \in [t]$) are linear constraints that are easy to handle in an optimization framework. A model f is considered biased if, and only if, an input pair x^*, x'^* with $|f(x^*) - f(x'^*)| > \delta$ belongs to the set of pairs x, x' described above. Since x and x' are interchangeable, a constraint for a bias instance could be presented as $f(x^*) - f(x'^*) < -\delta$. The optimization problem is therefore constructed as:

$$\begin{aligned} D^* &:= min\ f(x) - f(x') \\ s.t. |x_i - x'_i| &\leq \varepsilon_j, \forall i \in S_j, \forall j \in [t] \\ x_i, x'_i &\in [l_i, u_i] \cap (\mathbb{R}\ or\ \mathbb{Z}, \forall i \in [n] \end{aligned} \tag{3.2}$$

where l_i and u_i are the domain constraints on x_i and x'_i, t is the number of feature subgroups, and n is the number of features.

If the verifier solves the optimization problem with $D^* < -\delta$, then there exists a bias instance that the verifier can output. On the other side, if the verifier finds a certifiable lower bound which implies $D^* \geq -\delta$ then the verifier can output NO BIAS with certainty.

The algorithm steps will be described in the following lines. Firstly, a set V_p of input pairs is constructed such that only the pairs of data points that are "close" in relation to the before created thresholds ($|x_i - x'_i| \leq \varepsilon_j$) are considered. This is mathematically described in equation 3.3.

$$V_p = \{(v,\ v') \mid |v_i - v'_i| \leq \varepsilon_j, \forall i \in D \cap S_j,\ \forall j \in [t]\ \} \tag{3.3}$$

where D is the set of all non-protected features and t is the number of feature subgroups.

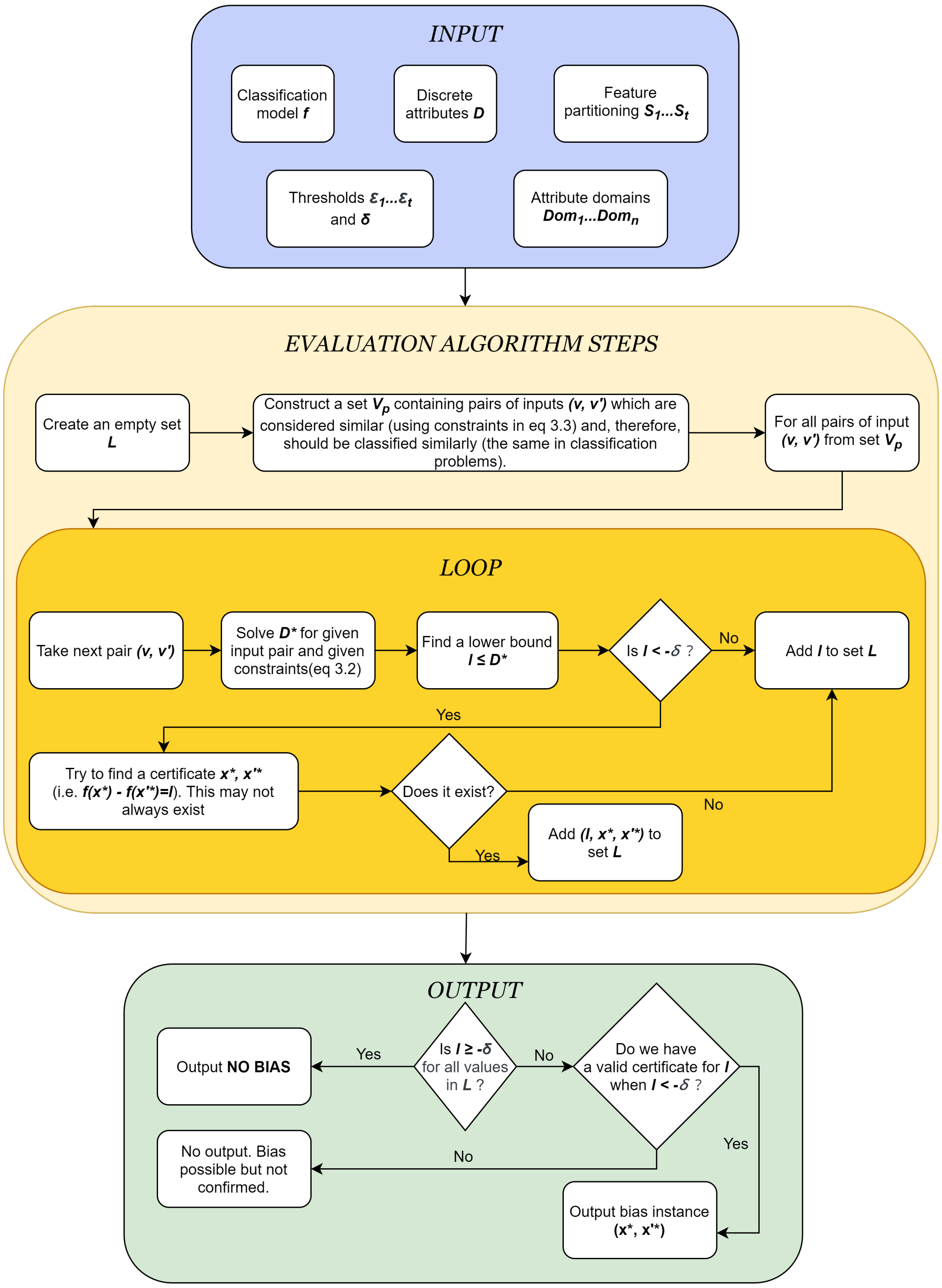
INPUT
Classification model f
Discrete attributes D
Feature partitioning $S_1 ... S_t$
Thresholds $\varepsilon_1 ... \varepsilon_t$ and δ
Attribute domains $Dom_1 ... Dom_n$
EVALUATION ALGORITHM STEPS
Create an empty set L
Construct a set V_p containing pairs of inputs (v, v') which are considered similar (using constraints in eq 3.3) and, therefore, should be classified similarly (the same in classification problems).
For all pairs of input (v, v') from set V_p
LOOP
Take next pair (v, v')
Solve D^* for given input pair and given constraints(eq 3.2)
Find a lower bound $l \leq D^*$
Is $l < -\delta$?
No
Add l to set L
Yes
Try to find a certificate x^*, x'^* (i.e. $f(x^*) - f(x'^*)=l$). This may not always exist
Does it exist?
No
Yes
Add (l, x^*, x'^*) to set L
OUTPUT
Output NO BIAS
Yes
Is $l \geq -\delta$ for all values in L ?
No
Do we have a valid certificate for l when $l < -\delta$?
Yes
No
No output. Bias possible but not confirmed.
Output bias instance (x^*, x'^*)

FIGURE 3.2: The evaluation algorithm of the proposed verifiers

After this, set V_p will contain all the pairs of inputs that are similar and should have a similar outcome. The next step for the verifier is optimization. For all of the pairs x, x' in set V_p, the verifier should attempt to solve D^* from equation 3.2. When a lower bound l is found such that $l \leq D^*$ and $l < -\delta$ the verifier will attempt to certificate this value by attempting to find an instance x^*, x'^* such that $f(x^*) - f(x'^*) = l$. Add the (l, x^*, x'^*) tuple to a set L. After going through all the pairs in set V_p we look through set L. If for all lower bounds l from set L it is true that $l \geq -\delta$ then the algorithm will output NO BIAS. On the other hand, if this is not true (there exist a lower bound l from set L such that $l < -\delta$) the algorithm will output a bias instance (x^*, x'^*).

The minimization process will be different for each of the three considered classification approaches since an optimization procedure for a general model f does not exist. However, this is not important for this thesis since we are using a linear model approach. In this thesis, we took inspiration from the constructed verifiers and created simplified versions of them in our evaluation of architecture for investigating RQ2.

This is the first work that considers the verification of individual fairness for ML models. The testing was conducted such that for each of the three models two cases were considered. In the first one, all features were considered without alterations. In the second case, the protected features were masked by setting the values of each of them to 0. From the results, we can observe that bias instances could possibly occur even in models which have masked protected features. This is possible due to the fact that some non-protected features could be used as proxy features, in order to discriminate individuals (e.g. postal code can be used to determine, in general, which living standard does a person have).

3.3 A Data-Driven Approach for Determining Weights in Global Similarity Functions

This paper covers the topic of case-based reasoning and weight calculations. The goal of [Jaiswal and Bach, 2019]'s paper is to propose a framework for an automated construction of global similarity metrics based on the characteristic of the dataset that is used. In order to get characteristics of a dataset, they propose using feature relevance scoring methods (FRSMs).

Figure 3.3 presents the steps proposed in [Jaiswal and Bach, 2019].

After deciding which dataset is used for the CBR model construction, one needs to decide which feature selection tool and feature relevance scoring methods (FRSMs) are going to be used for scoring the features. For their evaluation, [Jaiswal and Bach, 2019] used Orange [Demšar et al., 2013] as the feature selection tool and Oranges defaults feature relevance scoring methods. After this selection, the final parameter that needs to be decided is the percentage of top features that are assigned ranks in each FRSM, denoted as $prct$. Jaiswal and Bach argue that this percentage is required because often just a small subset of influential features actually needs to be weighted, while the rest of the features, that achieve low scores in the FRSM, don't require weights and should not be used for retrievals.

Since all the input parameters are decided, we can move on to the calculations. Firstly, all features are scored by all the FRSM, as equation 3.4 shows.

$$Features - set\ of\ all\ features$$

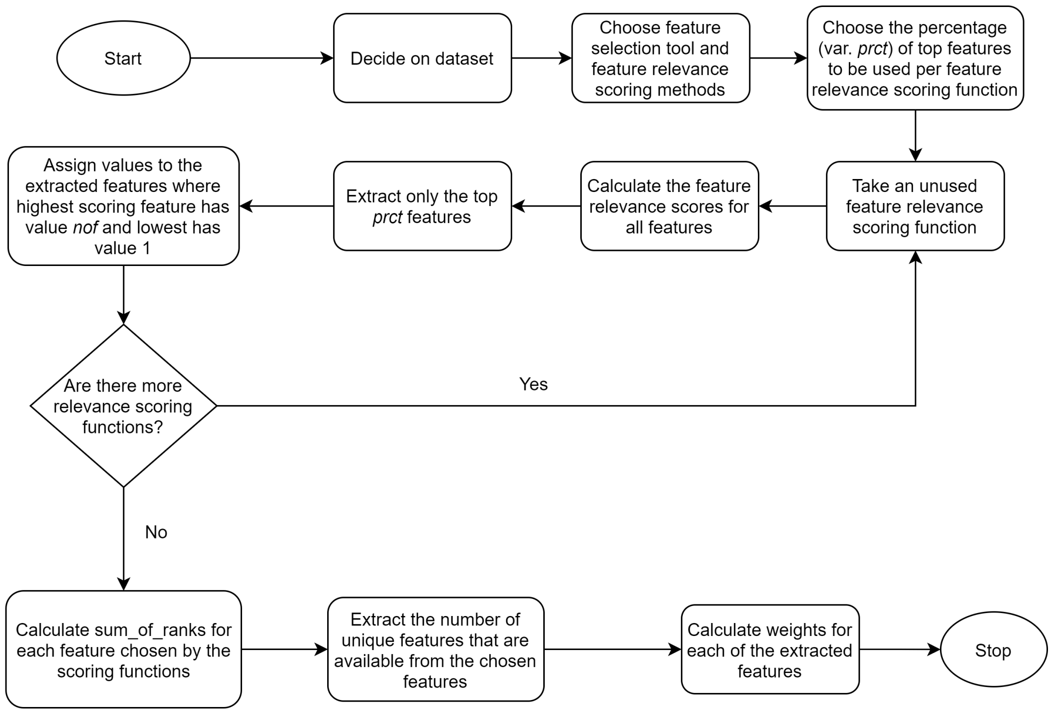
Start
Decide on dataset
Choose feature selection tool and feature relevance scoring methods
Choose the percentage (var. prct) of top features to be used per feature relevance scoring function
Take an unused feature relevance scoring function
Calculate the feature relevance scores for all features
Extract only the top prct features
Assign values to the extracted features where highest scoring feature has value nof and lowest has value 1
Are there more relevance scoring functions?
Yes
No
Calculate sum_of_ranks for each feature chosen by the scoring functions
Extract the number of unique features that are available from the chosen features
Calculate weights for each of the extracted features
Stop

FIGURE 3.3: Proposed steps in order to automatically create weights for each of the features in the chosen dataset

$$
\begin{gathered}
n \;-\; number\ of\ features \\
Scoring\ functions \;-\; set\ of\ all\ scoring\ functions \\
m \;-\; number\ of\ scoring\ functions \\
\lfloor x \rceil \;-\; provides\ the\ closest\ integer\ value\ to\ x \\
sort(A) \;-\; provides\ a\ sorted\ set\ of\ values\ from\ A\ in\ descending\ order \\
number\_of\_selected\_features \;=\; nosf \;=\; \lfloor n * prct \rceil
\end{gathered}
$$

$$\forall f \in Features,\ \forall s \in Scoring\ functions,\ rank(f,\ s) = s(f) \tag{3.4}$$

The next step requires ranking the top *prct* percentage of features. The ranks that are assigned are linear ranks. The highest scoring feature in a certain FRSM receives a value *nosf* as its rank. The rest of the features receive ranks linearly, decreasing by 1 for each next feature. The final feature that is included in the top percentage of features that is ranked, receives a rank of 1. All the features that are not scored in the top *prct* percentage of features, receive a rank 0. These steps are depicted in equations 3.5, 3.6 and 3.7.

$$s_{result} = (rank(f_1),\ rank(f_2), \ldots, rank(f_n)) \tag{3.5}$$

$$s_{result} = sort(s_{result}) \tag{3.6}$$

$$s = \{(f_i,\ y) \mid f_i \in s_{result},\ i \in (1, 2, ...nosf),\ y \in (nosf,\ nosf\ -\ 1,\ ..., 1)\} \tag{3.7}$$

After all the features have ranks in each of the FRSMs, the ranks need to be combined to one number for each feature. This is done by calculating the sum_of_ranks for each of the features. sum_of_ranks presents the sum of ranks in all FRSM for a certain feature. Once the process is repeated for all the features in the dataset, each feature has its own sum_of_ranks. The calculation is portrayed in equations 3.8 and 3.9.

$$S = \bigcup_{i=1}^{m} s_i \mid s \in Scoring\ functions \tag{3.8}$$

$$\forall f \in Features,\ sum\_of\_ranks(f) \ = \ \sum_{i\,=\,1}^{m} y \mid f_i = f,\ (f_i,\ y) \in S \tag{3.9}$$

Finally, having acquired sum_of_ranks for all the features from the dataset, we proceed to the weight calculation. The highest and lowest sum_of_ranks are assigned to variables $max(all\_ranks)$ and $min(all\_ranks)$ respectively. The final step provides the weight calculation formula. This is formula is used on each of the features in order to calculate its weight value. Once all the features have a weight value, the creation of the global similarity metric is completed. The final steps are presented in equations 3.10, 3.11 and 3.12.

$$all\_ranks \ = \ \{sum\_of\_ranks(f) \mid \forall f \in Features\} \tag{3.10}$$

$$N_{unique} = \ number\ of\ unique\ features\ in\ S \tag{3.11}$$

$$\begin{gathered}\forall f \in Features,\\ weight_f = (N_{unique} - 1)\frac{sum\_of\_ranks(f) \ - \ min(all\_ranks)}{max(all\_ranks) \ - \ min(all\_ranks)} + 1\end{gathered} \tag{3.12}$$

In order to evaluate the proposed system, Jaiswal and Bach created a case base for each of the chosen datasets. They also create local similarity metrics using the interquartile ranges for a numerical feature, and pair-wise similarity for a categorical feature. These are not explained further as they are not used in this thesis. The novel approach is compared to a global similarity metric constructed using domain experts and another having equal weights for all the features. The results of the evaluation show that the automated weight learning approach performs better than the equally weighted global similarity metric. When compared to the third global similarity, constructed with the help of domain experts, the proposed automated weight learning approach performs worse three out of the four tested datasets. Jaiswal and Bach argue this result was expected since domain experts have a deep understanding of the system. They mention that the benefits of the automated weight learning approach are that it reduces the time needed for CBR model development and prototyping and it also prevents over-fitting as it does not involve any learning iterations. Another benefit is that this approach minimises the need for human influence, which, in turn, helps remove unnecessary bias.

Considering the benefits of this approach, in this thesis, we use it to construct a CBR model and investigate can it be used to ensure fairness of a CBR model while maintaining high accuracy.

Chapter 4

Method and Architecture

This chapter describes the components of the architecture used to answer RQ1 and RQ2 defined in section 1.1. The chapter describes also, the evaluation method being used to see how the RQs are answered.

RQ1 relates to the pre-processing stage in the ML pipeline and deals with the detection of bias in the dataset. In order to investigate and answer RQ1, we propose an experiment that uses a certain part of [Jaiswal and Bach, 2019]'s framework as a basis. [Jaiswal and Bach, 2019]'s framework was not originally used in the context of bias and fairness, but rather as a data-driven approach to discovering weights in global similarity functions within a CBR system. The framework strongly relies on discovering the relationship between different features and their relevance on changing the class label, in order to calculate weights for each of the features. This work inspired us to propose a method capable of detecting dataset bias by observing the calculated relevance/influence of protected features. Our hypothesis was that protected features, in an unbiased dataset, must have a relatively low influence score compared to the score of the most influential feature. If this is not the case, we argue the dataset contains bias and shouldn't be used for machine learning models. A slight improvement of the feature relevance scoring was implemented and will be explained in this chapter. As [Jaiswal and Bach, 2019]'s framework used datasets lacking protected features they couldn't be used for testing dataset bias. Instead, we used datasets that were publicly available and used previously in different papers covering the topic of fairness in machine learning.

