Detection of Unfairly Treated Groups in Classification and Ranking

Yuval Moskovitch
University of Michigan
yuvalm@umich.edu

Jinyang Li
University of Michigan
jinyli@umich.edu

H. V. Jagadish
University of Michigan
jag@umich.edu

ABSTRACT

Machine learning (ML) tools are widely used in many real-life everyday applications. With their ubiquitous use in recent years, we also witness an increase in the reported cases where these tools discriminate unfairly. This, in turn, has given rise to increasing interest in the study of algorithmic fairness. Fairness definitions usually refer to a given "protected group" in the data, which is defined based on the values of some sensitive attributes. Given a protected group, confirming algorithmic fairness is a simple task. However, the groups discriminated against may not always be known in advance.

In this paper, we study the problem of detecting groups that are treated unfairly in classification and ranking algorithms, eliminating the need to pre-define sensitive attributes. The number of such groups possible can be exponential, making the problem hard. In the classification context, we propose a solution that employs pruning to significantly reduce the search space. Then we leverage this method to efficiently tackle the problem in the context of ranking, presenting an additional optimization utilizing a local search. We conclude with an extensive experimental study, demonstrating the scalability and benefits of our approach, and demonstrating the usefulness of the proposed optimizations.

PVLDB Reference Format:
Yuval Moskovitch, Jinyang Li, and H. V. Jagadish. Detection of Unfairly Treated Groups
in Classification and Ranking. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

Data science has become a vital tool for business, and so is the use of Machine Learning (ML) models in everyday applications. ML models are utilized to make decisions, typically by classification and ranking algorithms. For instance, using ML models, data-driven tools are now used to decide whether to grant a loan [1], whether to hire [2] and even whether to parole convicted criminals [6]. With the increasing use of data-driven tools, we also witness a large number of cases where these tools discriminate unfairly against some groups.

The increasing impact of data-driven methods on society and their effect on human life, has given rise to increasing interest in the study of algorithmic fairness. E.g., the problem of fairness in classification was studied in [10, 14, 16, 18, 20, 31] and [13, 30, 32] studied the problem in the context of ranking and recommendation. Fairness definitions usually refer to a given "protected group" in the data, which is defined based on the values of some sensitive attributes (e.g., gender, race, age, or combinations thereof), usually based on societal history of discrimination. When developing a data-driven system, a data scientist may wish to ensure the algorithm is fair according to a given fairness measure. Given a protected group, confirming algorithmic fairness is a simple task for any mathematical definition of fairness. However, "non-standard" protected groups cannot always be specified in advance, and such groups may be overlooked when analyzing the performance of a system.

Example 1.1. An example in the field of education was published recently in the New York Times [3]. The International Baccalaureate (IB) is a global standard of educational testing that allows U.S. high-school students to gain college credit. The final exams, which are a major factor in the students' scores, were canceled due to the COVID-19 pandemic. Instead, students were assigned grades based on a predictive model. As a result, high-achieving students from poor school districts were severely hurt, because the model placed great weight on school quality. For instance, students from low-income families were predicted to fail the Spanish exam, even when they were native Spanish speakers. Many of them had studied for IB hoping to save thousands of dollars on tuition by earning college credits with their scores.

In this case, the group of students treated unfairly, Hispanic students from poor school districts, is not a standard protected group. One of the sources of bias in the results was the use of historical IB results of the schools. However, using the school ID to define the protected group is not a natural or intuitive choice. Moreover, even if we consider the group of Hispanic students as a protected group, we may not find any unfairness, since this unfairly treated subgroup is only a small fraction of all Hispanic students.

Several recently developed tools [9, 11] allow users to assess fairness of ML systems, and even assist in mitigating the bias and provide explanations. However they focus on investigating only user-specific protected groups. When the group definition is unknown in advance, and can be defined by any value combination, searching for groups that are treated unfairly may be computationally prohibitive. In Section 6.3, we show that the number of different attributes involved in the definition of unfairly treated groups may be considerable, over 10 different attributes, which in

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

turn, may be used to define tens of thousands of different groups. Moreover, we show that the groups can be largely diverse in the attributes used to define them.

Different fairness measures are typically designed to capture case-appropriate properties: different definitions may be used in different use-cases. For instance, one natural fairness definition considers the accuracy among different groups. According to this definition, a classifier is fair if different groups in the data (e.g., Hispanic students from low-income families) have the same overall accuracy in prediction as other groups. Another plausible definition takes into account the false positive error rate, for instance, students who failed the exam but was falsely predicted to pass by the algorithm. Intuitively, by this measure the classifier should give similar results for all students who fail the exam. When considering different fairness measures, groups that are treated unfairly may vary in the set of sensitive attributes used to define them.

We will use the following example as our running example, to demonstrate the ideas presented in the paper.

Example 1.2. The Student Performance Data Set [17] contains information from two Portuguese secondary schools from the Alentejo region of Portugal, Gabriel Pereira (GP) and Mousinho da Silveira (MS). The data was collected during the 2005-2006 school year and it contains the performance of 1044 students in the Math and the Portuguese language exams, along with demographic, social and school related information. Figure 1 depicts a sample from the data with the attributes: gender, school, address (urban or rural), and failures (number of past class failures). The grade attribute depicts the students' grade in a scale of $0 - 20$. The prediction attribute describes the prediction, "pass" (grade > 10) or "fail" (grade ≤ 10) of a predictive model (a classifier M), and the Rank attribute is the output of a ranking algorithm (R) that ranks students by their grades (higher grades are preferred). In the case of similar grades, students with fewer failures are ranked higher.