The second part of our work attempts to answer RQ2 by relating to the in-processing stage of the ML pipeline and investigating individual fairness in a CBR system. As mentioned in section 2.2.2, the topic of individual fairness has not been researched extensively. This is due to several reasons but, as [Dwork et al., 2012] note, the difficulty of choosing appropriate distance metrics is one of the main issues. Deciding which instances should be similar (global similarity) or which features values are similar (local similarity) is usually done by domain experts. We propose using the data-driven approach from [Jaiswal and Bach, 2019] to eliminate potential human bias when creating global similarity functions. At the time of writing, only a handful of researchers explored methods of testing individual fairness of machine learning models. We took inspiration to verifying ML models' individual fairness from [John, Vijaykeerthy, and Saha, 2020] in order to investigate the fairness of a CBR system constructed using the data-driven approach from [Jaiswal and Bach, 2019]. Our goal is to show that using the above mentioned data-driven approach we can ensure individual fairness, while requiring less input from domain experts compared to other CBR approaches. The results of the fairness assessment are compared to the results achieved by ML models (using linear regression, kernelized polynomials, RBF kernelized classifiers) mentioned in [John, Vijaykeerthy, and Saha, 2020]. This is done to argue that using the data-driven CBR approach can provide individually fair results while achieving similar or improved accuracy compared to other tested ML models.

The flow chart presented in figure 4.1 shows our architecture for investigating RQ1 and RQ2. The blue part of the chart relates to attempting to answer RQ1, the green part correlates to answering RQ2. Since the first steps in both architectures are shared they are colored in yellow.

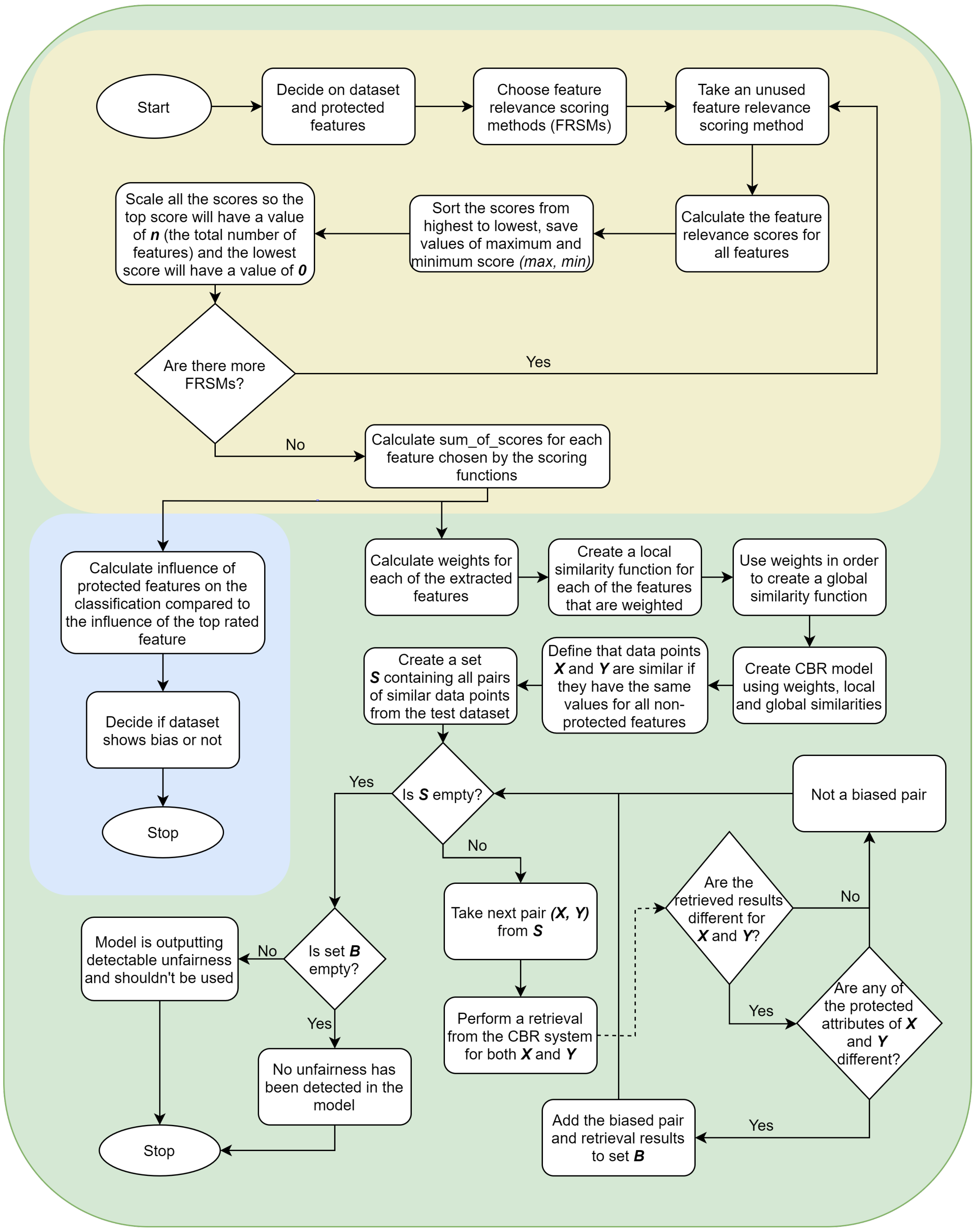
Start
Decide on dataset and protected features
Choose feature relevance scoring methods (FRSMs)
Take an unused feature relevance scoring method
Calculate the feature relevance scores for all features
Sort the scores from highest to lowest, save values of maximum and minimum score (max, min)
Scale all the scores so the top score will have a value of n (the total number of features) and the lowest score will have a value of 0
Are there more FRSMs?
Yes
No
Calculate sum_of_scores for each feature chosen by the scoring functions
Calculate influence of protected features on the classification compared to the influence of the top rated feature
Decide if dataset shows bias or not
Stop
Calculate weights for each of the extracted features
Create a local similarity function for each of the features that are weighted
Use weights in order to create a global similarity function
Create CBR model using weights, local and global similarities
Define that data points X and Y are similar if they have the same values for all non-protected features
Create a set S containing all pairs of similar data points from the test dataset
Is S empty?
Yes
No
Not a biased pair
Take next pair (X, Y) from S
Perform a retrieval from the CBR system for both X and Y
Are the retrieved results different for X and Y?
No
Yes
Are any of the protected attributes of X and Y different?
Yes
Add the biased pair and retrieval results to set B
Is set B empty?
No
Yes
Model is outputting detectable unfairness and shouldn't be used
No unfairness has been detected in the model
Stop

FIGURE 4.1: A flow chart of steps used to answer RQ1 and RQ2. The blue part relates to investigating RQ1, the green part to investigating RQ2, while the yellow part presents the shared steps used in investigating both RQs.

4.1 Detection of Bias in Datasets (RQ1)

Assessment of bias in datasets is a necessary step before testing model fairness of an ML algorithm. If a model is learned using a biased dataset, its results could be unfair while the algorithm itself is not. In order to answer RQ1, we propose a method to be used in the pre-processing stage of machine learning development that could indicate dataset bias.

4.1.1 Method Description

One approach could be to try and detect bias by comparing all the instances in the dataset with each other. This approach would be similar to the approach in [John, Vijaykeerthy, and Saha, 2020] mentioned in section 3.2. The difference is that the goal of their approach is to evaluate the fairness of ML models (i.e. the result of the created models) while our goal is to evaluate bias in the dataset itself. We would compare two data points by looking at all the features that are not protected features or the class label. If they are all the same except the protected features and their class values, we have found a *biased pair* of data points. Figure 4.2 shows examples of data points with their features. It also highlights if a certain pair of data points is similar or/and biased.

This comparison process would be repeated for all possible pairs of data points in the dataset. Having found all the biased pairs, we could simply remove all the biased data points from the dataset. This would guarantee an individually fair dataset. The approach is shown in figure 4.3.

The major drawback of this approach is time complexity. Since we would need to compare every data point of the dataset with every other data point, we will have nested loops resulting in a time complexity of n-squared (quadratic time) $O(n^2)$ in big O notation where n is the number of data points in the dataset. This implies that as the number of data points grows, the number of iterations grows exponentially. For example, if we have a case base with 5,000 data points we would need 25 million iterations to complete the pre-processing step while if we increased the number of data points to 10,000 the number of required iterations would become 100 million iterations.

While we could make some improvements using faster executable programming languages (such as C/C++) this solution is too time consuming to recommend for any dataset of significant size. This is why we propose a different bias detection method in the pre-processing step. This approach has an approximated time complexity of $O(log(n^2))$. The difference in the scale of execution times is shown in figure 4.4.

Our approach to this problem is to use feature relevance scoring methods (FRSMs). Inspired by the framework in [Jaiswal and Bach, 2019], which is explained in section 3.3, we propose a method for automatic identification of how influential each of the protected features are. The influence of each of the

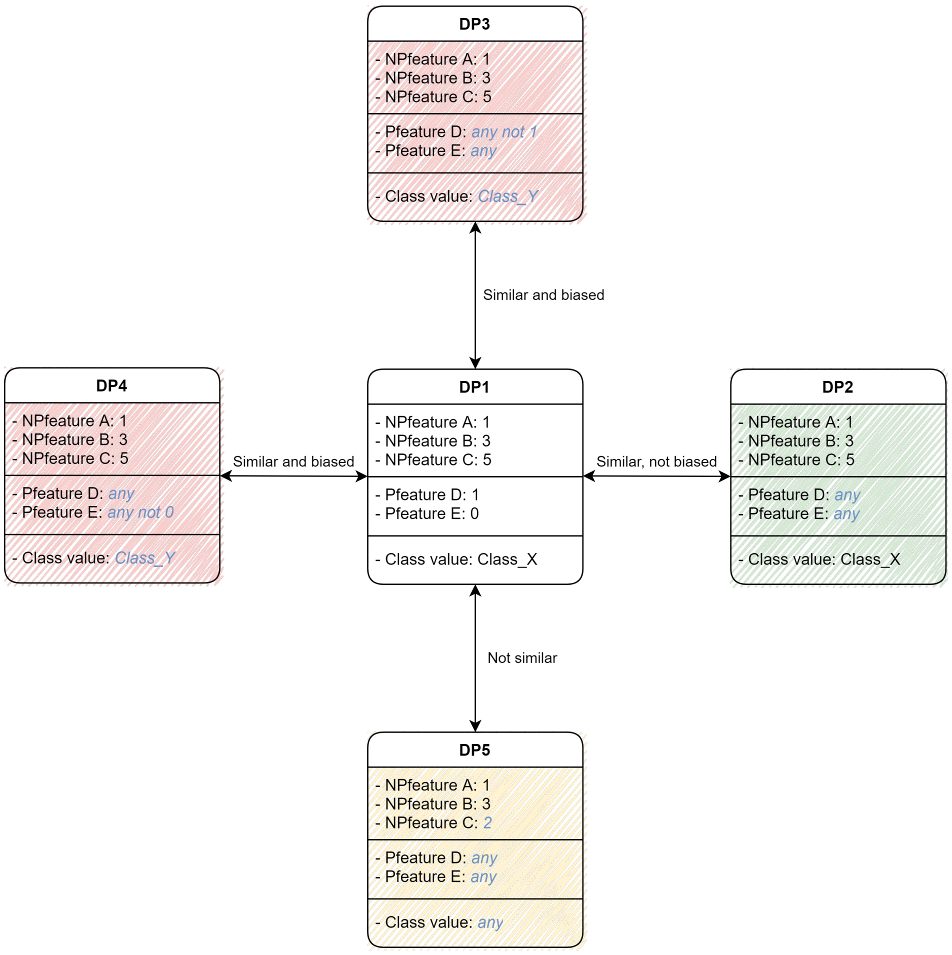

FIGURE 4.2: Comparing data points by their similarity and class values. If two data points have the same non-protected features (i.e. they are similar) but different class values, they are considered a biased pair of data points.

protected features will be quantified using a percentage. This percentage indicates how much influence does a certain protected feature have compared to the most influential feature. Ideally, protected features should have no influence on the class label. In practice, this certainty is difficult to ensure since we can not assume that FRSMs portray relevance perfectly so we propose creating an arbitrary threshold λ.

This threshold's value should be decided by the ML engineer before applying the pre-processing method to the dataset. The goal of having a threshold is to easily quantify and visualize what is the maximum amount of influence a protected feature can have in order for the dataset to be considered unbiased. As a result of this pre-processing method, a ML engineer can decide if the presented influence of protected features is too high (i.e. if the dataset presents bias) and ought to use a different dataset for a desired ML model.

Input for this pre-processing step are the dataset which is being tested and the features that are considered protected. One of the benefits using this approach to identify influence of protected features is that it is automated

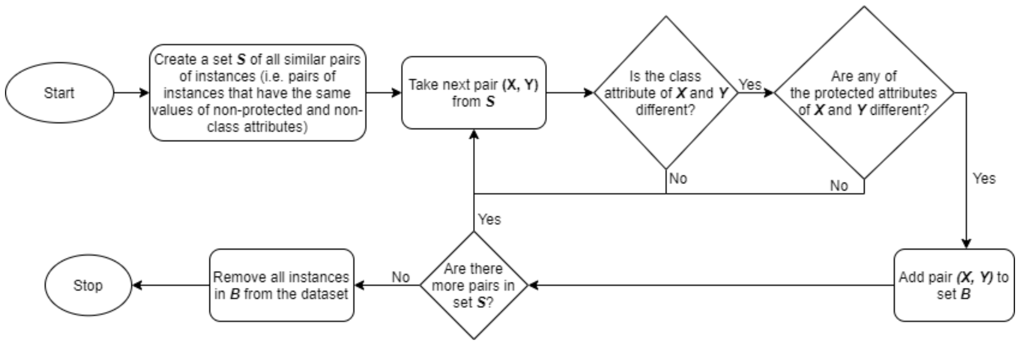
Start
Create a set S of all similar pairs of instances (i.e. pairs of instances that have the same values of non-protected and non-class attributes)
Take next pair (X, Y) from S
Is the class attribute of X and Y different?
Yes
Are any of the protected attributes of X and Y different?
No
No
Yes
Add pair (X, Y) to set B
Are there more pairs in set S?
Yes
No
Remove all instances in B from the dataset
Stop

FIGURE 4.3: Finding and removing all biased instances

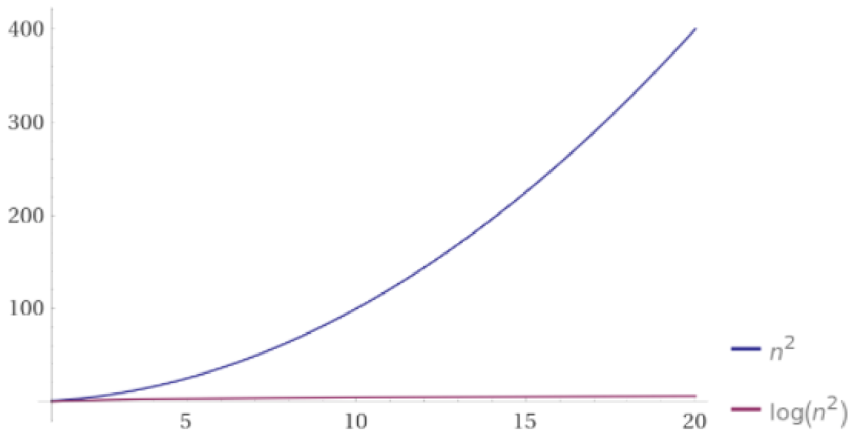

FIGURE 4.4: $O(n^2)$ compared to $O(log(n^2))$

and doesn't require domain experts to assist if the protected features are known/decided. Another benefit explained in this section is time complexity which indicates it can be efficiently used with datasets of any size.

It is important to remind that simply masking protected attributes in a dataset is not a sufficient pre-processing step. This is due to the fact that bias could be engraved in other proxy attributes, as was explained in section 2.2. This is however, outside the scope of this thesis.

4.1.2 Architecture for Answering RQ1

In order to answer RQ1, we rank the features using *feature relevance scoring methods (FRSMs)*. The general idea was introduced in section 4.1.1. Firstly, we explain FRSMs and the feature selection tool. Secondly, we go over to ranking of the features and the slight modifications we did in the relevance scoring approach presented in [Jaiswal and Bach, 2019]. Thirdly, we present the architecture as a whole and how the different parts connect. Finally, the evaluation method is presented.

4.1.3 Feature Relevance Scoring Methods and Feature Selection Tool

As explained in section 2.4.1, the relevance of features is assessed by the ability of a particular feature in distinguishing between class labels. If a specific feature f is highly correlated to a specific class c, then the feature f is considered relevant. This property can be useful when attempting to detect bias in a dataset.

As mentioned earlier, in order to be able to state that there is no bias in a dataset, the protected features of that dataset must have little to no relevance/influence on distinguishing different classes. It is virtually impossible to manually inspect the influence of features in a dataset so an automated approach is required. One automated approach that we propose is using feature relevance scoring methods (FRSMs).

Feature relevance scoring methods involve evaluating the relationship between each input feature and the target variable using statistics. In the end, input features' relationship with the target variable is scored such that a stronger relationship will yield a higher score. The benefit of using these methods for feature relevance scoring is that they are fast and effective. One possible drawback is that different FRSMs use different statistics which, in turn, provide different relationship scores for the same features. We argue that this could be used as a benefit if we combine scores from different FRSMs to calculate a unique value representing the strength of the relationship of an input feature and the target variable.

In order to use the scores from FRSMs, they need to be implemented and applied to a dataset. One might implement FRSMs manually using some programming language. While this approach is possible, it would take more time and could produce unexpected errors. To solve this issue, we propose using an open source data mining tool - Orange [Demšar et al., 2013]. Orange is designed to simplify assembly of data analysis workflows and crafting of data mining approaches from a combination of existing components. It has built in tools for many data problems, including FRSMs. Orange allows for various ML pipeline steps to be done in the tool itself using an intuitive GUI. One evident drawback of Orange is collaboration. For any large scale projects where multiple people are working and collaborating, Orange will not be an optimal solution. Yet, in our case of researching and small scale experimenting, it suits well.

Orange provides around ten different FRSMs. Different FRSMs will be available depending on the type of feature values that are provided. The choice of FRSMs in our experiments is explained in section 5.2.

4.1.4 Feature Selection and Influence/Relevance Rankings

The creation of the architecture, used to answer RQ1, is leveraging on work from [Jaiswal and Bach, 2019]. Their data-driven approach to weight calculation is based on FRSMs capturing the amount of relevance each feature has in distinguishing different classes. Our approach uses the same principle but for a different purpose. We use FRSMs ability to present relevance of features in order to consider the amount of influence protected features have. The result of our pre-processing method provides a quantifiable influence score that should alert an engineer if a datasets protected features influence classifying data points more than they should. By combining different FRSMs to a single score value, we argue that our approach presents a valuable pre-processing method for any people working with potentially sensitive data.

After deciding on the dataset and it's protected features, they are given as input to the FRSMs. Each of the FRSMs scores each of the features' relevance in distinguishing different classes. As mentioned earlier, different FRSMs use different statistics in order to score features. This leads to different FRSMs having completely different scales of scores. For example, during testing, one of the FRSMs ranged scores from 0.26 to 0.04 for different features, while another FRSM ranged the scores between 73 and 6. Since we are combining scores of various FRSMs, this difference would cause an issue where one FRSM would have a larger influence on the final score by simply having a larger range of values. Our solution to this issue is to scale the scores of each FRSM to the same range. This will ensure each FRSM will have the same influence in the final score.

In order to rank the features in each FRSM, [Jaiswal and Bach, 2019]'s approach assigns numeric values on a linear scale as ranks. The highest scoring feature in a FRSM is assigned n as a rank, where n is the number of features that should be scored out of all of the features. The lowest scoring feature that should be ranked will have a rank of one. All the rest of the features that are not to be ranked will have a rank of zero in this specific FRSM. After some testing, we noticed two issues that concern us with this approach.

The first issue arises in using an arbitrary number n to determine how many features are ranked. While this approach has benefits if a dataset has a large amount of features and an engineer wants to rank just some of the most important ones, we argue that it adds unnecessary human interference. It can be difficult to intuitively select a percentage of top features to be ranked in order to provide with optimal results. We decide to change the ranking system so n represents all the features in the dataset. If it were required, using a different n could be implemented with not much change. This will be further explored in section 4.2.

The second, more important issue stems from the linear scale of the ranks values. While a linear ranking might intuitively make sense, we found that this approach doesn't properly preserve the relevance of each feature. This will be easier to portray with an example.

Let's consider four features and one FRSM. Next, consider that feature A scored a 100 for that FRSM, feature B scored a 25, feature C scored a 10, and feature D scored a 4. In the original framework, the ranks for these features would simply be 4 for feature A, 3 for feature B, 2 for feature C and 1 for feature D. These ranks would indicate that feature A is four times more influential compared to feature D, while looking at the scores we see that it's actually twenty five times more influential. A similar pattern could also be noticed when considering feature C. Looking at the ranks, feature A is only twice as influential as feature C, while the scores indicate that feature A is ten times more influential compared to feature C.

The above mentioned example shows that if a certain feature is vastly more influential than the others, linear rankings will not preserve this property. Since our approach relies on preserving the relevance of features, as precisely as possible, a different ranking system must be used. This is the reason we propose the solution of scaling the scores rather than assigning linear ranks. The top scored feature in a FRSM is assigned the aforementioned number n, which now represent the total number of features in the dataset. The lowest scoring feature is assigned zero for this FRSM. All the features in-between are assigned scaled values of their score for that FRSM. This approach is common in the pre-processing pipeline of a ML system and is called normalization or min-max scaling of features[1]. It is used to ensure all value ranges of different features are scaled to be between 0 and 1. The goal of scaling is to prevent a feature with a larger range of values having a bigger influence on the predicted class labels. Instead of using it to scale features, we used the min-max scaling to scale scores of FRSMs.

Let's consider the same example as before, but instead of assigning linear ranks, min-max scaled scores will be provided. Since different FRSMs have vastly different scoring ranges, all FRSMs scores will be scaled to have the same maximum and minimum value. The maximum value of the scaling range is set to the number of features; in this example it will be 4. The minimum value is set to 0. Using this approach, for feature A the new score is 4, for feature B it's 1, for feature C it's 0.4, and for feature D it's 0. As shown in figure 4.5, the relevance of features is retained in the new ranks (scaled scores) more accurately than using the approach in [Jaiswal and Bach, 2019].

Since our approach clearly indicates better relevance score preservation, which is the main goal of the ranks, we use it for answering both RQs. In section 5.2.2, we compare the results using both approaches when calculating influence of features.

[1]Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as min-max scaling. Here, Xmax and Xmin are set, instead of 0 and 1, as the maximum and the minimum values of the feature respectively

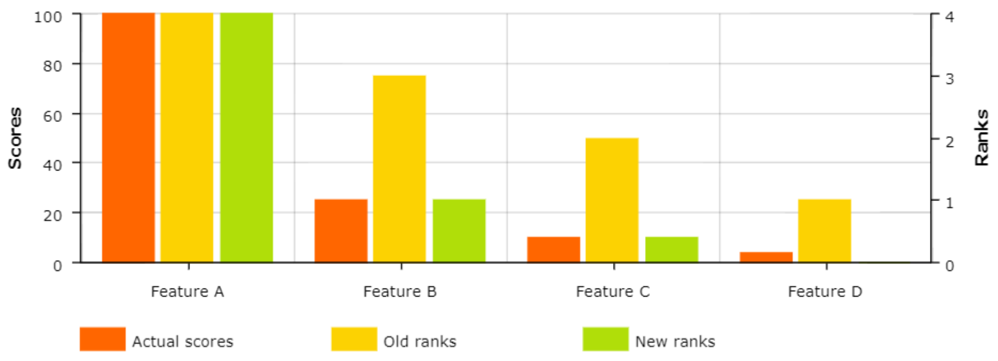

FIGURE 4.5: Old ranks (linear ranking approach from the original framework) vs new ranks (our proposed approach using scaling of FRSMs' scores)

.

4.1.5 Calculating Relevance/Influence of Features

After scaling the scores for each feature in each FRSM, we need to combine all those scores for each individual feature. To achieve this we simply add up the ranks (scaled scores), for a particular feature, assigned according to all the different FRSMs. This step is repeated for all the features in the dataset. At the end, each feature has as a number representing it's relevance to distinguishing different classes. For easier understanding, this number is refereed to as the *sum of scores* of a feature f - $sum\_of\_scores(f)$.

The equations below mathematically depict the steps described in the previous sections.

$$Features - set\ of\ all\ features$$
$$n - number\ of\ features$$
$$Scoring\ methods - set\ of\ all\ scoring\ methods$$
$$m - number\ of\ scoring\ methods$$
$$sort(A) - provides\ a\ sorted\ set\ of\ values\ from\ A\ in\ descending\ order$$

$$\forall s \in Scoring\ methods, \forall f \in Features,\ score(f,\ s) = s(f) \tag{4.1}$$

$$s_{result} = (score(f_1),\ score(f_2), ..., score(f_n))$$
$$s_{result} = sort(s_{result}) \tag{4.2}$$

$$min = min(s_{result}),\ max = max(s_{result}),\ scaled\_min = 0,\ scaled\_max = n$$
$$s = \{(f_i,\ y) \mid f_i \in s_{result},\ i \in (1,2,...n),$$
$$y = (score(f_i) - min) * \frac{scaled\_max - scaled\_min}{max - min} + scaled\_min\} \tag{4.3}$$

$$S = \bigcup_{i=1}^{m} s_i \mid s_i \in Scoring\ methods$$

$$\forall f \in Features,\ sum\_of\_scores(f) \ = \ \sum_{i\,=\,1}^{m} y \mid f_i = f,\ (f_i,\ y) \in S \qquad (4.4)$$

4.1.6 Protected Features' Relevance/Influence

In order to answer RQ1 and to find out whether the dataset is biased or not, we turn to the *sum of scores* of protected features. If the *sum of scores* of a protected feature is too high, the dataset has potential bias and shouldn't be used. The obstacle here is to present the value of *sum of scores* within a context. Without this, the actual value of *sum of scores* is not useful. We decided to use percentages instead of numeric values to present influence of features since percentages are more readable and easily portray context for humans.

We could simply add up the different *sum of scores* for each of the features. Using this number, the degree of influence could be presented by simply dividing the *sum of scores* for that feature and the total sum. In the end, all the percentages would add up to 100%. Even though this approach might intuitively come as the most natural one, we argue it's not the best one. Let's consider an example which depicts a simplified behaviour of a dataset that we encountered. Our dataset has 100 features. This indicates that we have 100 different *sum of scores* as well. Let's say that the total sum of scores is 80,000 and that a protected feature has a *sum of scores* of 700, while a different non-protected feature has a *sum of scores* of 150. This would indicate a percentage influence of 0.87% for the protected feature and 0.18%. On first glance, both are very low and could indicate there is no relevance of these features to the distinguishing of different classes. The problem that arises is that these percentages does not portray context well. In the same dataset, let's consider the feature with the highest *sum of scores* which is 2000. Its percentage influence would be 2.5%. We argue that such small percentages do not provide enough context around the actual influence of features.

We propose to use percentages that relate to the highest *sum of scores*. Instead of considering how influential a feature is in comparison to all the other features, we consider how influential a feature is compared to the top rated feature. This allows us to have context in our percentage values without having to look at other features' scores. If we consider the same example as before, the new calculations will provide a percentage value of 35% for the mentioned protected feature and 7.5% for the non-protected feature. Having been able to observe the actual scores of the features, we argue that this approach achieves the desired effect; portraying influence of features as a percentage without the need to look at other features for context.

4.1.7 Threshold λ

As all the influence percentages are calculated, the final step is to decide how large of an influence a protected feature can have in order for the dataset to still be unbiased. While no fixed threshold is necessary, and percentage values can be interpreted separately, we introduce a threshold λ in order to present the results more visually. Another goal of the threshold is to entice the engineer to consider what threshold needs to be set, before doing the pre-processing step. This will ensure that the threshold isn't skewed and changed after viewing the results.

The value of λ is calculated using a percentage p and the maximum sum of scores. The percentage p represents the maximum percentage amount of influence that a protected feature can have (when compared to the most influential feature) in order for the dataset to be considered unbiased. If a protected features' influence percentage exceeds λ, the dataset contains bias and shouldn't be used for ML models.

For example, let's consider that the most influential feature has a sum_of_scores of 200 and we choose p to be 15%. This means that λ will have a value of 30. If one (or more) of the protected features' sum_of_scores has a value larger than 30, that feature is more influential than it should be and the dataset contains bias.

The equations below mathematically depict the calculation of λ and its role in the architecture.

$$\begin{gathered} max\_sum \; = \; \{\, max\; (sum\_of\_scores(f)) \mid \forall f \in Features\} \\ Protected\; features \; - \; set\; of\; all\; protected\; features \\ p \; - \; the\; max\; influentialilty\; the\; protected\; feature \\ should\; have\; compared\; to\; top\; ranked\; feature \\ \lambda \; - \; p \, * \, max\_sum \\[1ex] A\; dataset\; contains\; bias\; if \\ \exists f \in Protected\; features,\; sum\_of\_scores(f) \; > \; \lambda \end{gathered} \tag{4.5}$$

It is important to reiterate that the threshold λ is introduced for easier visualization of the percentages and has no influence on the percentage calculations.

4.1.8 Evaluation Method

In order to investigate how well the method detects bias in datasets, we propose an evaluation method. The evaluation method is constructed such that we use one of the datasets that has bias detected. After choosing a dataset, we perform removing of biased pairs with the approach presented in 4.1.1. As was mentioned, this approach is only feasible for a dataset with a relatively small number of data points. Once we remove all the biased pairs of data points from the dataset, we provide this modified dataset to the pre-processing method. Finally, we compare the results of the pre-processing method. If the method presents the modified dataset as significantly more fair when compared to the original dataset, we argue this indicates the method is correctly presenting the influence of features and consequently, correctly presenting datasets that contain biased data points. It is important to notice that this type of evaluation should be applied to more datasets in order to have a stronger conviction of the method detecting bias properly. Due to the time and resource constraints, this is not attempted but is advised as a required step for future work.

4.1.9 Pseudocode for Investigation of RQ1

The pseudocode for the pre-processing method used to answer RQ1 can be seen in algorithm 1.

Algorithm 1: Calculating influence of features compared to top rated feature

```
Input: $dataset \leftarrow chosen\ dataset$
Input: $methods \leftarrow all\ the\ selected\ Feature\ Relevance\ Scoring\ Methods$
Input: $protected \leftarrow labels\ of\ features\ considered\ protected$
Output: $Influence\ of\ protected\ features$
$influence \leftarrow calculateProtectedInfluence(dataset,\ methods,\ protected)$
Function calculateProtectedInfluence($dataset, methods, protected$):
    $scores \leftarrow getScores(dataset, methods)$
    $sum_{scores} \leftarrow getSumOfScores(scores)$    // list of features and
     their respective sum of scores in different FRSMs
    $sort(sum_{scores})$    // sort $sum_{scores}$ in descending order
    $bestScored \leftarrow first(sum_{scores})$
    $allInfluences \leftarrow empty\ list$
    for (label, sum) in $sum_{scores}$ do
        if label in protected then
            $influence \leftarrow \frac{sum}{bestScored} \cdot 100$
            $allInfluences.insert(label,\ influence)$
        end
    end
return $allInfluences$
```

As input for the algorithm the dataset, the protected features, and the selected FRSMs are required. The first step is to calculate the scores of all features in each FRSMs. After this, the scores are scaled as was described in section 4.1.4. These steps are shown in algorithms 2 and 3.

Algorithm 2: Calculating scores of features for each of the FRSMs

```
Function getScores(dataset, methods):
    $allScores \leftarrow empty\ list$
    $n \leftarrow total\ number\ of\ features$
    for method in methods do
        $scores \leftarrow empty\ list$
        for featureLabel in dataset do
            $score \leftarrow method(feature)$
            $scores.insert(featureLabel, score)$
        end
        $scores \leftarrow scaleScores(scores, n)$
        $allScores.insert(scores)$
    end
return allScores
```

Algorithm 3: Scaling scores of a FRSM

```
Function scaleScores(scores, n):
    $sort(scores)$                        // sort scores in descending order
    $min \leftarrow min(scores)$                  // minimum score value
    $max \leftarrow max(scores)$                  // maximum score value
    $scaled_{min} \leftarrow 0$                  // new minimum to scale to
    $scaled_{max} \leftarrow n$                  // new maximum to scale to
    for (label, score) in scores do
        $score \leftarrow (score - min) \cdot [\frac{scaled_{max} - scaled_{min}}{max - min}] + scaled_{min}$
    end
return scores
```

After calculating and scaling the scores in each FRSM, we compute the *sum of scores* for each feature in the dataset. This process can be seen in algorithm 4.

Algorithm 4: Calculating sum of scores for all features

```
Function getSumOfScores(scores):
    $sum_{scores} \leftarrow empty\ list$
    for (label, score) in scores do
        if label in $sum_{scores}$ then
            $sum_{scores}[label] \mathrel{+}= score$
        else
            $sum_{scores}.insert(label, score)$
        end
    end
return $sum_{scores}$
```

Finally, using the best ranked features *sum of ranks* we calculate the influence/relevance of protected features. As the output of algorithm 1, the engineer has access to percentile influence values of protected attributes in the dataset. These values are used to determine if the dataset that was used contains bias.

4.2 Evaluation of Fairness of a CBR Model (RQ2)

As shown in figure 4.1, the architecture for answering both RQs has initial similarities. In order to answer RQ2, we use the data-driven approach presented in [Jaiswal and Bach, 2019] to calculate global similarity of a CBR system. Our goal is using this approach, we can ensure individual fairness in a CBR system while also requiring less input by domain experts compared to other CBR approaches. Since the fairness needs to be tested, we took inspiration from [John, Vijaykeerthy, and Saha, 2020] and used a simplified version of their framework to detect possible unfairness.

4.2.1 Method and Architecture for Investigation of RQ2

In order to answer RQ2 we construct a CBR model and test its individual fairness. Firstly, the model construction is discussed and explained. Afterwards, steps in order to test the CBR models individual fairness are presented. We then observe the architecture as a whole and mention how the components connect. At the end, the evaluation method is discussed.

4.2.2 Initial Model Construction

The first parts of the model construction are identical to the steps presented for answering RQ1. Therefore, they haven't been thoroughly explained here again in order to avoid repetition. In the beginning, a dataset needs to be selected. Since the construction of this model has the purpose of testing its fairness afterwards, we need to remove any potential outside bias. This is why an unbiased dataset should be the first priority in model construction. One important difference compared to steps used for the construction of experiment 1 is the need to split the dataset into two parts; a training and test dataset. Since we perform retrievals on the model in order to test fairness as well as measure accuracy, a split is necessary. Only the training dataset will be used as the models case base as well as in the pre-processing steps. The test dataset will be used in the fairness testing phase.

The next step is choosing the feature relevance scoring methods (FRSMs) that will be used for scoring the features, as well as scoring all the features in each FRSM. As before, we use the open source data mining tool Orange for FRSMs choosing and scoring the features. After the scores for all the features have been calculate using a FRSM, we proceed to scaling those scores. The scores require scaling since different FRSMs use different statistics and provide different ranges of values. If FRSMs are not scaled to the same range, some FRSMs will have more influence on the weights, used for global similarity functions, simply based on their higher range of values. We scale scores inside each FRSM such that the highest scored features score is scaled to the value of n where n represents the total number of features that are being scored. At the same time, the lowest scored feature in that FRSM is scaled to zero. All the other features are scaled so they fit between this range, while still retaining their score differences in the new scores. This scaling process is called min-max scaling and is explained in more detail in section 4.1.4. After scaling scores in one FRSM, the process is repeated for all the FRSMs.

When all the scores have been scaled, we proceed to calculating a number for each feature which represents values from all FRSMs - sum of scores. For one feature, all the scores from various FRSMs are added up and thus create the sum of scores of a that feature. This process is repeated for all the features in the dataset. After the sum of scores are calculated for each feature in the dataset, architecture for the second experiment separates from the one explained for the first experiment.

4.2.3 Calculating Weights and Creating Local and Global Similarity Functions

The next step relates to the weight calculation. As mentioned earlier, our CBR system will use weighted sum as the global similarity method. This approach requires us to define a weight value for each feature in the dataset. Using the weighted sum method, weights represent how important a certain feature is when performing retrievals from the case base. The higher the weight value for a certain feature, the more important that feature is. Inspired by the framework in [Jaiswal and Bach, 2019], weight calculation for a feature will be based solely on its relevance scores from FRSMs. The weight calculation is done using the formulas presented below.

$$\begin{aligned}
all\_ranks &= \{sum\_of\_ranks(f) \mid \forall f \in Features\} \\
n &= number\ of\ features\ that\ were\ scored \\
& \qquad \forall f \in Features, \\
weight_f &= (n-1)\frac{sum\_of\_ranks(f)\ -\ min(all\_ranks)}{max(all\_ranks)\ -\ min(all\_ranks)} + 1 \qquad (4.6)
\end{aligned}$$

The fraction in equation 4.6 performs another instance of min-max scaling where value of sum of scores for each feature is scaled between 1 and 0. The feature with the highest sum of scores has 1 as its fractional value, while the one with the lowest sum of scores will have 0 as its fractional value. The fractional value is then multiplied by the number of features that were scored decreased by 1. This is done to ensure that the weights values will range from the number of features all the way down to one. In the end we add one to the calculation. Decreasing *n* by 1 and adding 1 at the end is done to ensure that all the features will be weighted. If this step was removed, some of the lower scored features would have a weight value of 0. This approach was also considered and the results are presented in section 5.3.5. The benefits and shortcomings of the scalar additions and subtractions, as well as possible further improvements to the weight calculations will be discussed in chapter 6.

In order to finish-up the creation of the model and make it ready for retrievals, we need to define local and global similarities. Local similarities define relationships between different values of a certain feature. Using local similarities, how similar are all the possible values for a feature is established. On the other hand, global similarities define relationships between different data points. Here, we establish how to consider two data points as more or less similar. Local and global similarities have been explained in detail in section 2.4.2.

It is important to reiterate that different types of features require different local similarities. After creating a local similarity for all of the features that are being scored, the last step is to create one (or multiple) global similarity function that is used for retrieval from the case base. Since we are using a weighted sum as our global similarity method, and have already calculated the weights for each of the features, simply connecting the weights with the respective feature and its local similarity is done to create a global similarity function. For finding the optimal solution, we create multiple global similarity functions. The results are presented in section 5.3.5.

Having created the local and global similarity functions, the construction of the model is complete and the model is ready for retrieval.

4.2.4 Distance Metric and Similarity Computations

After the model creation, we move on to testing model fairness. As mentioned, verifying model fairness is inspired by the verifier presented in [John, Vijaykeerthy, and Saha, 2020]. Their verifier uses an optimization approach to determine fairness of a given model. This is explained in detail in section 3.2. While this approach would work with our model, we argue that it could be simplified for our use case, without losing its functionalities. There are two key reasons for this. The first one is since we are considering a CBR system, which covers classification problems, there is a defined and countable number of possibilities that a class value can be. This differs to a regression problem, which their verifier tackles, among others. This difference removes the need to perform minimization on the retrieved values. The second reason that we can simplify the verifier is that we have pre-determined values of thresholds ε_t for different feature partitions S_t.

In order to define local similarities, the verifier in [John, Vijaykeerthy, and Saha, 2020] introduces feature partitions S_t and accommodating thresholds ε_t. How different values of a feature may be in order to be considered similar is engraved in the threshold ε. Let's consider two data points DF1 and DF2 and consider they both have a feature A. The values of feature A for DF1 is x_{A_1} and for DF2 it's x_{A_2}. To be able to compare how similar these data points are when considering feature A, we need to define how different they can be in order for them to still be considered similar. The threshold ε presents this value. To state two features' values are similar, they have to satisfy the following condition:

$$|x_{A_1} - x_{A_2}| \leq \varepsilon$$

For example, if x_{A_1} is one, x_{A_2} is zero and the threshold ε is one or more, the two features' values are considered similar. Logically, if ε is less than one, the two feature values are not considered similar. Instead of creating a threshold ε for each feature, [John, Vijaykeerthy, and Saha, 2020] propose grouping features in partitions. This should be done by domain experts and allow similar types of features to have the same threshold values. In the end, every feature partition S_t has its own threshold ε_t.

We argue that for our use case, this is not necessary. Firstly, in order to properly separate features into partitions and set accurate threshold values, domain experts are required. This requirement might not always be available, and in our case, it would require additional resources which wouldn't been easy to acquire. Secondly, our goal is to prove that CBR systems can be used to ensure fairness. As a proof of concept, we argue that separating our features into only two partitions; protected and non-protected features is sufficient. The threshold for protected features is set to infinity, which implies that every value for a protected feature is similar to another one. This is done since to ensure individual fairness, protected features shouldn't influence similarity of two data points. On the other hand, the threshold for the non-protected features is set to zero. This is done to remove the need for domain experts since we consider only features that have identical values to be similar. We argue that, as a proof of concept, these thresholds are sufficient. In any future research and improvements, more partitions and thresholds could be seamlessly added.

Having defined the similarity of two data points, the next step is to collect all the similar pairs of data points.

4.2.5 Collecting Similar Pairs of Data Points and Evaluating Fairness

After defining the thresholds, we proceed to recognition of similar pairs. To perform this step, we compare every data point to every other data point in the test dataset. More precisely, their features values are compared. Since we set the threshold for protected features to infinity, every value of a protected feature is similar, thus they are not considered when comparisons are done. Knowing this, we examine every non-protected feature of two data points. Since the threshold for non-protected features is set to zero, the two data points must have the same values for a non-protected feature so it can be said that they are similar according to that feature. If this is true for all the non-protected features of two data points, we state that these two data points are similar. A pair of these two data points is created and added to a set S. After all the combinations of different data points are compared, set S contains all pairs of data points that are considered similar in the test dataset.

Once set S is created, we move on to the final step in order to test model fairness. This requires performing retrievals from the model and comparing their results.

In order to proclaim a model is individually fair, we must show that it treats similar data points in a same manner, regardless of protected features' values. In the final step, we iterate through pairs collected in set S. For each of the two data points constructing a pair, a retrieval is made to the CBR system. As we are using a *weighted kNN* approach, the CBR system will return k most similar cases to the queried data point. To determine the predicted class label, a majority voting[2] is performed. After the predicted class values are obtained for both data points, they are compared. If the predicted class values are the same, the similar data points are handled correctly and we move to testing other pairs. However, if the predicted class values are different, it should be checked if it is due to the difference in protected features' values. If any of the protected features' values are different in the two data points, we have found a biased pair of data point i.e. data points that are similar and should be treated equally, but are not. This indicates that the model is individually unfair and should not be used for future retrievals before modifications are made.

[2]Majority voting is a method used to determine the predicted class of a kNN retrieval. It counts up the different class values in all the retrieved cases and uses the class value which occurs the most as its prediction of the class value.

At this point, further comparison of similar pairs is done in order to provide more data to the engineer(s) who is(are) developing the CBR system to, more easily, identify and resolve the underlying issue. Each biased pair is added to a set called *bias*.

4.2.6 Pseudocode for Investigation of RQ2

The pseudocode used to answer RQ2 is provided in the following algorithms.

Algorithm 5 displays how the CBR system is built. In order to calculate the weights required for the global similarity function, we compute the sum of scores in the same manner as in algorithm 4. After we have obtained the sum of scores for each feature, we use equation 4.6 to calculate the weights for each of the features. Having created the global similarity, the system is finished and ready for retrievals.

Algorithm 5: Modified relevance-based feature weighting algorithm

```
Input: $dataset \leftarrow chosen\ dataset$
Input: $methods \leftarrow feature\ relevance\ scoring\ methods$
Output: $Weights$
$weights \leftarrow computeFeatureWeights(dataset,\ methods)$
Function computeFeatureWeights($dataset, methods, percent$):
    $rank_{sum} \leftarrow getSumOfScores(getScores(dataset, methods))$    // From algorithms 2 and 4
    $N \leftarrow size(rank_{sum})$
    $min \leftarrow min(rank_{sum})$    // minimum score value
    $max \leftarrow max(rank_{sum})$    // maximum score value
    $weights \leftarrow empty\ list$
    for (label, score) in $rank_{sum}$ do
        $weight \leftarrow [N-1] \cdot [\frac{score-min}{max-min}] + 1$
        $weights.insert(label,\ weight)$
    end
return $weights$
```

The process of testing model fairness is presented with algorithm 6.

Algorithm 6: Testing fairness of CBR system

```
Input: $dataset \leftarrow chosen\ test\ dataset$
Input: $f \leftarrow provide\ the\ model\ for\ retrieval$
Input: $protected \leftarrow labels\ of\ features\ considered\ protected$
Output: $Set\ of\ B\ pairs\ or\ empty\ set$
$B \leftarrow verifyModel(dataset,\ f,\ protected)$
Function verifyModel($dataset, f, protected$):
    $S \leftarrow createS(dataset, protected)$
    $B \leftarrow empty\ list$
    for (X, Y) in S do
        $predicted_X = f(X)$
        $predicted_Y = f(Y)$
        if $predicted_X\ != \ predicted_Y$ and
         $differentProtectedFeatures(X, Y, protected)$ then
            $B.insert(\{\{X,\ Y\}, \{predicted_X,\ predicted_Y\}\})$
        end
    end
return $B$
```

As input the test dataset, the model that is being tested, and the protected features' labels are required. Firstly, a set S of similar pairs is created, as shown in algorithm 7.

Algorithm 7: Collecting similar pairs of data points

```
Function createS($dataset, protected$):
    $S \leftarrow empty\ list$
    $n \leftarrow total\ number\ of\ data\ points\ in\ dataset$
    for i from 0 to n do
        for j from i+1 to n do
            if $similarInstances(dataset[i],\ dataset[j],\ protected)$ then
                $S.insert(dataset[i],\ dataset[j])$
            end
        end
    end
return $S$
```

Similar pairs are considered by comparing all non-protected feature values of pairs. This is presented in algorithm 8.

Algorithm 8: Comparing similarity of two data points

```
Function similarInstances(X, Y, protected):
    for feature in X do
        if feature not in protected and $X[feature]$ != $Y[feature]$ then
            return False
        end
    end
return True
```

After the creation of set S, we turn to model retrievals. Iterating through set S, we perform a retrieval on each of the similar data points in a pair. If the retrievals' predicted results differ for the two data points the protected features are compared. This is shown in algorithm 9.

Algorithm 9: Comparing protected features of two data points

```
Function differentProtectedFeatures(X, Y, protected):
    for feature in protected do
        if $X[feature]$ != $Y[feature]$ then
            return True
        end
    end
return False
```

Finally, if the protected features are different, we have found a biased pair of data points and add it to set B.

At the end, set B is presented to the engineer(s). If set B is empty, no individual unfairness has been detected. If not, the model outputs individual unfairness and should not be used without modifications.

It is important to recognize that if set B is empty, the model is not necessarily fair. This is due to the fact that we queried only a finite number of data point from the test dataset which might not cover all possible combination of feature values. This is discussed in more detail in section 6.1.

Chapter 5

Experiments and Results

In chapter 4, the methods and the architecture underlying our investigation of RQ1 and RQ2 from section 1.1, is described. This chapter presents the related experiments and the obtained results. Firstly, the used datasets are described along with their respective protected features. Secondly, a detailed explanation of the experiments' plan, including the setup and parameters, is provided. Finally, the results of the experiments are presented and analysed.

As mentioned in chapter 4, in order to answer RQ1, we use an approach relying on computing feature scores in such a way that captures the relationships of features with the different labels of classes. This approach provides us with protected features' influence values. By examining these, we show how one can use them in order to detect dataset bias. The setup for experiment 1 which relates to RQ1, is based on this approach. Firstly, the datasets, and their protected features, used in experiment 1 are described and explained. After this, the pre-processing method, used for calculating protected features' influentiality, is applied to five different, publicly available, datasets. Finally, the results of the pre-processing method are presented and analysed. In section 5.3 the second experiment setup is explained. Firstly, we explain dataset choices and splitting the data into a train and test dataset. Next, local and global similarity function choices that are used in each experiment, are explained. Afterwards, the CBR models are tested in regards to individual fairness. Finally, the results of the fairness testing are presented and analysed.

5.1 Description of Datasets

Since our experiments rely heavily on being able to determine if a dataset contains bias or not, we used datasets that are commonly used in fairness research and have been explained extensively. One additional dataset not previously used in this context was added; Credit Card Approval. This dataset is included since it is used in similar types of decision making systems as the rest of the mentioned datasets. In total, five datasets are used throughout the experiments. They are, as follows:

- ***Adult Census Income Dataset (ACID)*** [Kohavi and Becker, 1996] - The Adult Census Income Dataset relates to income of adults in the United

States of America. More precisely, the ACID is created to predict if the income of adults is above or equal/below 50.000 $ annually. The original dataset provide fourteen different features such as education, occupation, capital gain/loss among others. In order to remove unnecessary features, we use a reduced version of the dataset containing eight features. This is provided by [Mothilal, Sharma, and Tan, 2020]. ACID has two possible class labels: annual income equal/higher to/than 50.000 $ or annual income lower than 50.000 $.

- ***COMPAS Dataset*** [Bellamy et al., 2018] - COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is a popular commercial algorithm used by judges and parole officers for scoring criminal defendant's likelihood of reoffending (recidivism). This dataset was made famous after an article was published describing how a US predictive recidivism system (called COMPAS) was biased against black people Angwin et al., 2016. More in depth uses and issues were presented in section 2.1. This dataset provides features describing criminal defendants that were scored by the COMPAS system. The class labels present the decision if the system predicts that the criminal defendant will/won't recidivate[1].

- ***Fraud Detection Dataset (FDD)*** [Dua and Graff, 2017] - It is important that car insurance companies are able to recognize fraudulent insurance request so that customers are not compensated for fraudulent claims. This dataset provides data instances classified into fraudulent and non-fraudulent car insurance claims. The features contain categorical values for police reports, policy claims and also numerical values as days until the incident was reported. The two possible class labels are fraudulent and non-fraudulent. It is important to note that the distribution of fraudulent to non-fraudulent claims does not represent the actual real world distribution. Usually, fraudulent claims are very rare, but being able to detect them properly is significant.

- ***Credit Card Approval Dataset (CCAD)*** [Tse, 2020] - Automated credit card approval ML systems have been growing in popularity. The CCAD contains data points representing people from the real world which are classified into risk or no risk people when considering approving their new credit card request. In these types of classification problems, some of the more important features are duration of employment, credit history, annual income and type of job, to name some. When evaluating if a person should be granted a credit card, the class label can be either high or low risk.

- ***German Credit Dataset (GCD)*** [Hofmann, 1994] - The German Credit Dataset contains data points describing people that have applied for a credit loan. The class label concerns with classifying a persons credit rating as one of two possibilities; good or bad. Similarly to CCAD, it

[1]To relapse into a previous condition or mode of behavior and especially delinquency or criminal activity : to exhibit recidivism

contains a variety of features such as credit history, amount, duration of credit in order to make an informed decision in regards to a persons' credit rating.

All the dataset are designed for multivariate classification tasks and their features have categorical and numerical values. The instances which have missing values in the datasets have been removed in order to prevent issues when comparing data points. A summary of the five mentioned datasets is presented in table 5.1.

Dataset	Instances	Features	Class labels
Adult Census Income Dataset (ACID)	32,560	8	2
COMPAS Dataset	6,172	12	2
Fraud Detection Dataset (FDD)	1,100	9	2
Credit Card Approval Dataset (CCAD)	537,667	17	2
German Credit Dataset (GCD)	1,000	20	2

TABLE 5.1: Description of the datasets used in the experiments

5.1.1 Selection of Protected Features

As mentioned in section 2.2, protected features contain sensitive information about the individuals represented as data points in a dataset. For this reason, protected features must not influence the result of classifications. Since these features must be handled properly, they need to be defined before using the data points from the dataset. In order to recognize which features should be protected, one must understand the full structure and context of the dataset. This is the reason why protected features should be decided by domain experts, if possible. When this choice is done by domain experts, the ML engineer is relieved of deciding the protected features for individual datasets and can focus on designing automated systems that handle any type of protected features properly. Of course, if this separation is not possible, the ML engineer should decide the protected features by acquiring as much context about the dataset as possible. Since our resources are limited, we used the latter approach.

For each dataset, we researched and contextualised the information that is available. In the Adult Census Income Dataset, projected annual income is predicted. From all the features, we decided that gender and race are protected features since their difference should not affect the predicted income of a person. When researching the COMPAS dataset, a similar conclusion was made. Since the goal of COMPAS is to decide if the person will recidivate or not, race or gender are not relevant in calculations. The final dataset that has the same protected attributes is Fraud Detection Dataset. Similarly as before, we argue that validness of a claim should not be evaluated using race or gender of the person that reported the claim.

The fourth dataset that was considered was the Credit Card Approval Dataset. Even though it contains seventeen features, the vast majority are non-protected features that should be used for evaluation. The only protected feature we identified is gender, as this should have no part in evaluating credit card requests. The final dataset is the German Credit Dataset. It is different from the others as some of the features have twofold information values. One instance of this, as well as the first protected feature, is sex_marital_status. This feature retains information about the gender of the person applying for a credit as well as their marital status. We argue that this feature should be protected as its value could discriminate a gender type if it was used in evaluation of a credit request. The other feature we identified as protected is is_foreign_worker. We advocate that this feature could be used to ethnically profile people and therefore shouldn't be used in evaluation.

Intuitively, one might think that masking protected features would be enough since if they are all the same, there shouldn't be a possibility for biased. While this approach might work sometimes, we argue it does not ensure an unbiased dataset. The reason for this is that even without the protected features, bias could be hidden in proxy features. Proxy features are non-protected features being used to create biases against protected classes of features by storing bias in their values.

For example, regulators do not deem postal codes as a protected attribute and allow them to be used in lending decisions. However, a postal code could be a proxy for race or religion and unintentionally discriminate against some individuals. However, the topic of proxy attributes is outside the scope of this thesis.

5.2 Setup and Plan for Experiment 1

In order to answer RQ1, we construct an experiment as described in detail in section 4.1.2. As the datasets and their chosen protected features have been discussed, following the steps shown in figure 4.1 we move on to feature relevance scoring methods (FRSMs) selection.

To perform our experiment, we combine scores from different FRSMs. The easiest implementation of FRSMs, for our use case, was using the open source data mining tool Orange. This decision was justified in section 4.1.3. As Orange provides a variety of FRSMs, we use six of the default ones. They are, as follows:

- ***Information Gain*** - measures the gain in information entropy by using a feature with respect to the class [Heilprin, 1960].
- ***Information Gain Ratio*** - is a ratio of the information gain and the attribute's intrinsic information. This reduces the bias towards multi-valued features that occurs in the information gain [Quinlan, 1986].
- ***Gini decrease*** - is a measure commonly used in decision trees to decide what is the best attribute to split the current node for an efficient decision tree construction. It is a measure of statistical dispersion and can

be interpreted as a measure of impurity for a feature or the inequality among values of a frequency distribution [Ceriani and Verme, 2012].

- ***Chi*^{2} *(X*^{2}*)*** - evaluates each feature individually by measuring the chi-squared statistic with respect to the class [Pearson, 1900].
- ***ReliefF*** - uses the ability of an attribute to distinguish between classes on similar data instances [Robnik-Sikonja and Kononenko, 2000].
- ***Fast Correlation Based Filter (FCBF)*** - is an entropy-based measure, which also identifies redundancy due to pairwise correlations between features [Yu and Liu, 2003].

The next step is to score the influence of features in the dataset. After scoring all the features in each FRSM, we proceed to ranking the features. In this step, we introduced a modification of the approach shown in [Jaiswal and Bach, 2019] to ranking features relevance. The original approach uses linear rankings to rank features inside each FRSM. We argue that this way of ranking the features does not portray the actual influence of features presented in scores of a FRSM. Since our architecture relies heavily on being able to calculate influence, this is something that needed to be handled. Instead of the linear ranking, we scale the scores of features inside a FRSM to a fixed range. Using scaling, the scores are preserved in the ranking of features. This is explained in detail in section 4.1.4. In experiment 1, we decided to present both approaches to ranking the features, in order to compare the results and show if there are improvements to using our approach.

After scoring the features for each of the methods, and ranking the features (or scaling their scores) in each of the FRSM, we calculate the sum of scores for each individual feature. The next step is to calculate the influence of the protected features compared to the top scored feature using their respective sum of scores. Finally, as an optional step, we choose the value of parameter p which decides the value for the threshold λ. As explained in section 4.1.7, parameter p represents the maximum amount of influence a protected feature can have if the dataset is to be considered unbiased. The threshold λ is calculated by multiplying the parameter p and the $sumofscores$ of the highest scored (most influential) feature. Even though the threshold does not affect the results, we argue it's is useful so there is a clear percentile value p above which protected features' influence must not be in order to label the dataset unbiased.

For the value of p we choose 10% as we feel it represents a large enough gap to allow for errors in calculations while also small enough where protected features that have a larger influence are genuinely handled incorrectly.

In order to evaluate the proposed method, we use a bias mitigation technique presented in section 4.1.1 to ensure that we have one dataset that is entirely individually fair. After this, we use this dataset as input to the pre-processing method and compare the results to the original.

5.2.1 Resources and Tools

The datasets are publicly available and were retrieved from the links provided as references in section 5.1. Most of the calculations were done using the GUI elements of Orange version 3.28.0. The calculations of sum of scores and influence were done using a Python script in the Orange GUI. In order to remove biased instances from the dataset, a separate Python script was created. Both scripts were executed using Python 3.6.

5.2.2 Results of experiment 1

Firstly, let us look at the results of the pre-processing method using our scaling of scores and a percentage p of 10% meaning that 10% is the highest influence a protected feature can have in order to label the dataset unbiased. The results are presented in table 5.2. For each of the datasets, we show the highest scoring features' sum of scores, the lambda of that dataset when p is 10%, sum of scores for each protected feature and, a decision whether our method detects bias in the dataset, using these parameters.

Dataset	Highest scoring feature sum_of_scores	Lambda	Protected feature sum_of_scores		Bias detected
			Gender	Race	
ACID	48.0	4.8	8.89	2.88	Yes
COMPAS	1940.79	194.07	343.98	506.88	Yes
FDD	52.87	5.28	7.14	5.57	Yes
CCAD	65	6.5	4.87	N/A	No
			sex_marital_status	is_foreign_worker	
GCD	120	12	13.96	17.48	Yes

TABLE 5.2: Results from experiment 1 using p = 10% showing sum_of_scores for the highest scoring and protected features. The final columns shows the decision if the dataset shows bias i.e. if one of the protected features' sum_of_scores exceeds value of λ

Next, let's present the actual percentage values of each protected feature, since p choosing is optional. This can be seen in table 5.3.

Dataset	Protected features influence	
	Gender	Race
ACID	18.5%	6%
COMPAS	17.72%	26.12%
FDD	13.5%	10.53%
CCAD	7.49%	N/A
	sex_marital_status	is_foreign_worker
GCD	11.63%	14.56%

TABLE 5.3: Results from experiment 1 showing the influence of each feature on the classification result, in terms of a percentage.

In order to compare our scaling method for computing feature scores in each FRSM with the ranking method by [Jaiswal and Bach, 2019], we perform the same experiment using the linear rankings. These influence percentages can be seen in table 5.4. As mentioned in 3.3, their approach required setting a percent of features that are scored in each FRSM. For the percent value, we decided for two possibilities: 75 and 100 percent. This decision is driven by the fact that these percentages have shown the best model results with various datasets used in [Jaiswal and Bach, 2019].

percentage of features ranked	75%	100%	75%	100%
Dataset	Protected features influence			
	Gender		Race	
ACID	21.2%	40.9%	9.09%	27.27%
COMPAS	79.04%	84.54%	79.04%	84.54%
FDD	24.4%	41.5%	0%	15.09%
CCAD	76%	80.59%	N/A	
	sex_marital_status		is_foreign_worker	
GCD	26.97%	45.37%	43.82%	57.98%

TABLE 5.4: Results from experiment 1 showing influence percentages per feature using linear rankings

Finally, in order to discover if our method actually detects bias in a dataset, we removed all the bias instances from the Fraud Detection Dataset, and ran the experiment again. This time, we used the cleaned Fraud Detection Dataset. The results of this run are presented in table 5.5. For an easier comparison, the results of the original run are also shown.

Dataset	Clean	Highest scoring feature	Lambda	Protected features				Bias detected
				Gender		Race		
FDD	No	52.87	5.28	7.14	13.5%	5.57	10.53%	Yes
	Yes	54	5.4	0.38	0.7%	1.65	3.07%	No

TABLE 5.5: Results of experiment 1, with the original and cleaned versions of the Fraud Detection dataset.

5.2.3 Analysis of Results

As presented in table 5.2, most of the original datasets are flagged as biased datasets. Only in the Credit Card Approval Dataset, no bias was detected. If we consider the percentage values shown in table 5.3, we argue that the flagged datasets should be considered biased. While some feature percentages barely go over the threshold (e.g. race in FDD), some show undoubted biased as is the case with race in the COMPAS dataset. These results fit with what previous researchers reported when working with these datasets (for example, see [John, Vijaykeerthy, and Saha, 2020], [Aggarwal et al., 2019]). While one could argue that with a different value of p, the dataset would render unbiased, we reiterate that, as explained in section 4.1.7, percentage p is used purely for easier visualization, while the actual influence is the same regardless of its value.

When considering whether scaling the scores was the correct choice, we examine table 5.4. Even though we can not say for certain which approach is better, intuitively we notice that the percentages are much higher for each of the protected features. While this could be the case, it is highly unlikely that all of the datasets we choose contain bias to this magnitude without being noticed and discussed earlier.

In order to evaluate our method for investigating RQ1, we decided to clean one of the datasets that was labeled as biased in table 5.2, and compare the cleaned results with the original results. The cleaning was performed by identifying and removing all biased instances from the original dataset. A total of 42 biased data points were found and removed, leaving the dataset with a total of 1058 data points. As has been shown in table 5.5, after removing these biased data points, the dataset is no longer considered bias. Not only this, but the influence percentage values of protected features have decreased drastically. We argue that this is a strong indication of the presented pre-processing method successfully detecting bias in datasets, when it actually exists, and in turn, answering RQ1. One issue with definitively stating that the pre-processing method works, is that it needs to be tested on multiple datasets where the before and after cleaning values are compared. Due to time and resource constraints, these experiments are outside of the scope of this thesis, but are considered required future work to certify the methods' usefulness. This will be discussed in chapter 6.

5.3 Setup and Plan for Experiment 1

In order to answer RQ2, we constructed an experiment as described in detail in section 4.2. As mentioned in section 4.2.1, in order to test fairness, a CBR system needs to be constructed. To complete the construction of a CBR system we need to create an unbiased case base and local and global similarity functions. The first part of model construction is identical to the one used for experiment 1 so it will not be explained in detail. Firstly, we needed to choose which datasets are going to be used to construct the CBR model. We chose to use two of the datasets already covered and used in experiment 1. These are the Fraud Detection Dataset and the Adult Census Income Dataset.

5.3.1 Chosen Datasets and Label Distributions

As we mention throughout this thesis, we argue that it's highly important to ensure we are using unbiased datasets in order to be able to study the fairness of the classification algorithm itself. This is why we decided to remove all the biased instances from both the datasets before using the data to construct the models. The mitigation was done using the approach explained in section 4.1.1. We collected all the biased pairs of instances in both datasets and removed them entirely. One issue that occurred, which was expected and explained in section 4.1.1 relates to the number of comparisons necessary to perform the detection of biased pairs of data points in the Adult Census Income Dataset. Since this dataset has more than 30,000 data points, it would require 900,000,000 comparisons of data points to be performed. Using the computing resources we have, the time to finish this step would take around 6 days. Since this wasn't feasible, we decided to reduce the number of data points that are used to construct the CBR model.

Instead of the original 30,000, we used the first 4,000 data points in the dataset. This reduced the execution time to about half an hour, which we deemed as acceptable. There are a few drawbacks that one might notice with this reduction. One is regarding using only the first 4,000 data points instead of randomly sampling the data points. In an attempt to present the original dataset as best as possible, we explored the distributions of class and protected features' labels in both the original and the reduced version of the dataset. The results, presented in figures 5.1, 5.2, show that the distribution of the labels was preserved in the reduced variant of the dataset, hence there was no need for random sampling.

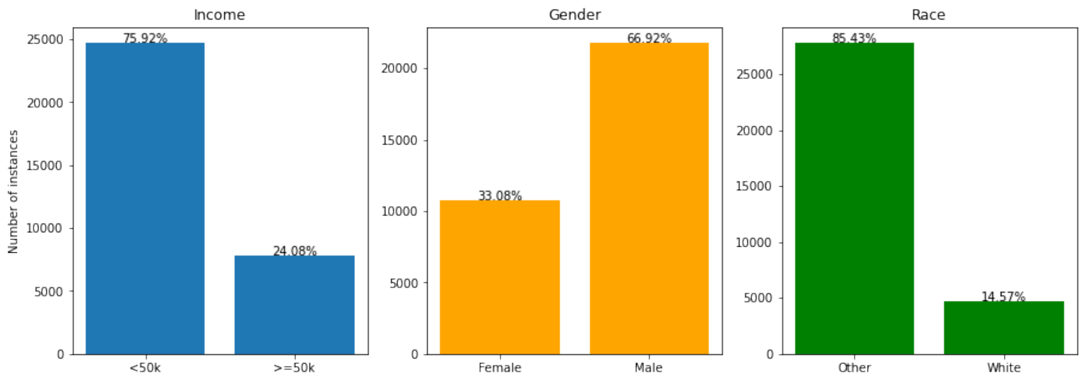

FIGURE 5.1: Distribution of class labels (blue) and values of protected features in the full version of the ACI dataset

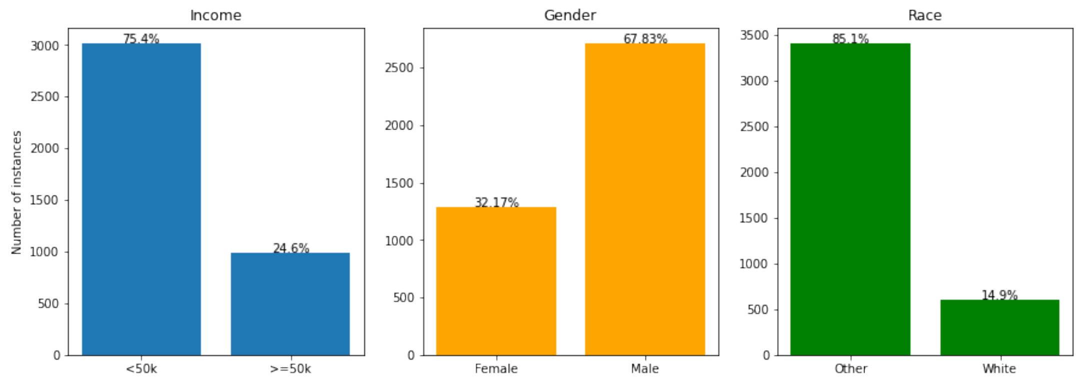

FIGURE 5.2: Distribution of class labels (blue) and values of protected features in the reduced version of the ACI dataset

The second issue is in regards to individual unfairness itself. Since its approach compares data points directly, even removing one unfair data point can change the results altogether. Knowing this, using the full dataset and removing its biased instances is the preferred approach. However, due to the limitations we mentioned earlier, we argue that reducing the dataset is a valid approach and will serve as a proof of concept. We encourage future work to be done using the full dataset as it could provide for useful comparisons to our model. Having removed all the biased data points from the two datasets, the next step is to split the data. In a CBR system, there is not a training step per-say, but the training dataset is used as a case base for retrieval purposes. We opted for a 75%-25% split. The reason for choosing this split is to fall in line with the split used in [John, Vijaykeerthy, and Saha, 2020] since we compare results to those presented in their work. As with the reduced dataset, we strived to ensure both the train and test dataset have equal distributions of class and protected values labels. As is shown in figures 5.3, 5.4 ,5.5 for ACID and in figures 5.6, 5.7, 5.8, 5.9 for FDD the distributions are similar enough in both the train and test datasets.

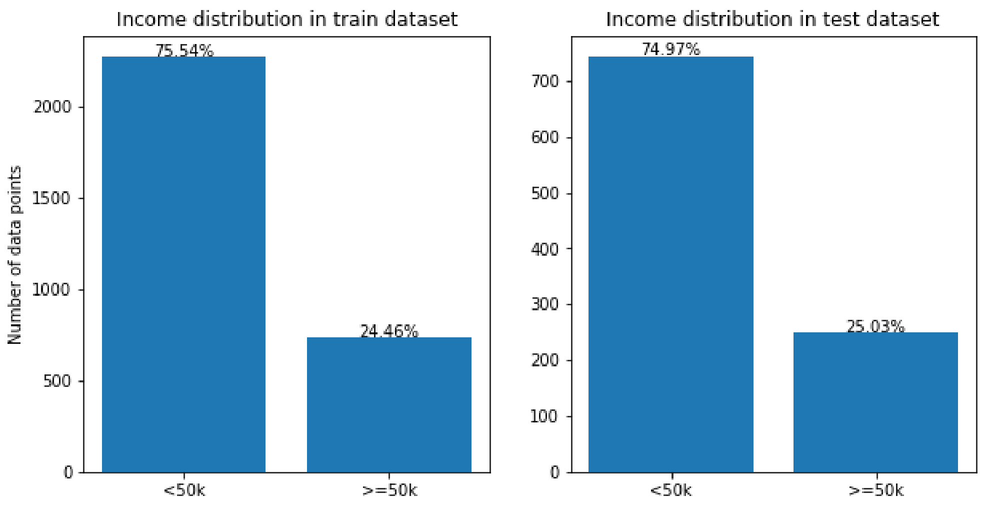

FIGURE 5.3: Distribution of class label *income* in the reduced version of the ACI dataset split into train and test datasets

Having ensured we use unbiased datasets for model construction, we explain how we calculated the weights and created local and global similarity functions for both models.

5.3.2 Calculating Weights and Creation of Local and Global Similarity Functions

As discussed in section 4.2.3, calculating the weights and creating local and global similarity functions are the final steps of constructing a CBR model. To reiterate, using local similarities relationships between different values of a certain feature are established. How similar are all the possible values for a feature is decided. On the other hand, global similarity functions define relationships between different data points. Since the two models have the same

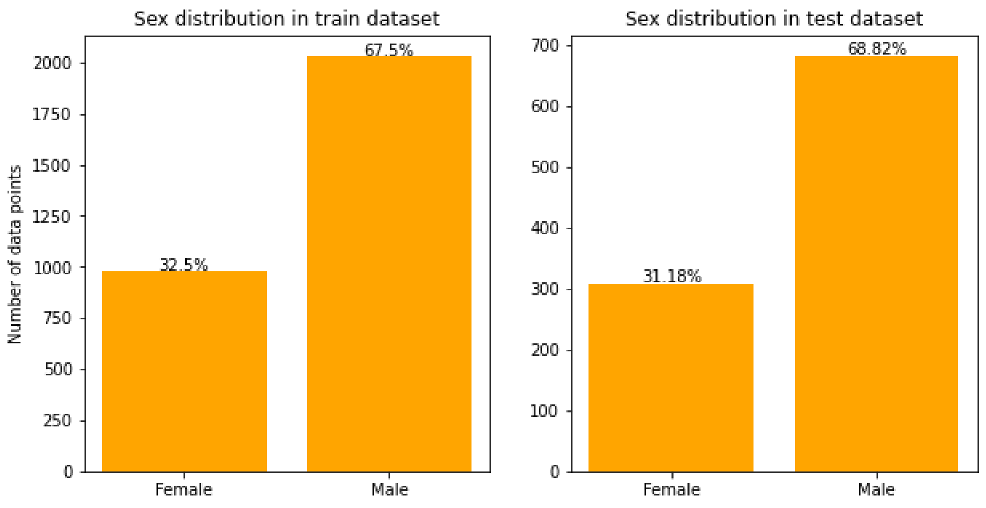

FIGURE 5.4: Distribution of protected feature *gender* in the reduced version of the ACI dataset split into train and test datasets

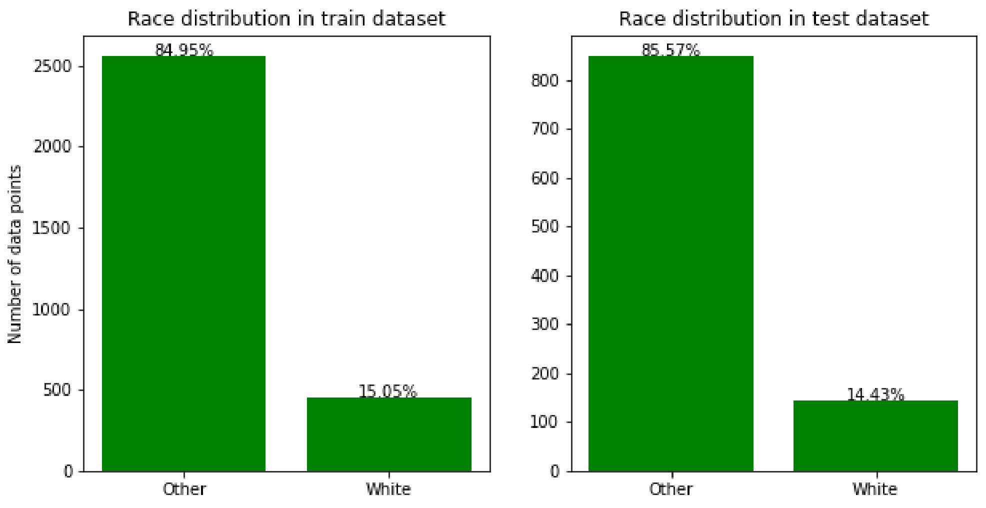

FIGURE 5.5: Distribution of protected feature *race* in the reduced version of the ACI dataset split into train and test datasets

types of global similarity functions with different features and weights, we list all the different weight calculations we used in both models and explain the rationale of using them for retrieval purposes.

In order to investigate how our proposed weight calculation method compares to the methods in [Jaiswal and Bach, 2019], we implemented the best performing methods from the paper. This framework is explained in section 3.3. The methods that showed the best results in [Jaiswal and Bach, 2019]'s paper were scoring 75 and 100 percent of features, so these are implemented. The third weight calculation we implement is our proposed method. It uses scaling of scores in FRSMs and is explained in detail in section 4.1.4. The final weight calculation method we implement is slightly modified to the previous

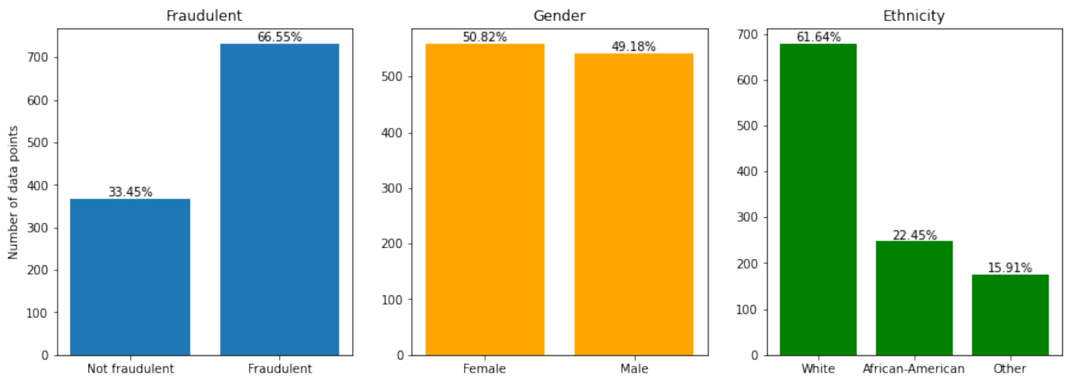

FIGURE 5.6: Distribution of class labels (blue) and values of protected features in the FD dataset

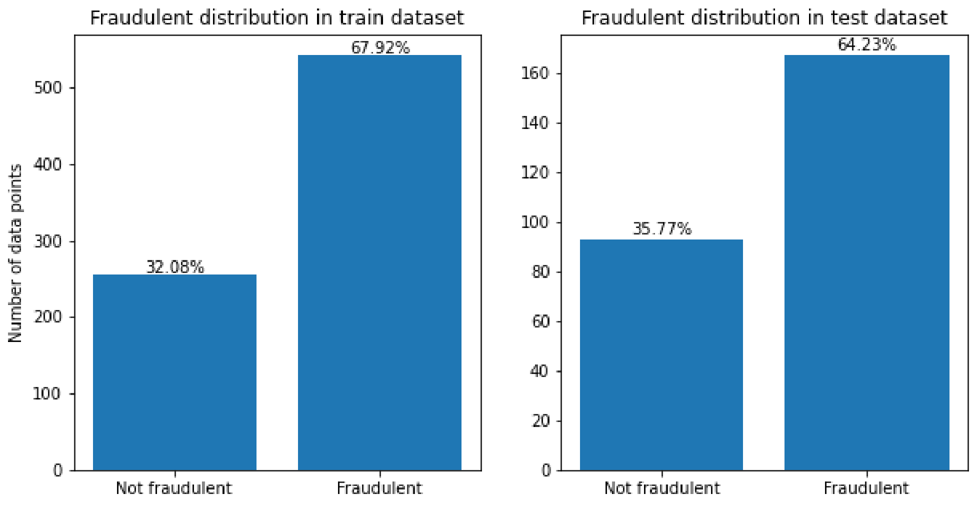

FIGURE 5.7: Distribution of class label *fraudulent* in the FD dataset split into train and test datasets

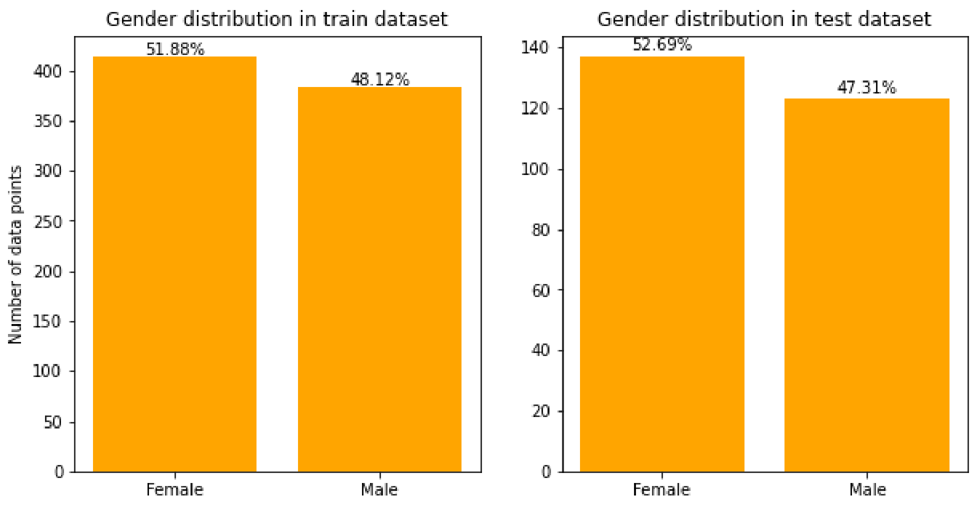

FIGURE 5.8: Distribution of protected feature *gender* in the FD dataset split into train and test datasets

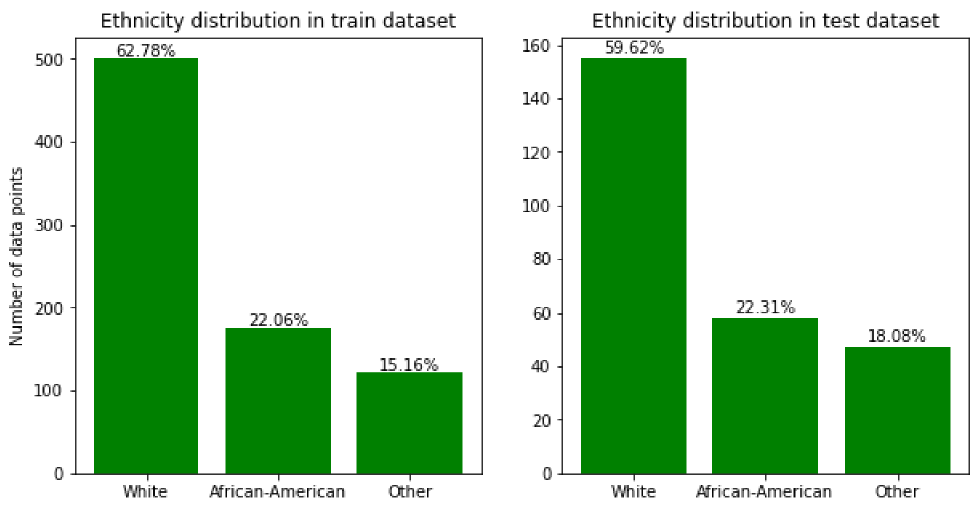

FIGURE 5.9: Distribution of protected feature *ethnicity* in the FD dataset split into train and test datasets

one. Once again, we scale the scores in the FRSMs but the formula for calculating the actual weights, presented in equation 4.6, is modified. In the original equation, n is subtracted by 1 while 1 is also added to the final result. This was done to ensure all the features are weighted. We attempted removing this subtraction and addition. We argue that this approach is worth exploring since it could be possible that some features with a low score should not have a weight value at all i.e. they shouldn't be used to help with retrievals.

On top of the mentioned calculation types, we introduce one difference that doubles every type use. For each of the mentioned calculation types we add a variant in which the protected features are masked. This is achieved by removing the protected features from the FRSMs. As a result, in these weight calculations, all the protected attributes have zero as their weight value and are not used for the retrieval process. This is introduced in order to both test accuracy of the models compared to non-masked approaches as well as to see if there is any unfairness removed when switching to masked approaches.

As our global similarity functions are based on the weighted sum method, the creation of global similarity functions requires weight calculation and local similarity function definitions. Since we mentioned different weight calculation approaches, to finish-up creation of the global similarity functions, we explain the rationale of creating local similarity functions.

In order to properly compare different values of a feature, one must define a local similarity function. Local similarity functions vary in type based on the type of the features' values it is used for. In general, domain experts are tasked with deciding how to create local similarity functions. Lacking domain experts' knowledge and in an attempt to simplify the system, we used the default local similarity functions for each of the different types of features. This implies that for categorical features, two values are considered similar only when they are identical (i.e. have the same value), and their similarity score is 1. In all other cases, the similarity between the different values is 0.

In the case of numerical values, we used a simple symmetric polynomial function. This implies that the closer two numbers are, the higher the local similarity score is. For example, if we consider the age feature. If we have a age value of 30 and 35, the difference between the age values is only 5 years and the similarity between the values is very high (around 0.95 in this case). Contrasting this, if we compared ages 20 and 80 the difference is much larger, producing 50 years of difference. This induces a low local similarity score (around 0.2 in our use case).

Creating and testing different local similarity functions would be a highly valuable venture for future work, but due to the limited time, resources and scope of this thesis, we do not explore this venture.

Having calculated the weights, and created both global and local similarity functions, the CBR model construction is completed.

5.3.3 Verifying Individual Fairness of Models

The first step in evaluating the individual fairness of a model is defining the similarity of two data points. This is explained in detail in section 4.2.4. As explained, in order to define similarity, the approach in [John, Vijaykeerthy, and Saha, 2020] uses feature partitions and thresholds for each of these. Following rationale from section 4.2.4, we use only two partitions and thresholds. One partition is created using all the protected features, while the other contains all the non-protected features. Protected features partition has a symbolic threshold value of infinity, indicating that data points shouldn't be distinguished at all using protected features. The non-protected partition has a threshold value of zero. This is done to ensure that a similar pair of data points is one where all the non-protected features are the same value. Using equality of values, similarly as with local similarity functions during the construction of the models, we remove the need for domain experts in deciding the thresholds. There is one exception to the non-protected threshold that we made. This is in regards to the Fraud Detection Dataset.

FDD has a feature called *days_to_incident*. This features contains numerical values vary from 2 to 15,000. If we were to keep the zero threshold i.e. expecting to find two data points with the same amount of days passed to incident, we find almost no similar data points. It is for this reason, we decided not to consider similarity of this feature when looking at similarity of two data points. If more time was present, we would create an acceptable threshold by examining the values of data points, and figuring out how big of a gap in values can be allowed for the data points to still be considered similar.

After defining similarity of two data points, the next step is collecting similar pairs from the dataset. Collecting the pairs is achieved by comparing all combinations of two data points. In every comparison, we examine the non-protected features and their values. If the two data points have the same values for all non-protected features, they are considered similar and are added to a set of pairs of similar data points S.

The final step of testing fairness of a model is by performing retrievals on the pairs of similar data points collected in set S. For one pair of similar data points, both data points are queried to the CBR model. After the results for both are retrieved, the classification predictions are compared. If they are the same, this is not a biased pair and the system can continue looking at other pairs. However, if the results are not the same, we observe the protected features. If the protected features' values are different (for at least one protected feature), we have found a situation where the system handles similar people differently (based on protected features) and label it as a system that outputs individual unfairness. As an optional step, we opted to continue the retrievals, even though the system has already showed unfairness. We argue that additional data can be used by the ML engineer to more easily find and mitigate the underlying issue.

One last parameter that needs rationale is the value k in the retrieval from CBR models. The value k decides how many similar cases, to the case that is queried, are retrieved. Once more, in order to simplify results, we use the convention of choosing k as the closest odd number to the square root of the number of data instances in the test dataset. Testing with different k values could provide an interesting venture for future work.

5.3.4 Resources and Tools

The steps of experiment 2 that are shared with experiment 1, as well as the weight calculations are implemented using Oranges GUI. The actual CBR models, including the global and local similarity functions, are constructed using the myCBR tool, more precisely, its workbench. MyCBRs REST API module is used when performing retrievals to the models. MyCBR provides an easy to use GUI allowing quick construction of models, its case base, and similarity functions. For more information on myCBR, refer to [Stahl and Roth-Berghofer, 2008].

Algorithms used for testing models fairness were coded using Jupyter Notebooks. The code was executed using Python version 3.6. Plotting of graphs and charts was done using the matplotlib.pyplot package.

5.3.5 Results of Experiment 2

In order to ensure unbiased datasets, we detected all the biased pairs of data points and removed them from both training datasets. The datasets characteristics before and after the removal are shown in table 5.6.

After the datasets were cleaned of all biased data points, we proceeded to perform retrieval to all similar pairs of data points in the test dataset. If the

Dataset	Total data points	Biased data points in train dataset	Train dataset (after removal of biased data points)	Test dataset	Similar pairs in the test dataset
ACID	4000	86	2923	991	53
FDD	1100	42	798	260	11

TABLE 5.6: ACID and FDD numbers for train and test datasets

predicted classification differs for two data points found in the same pair, we detected a pair handled unfairly. For each type of weight calculation we tested a version with and without masking the protected features. The result of the retrieval using calculation of weights by scaling the sum of scores is presented in table 5.7.

Dataset	Number of pairs found containing biased data points			
	Weights calculated using scaling each features' sum of scores		Weights calculated using scaling each features' sum of scores without addition and subtraction	
Protected features masked?	No	Yes	No	Yes
ACID	0	0	2	0
FDD	0	0	0	0

TABLE 5.7: Number of biased data points found using the method of scaling sum of scores of all features to define the global similarity function

To test the fairness results with the weight calculation mentioned in Jaiswal and Bach, 2019 to the weight calculation we propose, we implemented their weight calculation using 75 and 100 percent of features being scored. As earlier, for each type of weight calculation we tested a version with and without masking the protected features. The results can be seen in table 5.8.

Dataset	Number of pairs found containing biased data points			
	Weights calculated using ranking 75% of features (old weight calculation)		Weights calculated using ranking 100% of features (old weight calculation)	
Protected features masked?	No	Yes	No	Yes
ACID	2	0	0	0
FDD	4	0	1	0

TABLE 5.8: Number of biased data points found using the method of scoring a selected percentage of features' in each FRSM to calculate sum of scores and define the global similarity function

After performing the fairness test, we found unfairness in classification using some global similarity functions. In figure 5.10, we can observe unfairly classified data points by two global similarity functions when applied to the Adult Census Income Dataset. Firstly, the one using scaling scores in each FRSM without addition and subtraction of one in the weight calculation. The same results are observed when using the global similarity function scoring 75% of top features in each FRSM and assigning linear ranks to those features.

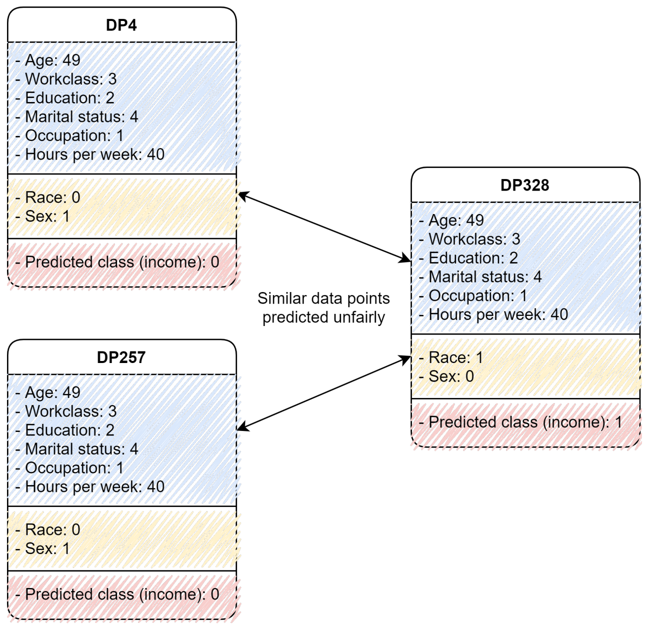

FIGURE 5.10: Similar data points from ACID predicted unfairly by two global similarity functions. One is the global similarity function leveraging on scaling of scores without subtraction and addition of one in weight calculations, while the other is using linear rankings and scoring 75% of features in each FRSM.

After presenting the fairness results, we provide the accuracy results (i.e. the percentage of true predictions compared to total number of predictions), for each of the mentioned global similarity functions, in tables 5.9 and 5.10. To complement the accuracy results, we calculated the confusion matrices and present them in figures 5.11 and 5.12. The first row of the matrices presents the true and false negatives, while the second row shows false and true positives respectively. For the Adult Census Income Dataset, a positive results is a predicted income of above 50k$, while for the Fraud Detection Dataset, it is a fraudulent case prediction. Global similarity marked *new_weights_all* is our method of calculating weights, while *new_weights_no*_1 presents the approach where the subtraction and addition is removed. Global similarities marked *old_weights*_75 and *old_weights*_100 are the weight calculation methods from [Jaiswal and Bach, 2019]. Each of the mentioned methods has a masked version where the protected features are not considered i.e. they are not scored by FRSMs.

Dataset	Accuracy of system			
	Weights calculated using scaling each features' sum of scores		Weights calculated using scaling each features' sum of scores without addition and subtraction	
Protected features masked?	No	Yes	No	Yes
ACID	82.34%	80.92%	81.13%	81.63%
FDD	76.53%	75.38%	79.61%	80%

TABLE 5.9: Accuracy of the system using the method of scaling sum of scores of all features to define the global similarity function

Dataset	Accuracy of the system			
	Weights calculated using ranking 75% of features (old weight calculation)		Weights calculated using ranking 100% of features (old weight calculation)	
Protected features masked?	No	Yes	No	Yes
ACID	81.33%	81.53%	82.64%	80.92%
FDD	66.92%	70.38%	67.3%	70.38%

TABLE 5.10: Accuracy of the system using the method of scoring a selected percentage of features' in each FRSM to calculate sum of scores and define the global similarity function

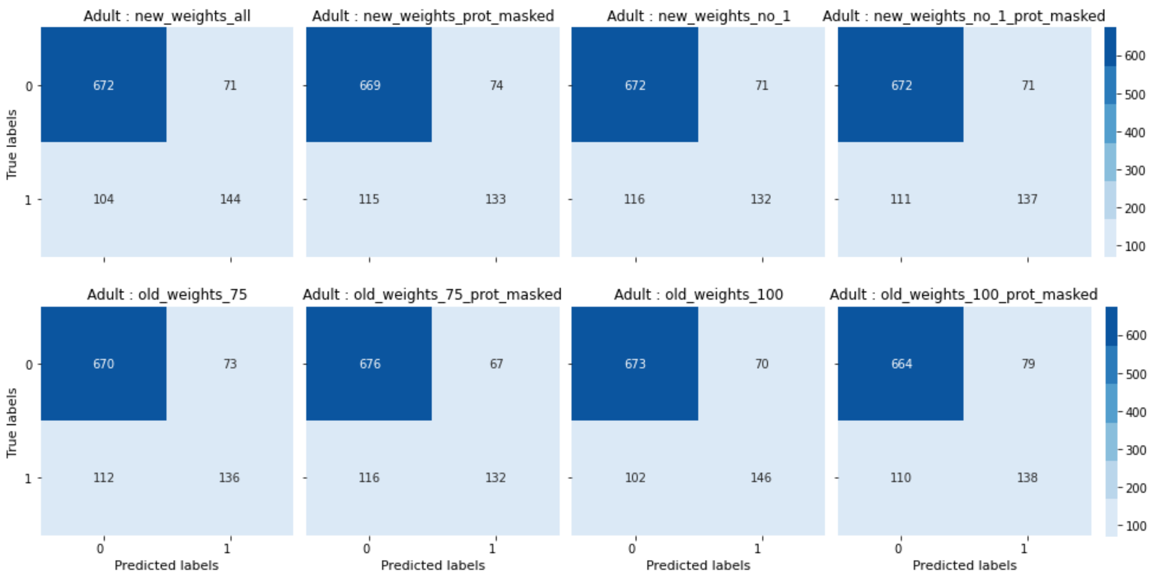

FIGURE 5.11: Confusion matrices for all used global similarity functions when applied to the Adult Census Income Dataset

5.3.6 Analysis of Results

After the datasets were cleaned from biased data, we performed retrievals using different global similarity functions on all the similar pairs of data points in the dataset. After the retrieval is returned for each data point in a certain pair, we examined whether there is a difference in the predicted class labels of the two data points. A difference indicates the existence of an unfair classification. The results of the retrievals were presented in tables 5.7 and 5.8.

The Adult Census Income Dataset had 53 pairs of similar data points in the

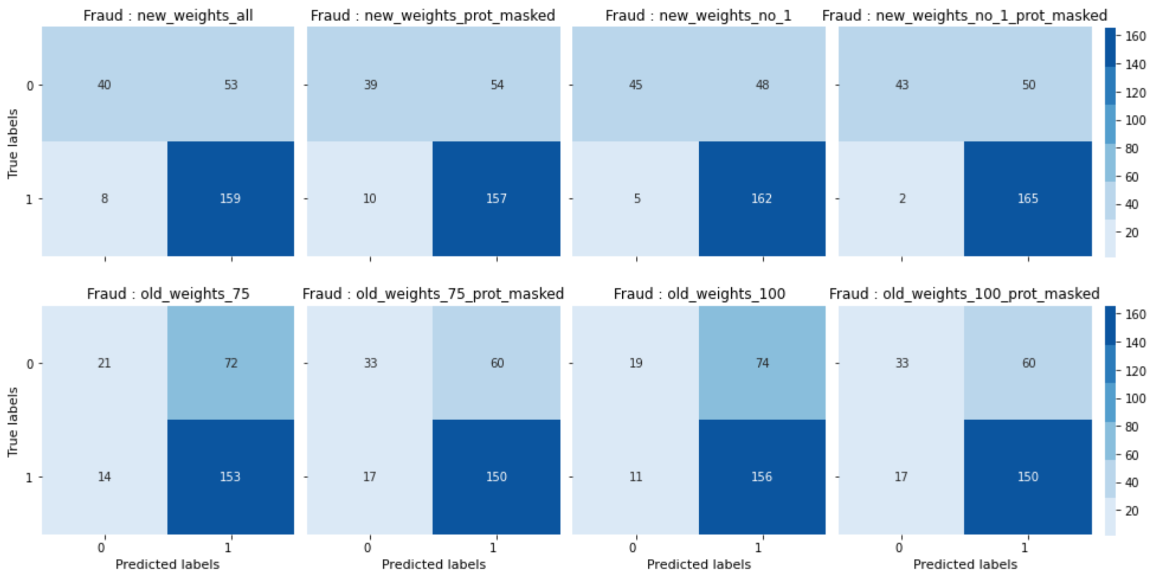

FIGURE 5.12: Confusion matrices for all used global similarity functions when applied to the Fraud Detection Dataset

test dataset, so 106 retrievals were performed for each global similarity function. The Fraud Detection Dataset contains less data points in general resulting in 11 pairs of similar data points being found, in the test dataset, or 22 retrievals being performed. Firstly, let's consider the results for non-masked versions of global similarity functions. When using our proposed approach to scaling scores inside FRSM, no individual unfairness was noticed in either dataset. The next global similarity considered used a modification to the previously explained scaling approach by removing the necessity to subtract and add one in the formula for weight calculation. While no unfairness was detected when using the FDD, this type of global similarity function produced an unfair classification of three data points, from the ACID, as seen in figure 5.10. The figure visualizes the apparent unfairness since all three data points have the same values for non-protected features while they differ on protected features' values. The same behaviour occurred when using the global similarity function which assigns linear ranks and scores 75% of top features in each FRSM. When using the same function on the FDD, even more unfairness is detected (even though the dataset is smaller in total data points), more precisely four pairs of data points were found unfairly treated. Finally, when changing the percent of features to be scored from 75% to 100% we discovered no unfairness when considering the ACID, while one pair of unfairly classified data points was found. It is important to reiterate that, in the case of the FDD, we did not consider *days_to_incident* when comparing two data points, for reasons explained in section 5.3.3. Due to this, unfairness might be avoided if domain experts were used to define a local similarity for this feature.

When considering the same four global similarity functions but with masked protected features, none induced any detectable unfairness in predicting data points that are similar. This might encourage an engineer that there is a simple solution to this issue; simply mask all protected features. While this might ensure individual fairness sometimes, it can not guarantee a fair approach. The reason for this is that protected features' values can be transferred to some other non-protected feature (referred to as a proxy). In that case, even after masking protected features' values, unfair classifications could still be present. This is explained in section 2.2.

Finally, in order to consider performance of models we looked at accuracy scores and confusion matrices for each of the used global similarity functions. The accuracy scores are presented in tables 5.9 and 5.10, while the matrices can be seen in figures 5.11 and 5.12. Since the Fraud Detection Dataset and a larger version of the Adult Census Income Dataset, were both used in [John, Vijaykeerthy, and Saha, 2020], we compare the accuracy scores to their models. The comparison indicates very similar accuracy numbers to the ones presented in their paper, in some cases, even surpassing their accuracy scores. This could be for a variety of reasons and does not indicate that our approach should be considered better in terms of accuracy. When comparing the global similarity functions presented in this thesis, accuracy scores vary when different datasets were used as well as masked approaches compared to non-masked approaches. To have a more clear understanding of which approach can ensure the highest accuracy, while also providing fair classifications, further test are required. This is discussed further in section 6.2.

Chapter 6

Discussion and Future Work

This chapter presents our concluding thoughts. Section 6.1 summarizes our results presented in chapter 5 while section 6.2 provides interesting avenues for future work.

6.1 Discussion

After presenting the related work that was done at the time of writing to research, detect and mitigate individual unfairness in ML models, we constructed two research questions. We argue that answering these will provide future researchers valuable insight into the topic.

Both research questions and goals were explained in section 1.1.

Research Question 1 *How to detect dataset bias?*

In order to answer research question 1, our goal was to propose a method that can be used to detect bias in datasets. We believe ensuring a dataset is unbiased is a necessary step in order to be able to discuss ML model fairness. Inspired by work done in [Jaiswal and Bach, 2019], we propose a method that calculates how much influence each feature has on distinguishing between different classes, i.e. on the decisions made by decision support systems. This is achieved by using a method for scoring the feature relevance to value how influential each feature is in decision making of an AI/ML system. We decided to combine scores from different feature relevance scoring methods in order to balance the statistical results that different methods use.

The results from applying the proposed bias detection in the pre-processing stage to the datasets show that most of the datasets contain bias to a certain degree, as seen in table 5.2. In order to evaluate this method, we chose one dataset that was considered biased and created a modified version where all the biased data points were removed. Finally, we applied the pre-processing method to this cleaned dataset and compared the results with the original dataset. The results, shown in table 5.5, indicate that the method detects no bias in the cleaned dataset, with significantly less influence values for protected features. We argue that this result indicates that the method is able to detect bias in datasets. To argue this with more certainty, more datasets should be tested.

After reviewing the results from section 5.2.2, we argue that we successfully answered research question 1 and achieved our goal of proposing a method to detect dataset bias.

Our contribution is in providing a pre-processing method that can be used to test dataset bias, when talking in the context of individual fairness. By providing this method, we successfully answered RQ1. While not revolutionary, with additional testing, we believe this pre-processing method could be widely used before deciding on a dataset for a ML model. The advantages of the approach are that it is quick in its calculations and portrays the influence of features on the classification in an easy to understand format.

Research Question 2 *Can case-based reasoning systems using automated weight calculations be used for ensuring individual fairness?*

When attempting to answer research question 2, we set a goal of creating a case-based reasoning system that can be used for ensuring individual fairness. For this purpose, we examined the framework explained in [Jaiswal and Bach, 2019]. The intrigue of their approach is the automatising of weight calculations. We believed this approach could remove potential human bias when constructing the models, if we ensure an unbiased dataset is presented. To construct the model, we used two datasets that were evaluated for answering research question 1. They were stripped of all biased data points, so we can state, with a high certainty, they do not contain any biased data points. When considering the automated weight calculation approach from [Jaiswal and Bach, 2019], we noticed some improvements that could be implemented. We argue, these improvements help with weights more accurately representing the scores that the feature relevance scoring methods provide. This approach is explained in detail in section 4.1.4. In order to determine if this approach improves or worsens the performance of the CBR model, we constructed the same model that was evaluated in [Jaiswal and Bach, 2019], and added the new approach as an additional global similarity function. To have the same environment, we tested using the same dataset as used in their paper, the Car dataset [Bohanec, 1997]. This dataset doesn't have any protected features, so only comparing performance was considered. To avoid complicating the discussion, the individual evaluation methods are not explained. More in depth explanations about their evaluation methods can be found in [Jaiswal and Bach, 2019]. Firstly, we show the confusion matrices of the three original global similarity functions described in the paper and our additional global similarity function marked as *new_wt_all*.

Next, in table 6.1, we present the scores of global similarity functions for different evaluation metrics, collected from 10 runs of the system with k values from 1 to 10. Figure 6.2 represents the 10-fold cross-validation for the 10 runs, using all four global similarity functions.

In order to more easily provide a visualization of the metrics' performance, figure 6.3 presents the max recall of the global similarity functions, for values of k from 1 to 10.

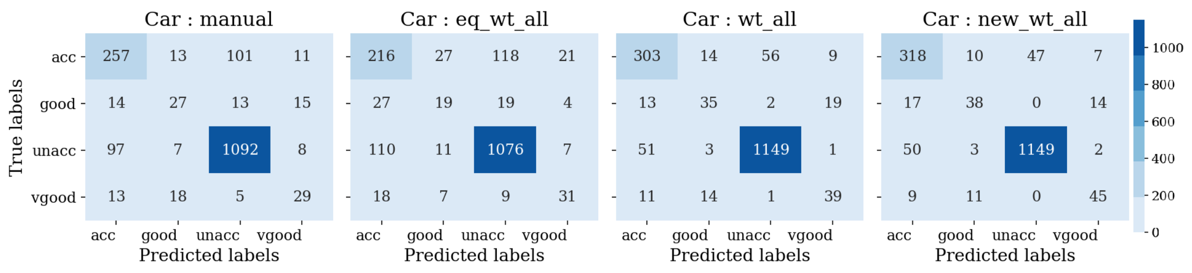

FIGURE 6.1: Confusion matrices for the different global similarity functions

Global similarity function (using which type of weight calculation)	F1 score	Precision	Recall	Accuracy
Manual weight calculations by domain experts	0.6	0.61	0.6	0.82
Equal weights for all the featues	0.56	0.56	0.55	0.78
Calculating weights by scoring 100% of features in each FRSM	0.71	0.71	0.71	0.89
New approach using scaling scores in each FRSM	0.76	0.76	0.76	0.9

TABLE 6.1: Presenting the mean for different evaluation metrics from 10 runs

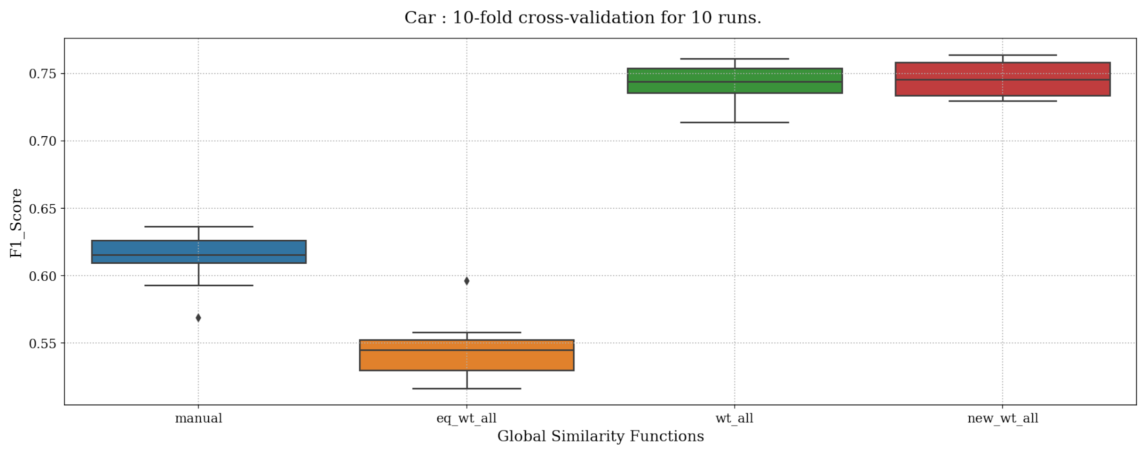

FIGURE 6.2: The 10-fold cross-validation for 10 runs, using all four global similarity functions

After reviewing the results of the evaluation, shown in figures 6.1, 6.2 and 6.3 and table 6.1, we conclude that our approach provides the same if not better performance using all the different evaluation metrics. Knowing this, we were confident of testing our approach in regards to ensuring fairness of ML models, as it does not degrade performance.

The fairness tests were done using an approach inspired by [John, Vijaykeerthy, and Saha, 2020]. Details of the approach were explained in sections 4.2.4 and 4.2.5. The results for different global similarity functions were presented in section 5.3.5.

The first observation one can make from tables 5.7 and 5.8 is that versions of global similarity functions with masked protected features do not produce any unfairly classified data points. While this is the case with the datasets

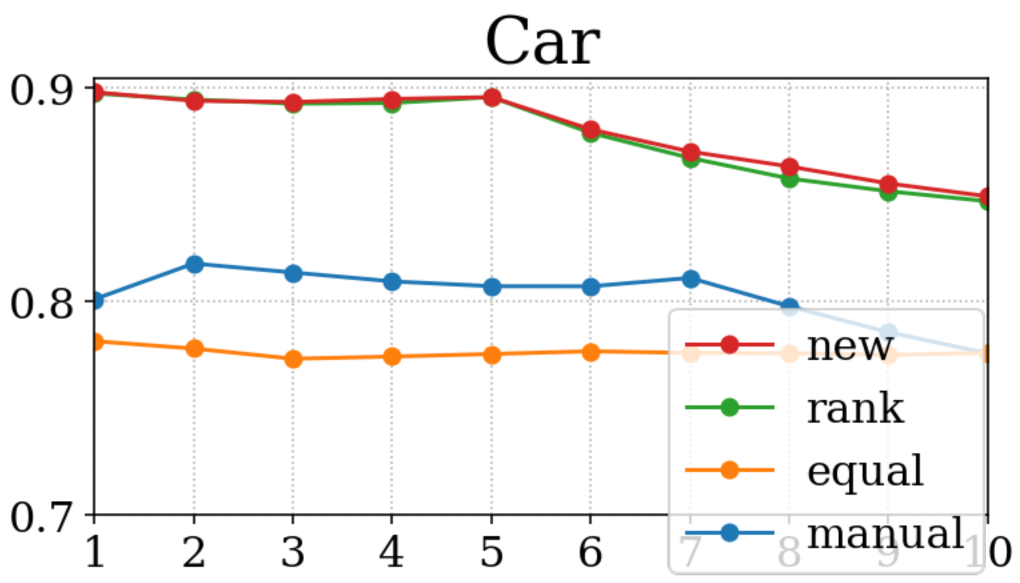

FIGURE 6.3: Max recall of the system with the Car dataset when using different global similarity functions

we tested, it is important to consider that this might not always be achieved. Unfair classification can be achieved even when protected features are not used, by using proxy features. The notion of protected and proxy features is explained in section 2.2.

We argue that after examining the results in section 5.3.5, our automated approach to weight calculation by scaling scores in feature relevance scoring methods can be used to ensure fairness, if an unbiased dataset is provided. Our approach also shows improved results in terms of accuracy to all similarly constructed global similarity functions, as seen in tables 5.9 and 5.10. While we can not, with certainty, state that our approach has improved performance, we argue that it shows promising signs and should be researched and developed further. Some ventures for possible future work are mentioned in section 6.2.

Our contribution is providing a proof of concept that case-based reasoning systems can make individually fair decisions. By providing this proof, we successfully answered RQ2. On top of that, we propose an automated approach to weight calculations that does not induce individual unfairness, if the dataset is unbiased. The advantages of using our approach is that there is less need for human input when constructing the model. We argue this is valuable since it reduces the possibility of human biased being introduced in the in-processing stage of the ML pipeline. One disadvantage to using this approach, which goes for all case based reasoning systems, is the necessity of defining local similarity functions. While this is certainly noticeable, we showed that even with having no domain knowledge, and using default local similarity function, a reasonable accuracy score can be achieved. On top of this, we argue that defining local similarity functions is easier than defining global similarity function, which is the part we automated. The reason is that for local similarity functions, it is important to identify which values should be similar for a particular feature, while for global similarity functions, one needs to put a quantifiable value on how valuable a certain feature should be when classifying a data point, which is usually difficult for humans.

6.2 Future Work

As mentioned throughout chapters 4 and 5, there are different avenue for future work that should be explored to further develop the proposed pre-processing method and CBR system.

Firstly, in order to have a stronger case for the pre-processing method accurately detecting dataset bias, more datasets should be tested in the same manner as explained in section 4.1.8. Also, considering adding different feature relevance scoring methods in order to improve accuracy of the influence results could be an interesting future task.

As far as the proposed CBR system using an automated weight calculation, there are several input parameters that should be tested in order to potentially improve both accuracy and fairness results. The first input parameter that should be tested extensively is using different datasets. If computational resources allow, we propose performing experiment 2 on the full Adult Census Income Dataset (as well as other, suitable datasets that we didn't use) and comparing the results with the ones we have shown in section 5.3.5. Another input parameter that should be considered is the k parameter. We believe, given more time, we might be able to provide an automated calculation of k that will give optimal accuracy results, regardless of the dataset used. One more exploration in the model construction that we propose is to construct different local similarity functions for the existing global similarity function and exploring if these can also be automated to remove any need for a domain expert. This exploration should be extended to the fairness verifier. We argue that the feature partitions and their respective thresholds should be adjusted in such a way that would allow detecting unfairness when similar people are considered, as supposed to identical (barring the protected features) in this thesis.

Finally, a completely different verifier could be used to test the fairness of the CBR system. As mentioned in section 4.2.6, while our verifier did not find any unfairness, we can't label the system unfair. This is due to the fact that we queried only a finite number of the data points from the test dataset, and not all paths have been covered. To solve this issue, exploring the black box testing explained in [Aggarwal et al., 2019] might provide with more assurance.

Bibliography

Aamodt, Agnar and Enric Plaza (1994). "Case-based reasoning: Foundational issues, methodological variations, and system approaches". In: *AI communications* 7.1, pp. 39–59.

Aggarwal, Aniya et al. (2019). "Black Box Fairness Testing of Machine Learning Models". In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2019. Tallinn, Estonia: Association for Computing Machinery, pp. 625–635. ISBN: 9781450355728. DOI: `10.1145/3338906.3338937`. URL: `https://doi.org/10.1145/3338906.3338937`.

Angwin, J. et al. (2016). "Machine Bias". In: URL: `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`.

Barocas, Solon and Andrew D Selbst (2016). "Big data's disparate impact". In: *Calif. L. Rev.* 104, p. 671.

Bellamy, Rachel K. E. et al. (Oct. 2018). *AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias*. URL: `https://arxiv.org/abs/1810.01943`.

Berk, Richard et al. (2021). "Fairness in Criminal Justice Risk Assessments: The State of the Art". In: *Sociological Methods & Research* 50.1, pp. 3–44. DOI: `10.1177/0049124118782533`. URL: `https://doi.org/10.1177/0049124118782533`.

Binns, Reuben (2020). "On the apparent conflict between individual and group fairness". In: *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 514–524.

Bohanec, M. (1997). "Car evaluation database". In: URL: `https://archive.ics.uci.edu/ml/%20datasets/Car+Evaluation`.

Ceriani, Lidia and Paolo Verme (Sept. 2012). "The origins of the Gini index: extracts from Variabilità e Mutabilità (1912) by Corrado Gini". In: *The Journal of Economic Inequality* 10.3, pp. 421–443. DOI: `10.1007/s10888-011-9188-x`. URL: `https://ideas.repec.org/a/kap/jecinq/v10y2012i3p421-443.html`.

Chouldechova, Alexandra and Aaron Roth (2018). *The Frontiers of Fairness in Machine Learning*. arXiv: `1810.08810 [cs.LG]`.

Danks, David and Alex John London (2017). "Algorithmic Bias in Autonomous Systems." In: *IJCAI*. Vol. 17, pp. 4691–4697.

Demšar, Janez et al. (2013). "Orange: Data Mining Toolbox in Python". In: *Journal of Machine Learning Research* 14.35, pp. 2349–2353. URL: `http://jmlr.org/papers/v14/demsar13a.html`.

Dieterich, William, Christina Mendoza, and Tim Brennan (2016). “COMPAS Risk Scales : Demonstrating Accuracy Equity and Predictive Parity Performance of the COMPAS Risk Scales in Broward County”. In:

Dressel, Julia and Hany Farid (2018). “The accuracy, fairness, and limits of predicting recidivism”. In: *Science Advances* 4.1. DOI: 10.1126/sciadv.aao5580. eprint: https://advances.sciencemag.org/content/4/1/eaao5580.full.pdf. URL: https://advances.sciencemag.org/content/4/1/eaao5580.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

Dwork, Cynthia et al. (2012). “Fairness through Awareness”. In: ITCS ’12. Cambridge, Massachusetts: Association for Computing Machinery, pp. 214–226. ISBN: 9781450311151. DOI: 10.1145/2090236.2090255. URL: https://doi.org/10.1145/2090236.2090255.

Feldman, Michael et al. (2015). “Certifying and Removing Disparate Impact”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’15. Sydney, NSW, Australia: Association for Computing Machinery, pp. 259–268. ISBN: 9781450336642. DOI: 10.1145/2783258.2783311. URL: https://doi.org/10.1145/2783258.2783311.

Finnie, Gavin and Zhaohao Sun (2002). “Similarity and metrics in case-based reasoning”. In: *International journal of intelligent systems* 17.3, pp. 273–287.

Hardt, Moritz et al. (2016). “Equality of Opportunity in Supervised Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf.

Heilprin, L. B. (1960). “Information Theory and Statistics. Solomon Kullback. Wiley, New York; Chapman and Hall, London, 1959. xvii + 395 pp. Illus.” In: *Science* 131.3404, pp. 917–918. ISSN: 0036-8075. DOI: 10.1126/science.131.3404.917-b. eprint: https://science.sciencemag.org/content/131/3404/917.3.full.pdf. URL: https://science.sciencemag.org/content/131/3404/917.3.

Hofmann, Dr. Hans (1994). *UCI Machine Learning Repository*. URL: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data).

Jaiswal, Amar and Kerstin Bach (2019). “A data-driven approach for determining weights in global similarity functions”. In: *International Conference on Case-Based Reasoning*. Springer, pp. 125–139.

John, Philips George, Deepak Vijaykeerthy, and Diptikalyan Saha (2020). *Verifying Individual Fairness in Machine Learning Models*. arXiv: 2006.11737 [cs.LG].

Joseph, Matthew et al. (2016). *Fairness in Learning: Classic and Contextual Bandits*. arXiv: 1605.07139 [cs.LG].

Kleinberg, Jon, Sendhil Mullainathan, and Manish Raghavan (2016). “Inherent trade-offs in the fair determination of risk scores”. In: *arXiv preprint arXiv:1609.05807*.

Kohavi, Ronny and Barry Becker (1996). *UCI Machine Learning Repository*. URL: https://archive.ics.uci.edu/ml/datasets/adult.

Mehrabi, Ninareh et al. (2019). "A survey on bias and fairness in machine learning". In: *arXiv preprint arXiv:1908.09635*.

Mothilal, Ramaravind K, Amit Sharma, and Chenhao Tan (2020). "Explaining machine learning classifiers through diverse counterfactual explanations". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 607–617.

Pearson, Karl (1900). "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50.302, pp. 157–175. DOI: 10.1080/14786440009463897. URL: https://doi.org/10.1080/14786440009463897.

Quinlan, J. Ross (1986). "Induction of decision trees". In: *Machine learning* 1.1, pp. 81–106.

Robnik-Sikonja, Marko and Igor Kononenko (Feb. 2000). "An adaptation of Relief for attribute estimation in regression". In: *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*.

Ross, KA and CRB Wright (1992). *Discrete Mathematics, 193*.

Rudin, Cynthia and Kiri L Wagstaff (2014). *Machine learning for science and society*.

Speicher, Till et al. (July 2018). "A Unified Approach to Quantifying Algorithmic Unfairness". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. DOI: 10.1145/3219819.3220046. URL: http://dx.doi.org/10.1145/3219819.3220046.

Stahl, Armin and Thomas R. Roth-Berghofer (2008). "Rapid Prototyping of CBR Applications with the Open Source Tool MyCBR". In: *Proceedings of the 9th European Conference on Advances in Case-Based Reasoning*. ECCBR '08. Trier, Germany: Springer-Verlag, pp. 615–629. ISBN: 9783540855019. DOI: 10.1007/978-3-540-85502-6_42. URL: https://doi.org/10.1007/978-3-540-85502-6_42.

Sun, Zhaohao, Gavin Finnie, and Klaus Weber (2004). "Case base building with similarity relations". In: *Information Sciences* 165.1, pp. 21–43. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2003.09.020. URL: https://www.sciencedirect.com/science/article/pii/S0020025503003748.

Tang, Jiliang, Salem Alelyani, and Huan Liu (2014). "Feature selection for classification: A review". In: *Data classification: Algorithms and applications*, p. 37.

Tse, Lao (2020). *Kaggle*. URL: https://www.kaggle.com/laotse/credit-card-approval.

Yu, Lei and Huan Liu (Jan. 2003). "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution". In: vol. 2, pp. 856–863.

Zemel, Rich et al. (2013). "Learning Fair Representations". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, pp. 325–333. URL: http://proceedings.mlr.press/v28/zemel13.html.