The overall accuracy of the classifier M (based on the given data) is 0.875, however for students from the GP school with an urban (U) address the accuracy is only 0.67, significantly lower than any other group in the data (that may be defined by values of the different attributes). In this toy example, this difference could be just by chance. However, this sort of difference on a large data set is likely to indicate systemic bias. We would like to find such groups, to be able to fix such problems, keeping in mind that urban students and the "GP" school may not have been already identified as potential targets of discrimination.

We study in this paper the problem of detecting groups of data items that are treated unfairly by a given ML model. In other words, we want to let the data speak to (potential) unfairness, without requiring a human modeler to identify protected attributes ahead of time. We measure unfairness with known mathematical definitions commonly used to assess fairness in ML models. For instance, if we consider fairness as the accuracy among different groups, a classifier is fair if different groups in the data have the same overall accuracy in prediction as other groups. Our goal is to allow users to efficiently evaluate their ML models with respect to their preferred fairness definition (or the one that meets their needs) among these. Related problems have recently been studied in [12, 15, 26] (see further discussion in Sections 6.4 and 7). We next outline our main contributions.

Problem formulation. We formally define the problem of detecting groups treated unfairly by a decision system such as a classifier or a ranker. Groups are defined using value assignment to a set of attributes. The notion of unfairness is directly derived from the way we define fairness, and the model is treated as a black box, making the problem to be model agnostic. For classification, we consider various group fairness definitions that are based on the prediction and actual outcomes [31] (see Section 2). Given a fairness model, we quantify the *degree of unfairness* with respect to a given group as the difference between the overall fairness measure of the dataset and the measure of this group. The goal is to report meaningful, substantial groups (i.e., large enough) such that their degree of unfairness exceeds a given threshold. In the context of ranking we start with the fundamental definition of [13] which restricts the number of tuples in the top-k from each group in the data using upper and lower bounds. The goal is to report groups such that their number of tuples in the top-k does not lie within the given bounds for a given range of possible k's. We then consider the prominent class of fairness measures utilizing proportional representation. Intuitively, the representation of each group in the top-k should be proportional to its size in the data. We show that no polynomial algorithm exists for those problems (Section 3).

Detection of unfairness. We present algorithms for the problem of finding the set of all substantial unfairly treated groups in both classification and ranking. For classification (see Section 4), bearing in mind the complexity of the problem, we focus on optimizing the search. Our approach utilizes the notion of pattern graph presented in [8]. We show how to efficiently traverse the graph in a top-down fashion in order to find the unfairly treated groups, while pruning the search space when possible. We then turn to the problem of detecting unfairness in ranking (Section 5). A naive approach would be to apply the top-down search, similarly to the search in the problem for classification, for each k in the given range. We present an optimized solution, which relies on the fact that the set of top-k and top-$(k+1)$ tuples differ by a single tuple. As a result, the search spaces for succeeding k values are typically very similar. The optimized solutions utilize this observation to avoid parts in the search tree when possible.

Experimental study (Section 6). We complement our algorithmic development with an experimental study. We start by evaluating the performance and properties of the algorithms, i.e., the scalability and parameters setting effect (fairness measure, groups' size and unfairness degree). We examine the effect of the different parameters on the running time using five real-world datasets. To illustrate the benefit of our solution, we applied our algorithm on the COMPAS dataset, reproducing the discrimination reported in [6]. Our result reveals further issues with respect to other groups in the data. We show that different definition may lead to different (non-overlapping) groups in the data, demonstrating the benefits of eliminating the protected group definition in advance. Finally, we compare our proposed method with the existing solutions in the context of classification, showing the differences among them.

We overview related work in Section 7 and conclude in Section 8.

#	Gender	School	Address	Failures	Grade	Prediction	Rank
1	F	MS	R	1	11	Pass	8
2	M	MS	R	1	15	Pass	3
3	M	GP	U	1	8	Pass	10
4	M	GP	U	2	4	Fail	16
5	M	MS	R	0	19	Pass	2
6	F	MS	U	1	4	Fail	15
7	F	GP	R	1	7	Fail	11
8	M	GP	R	1	6	Fail	13
9	F	MS	R	0	14	Pass	4
10	F	MS	R	2	7	Fail	12
11	M	MS	R	2	13	Pass	6
12	F	GP	U	0	20	Pass	1
13	F	GP	U	2	12	Fail	7
14	M	MS	U	1	13	Pass	5
15	F	GP	U	1	5	Fail	14
16	M	GP	U	0	9	Fail	9

Figure 1: Students' grades ML prediction model for sample data from the Student Performance Data Set[17]. The ML prediction is wrong for the highlighted rows (false positive in green, false negatived in red). The Rank column depicts their ranking based on the grade and number of past failures.

2 PRELIMINARIES

Many definitions of fairness have been proposed in the literature. Our attempt in this paper is to cover as many of these definitions as possible. In this section, we provide necessary background on the concept of fairness in classification and ranking. Then, in the next section, we formally define the problem of unfairness detection.

Group fairness in classification. We consider definitions based on comparing the model prediction and actual outcome. For a binary classifier, the (actual or predicted) outcome can be either positive or negative. For instance, a passing grade is a positive outcome, and a failing grade is a negative outcome. Considering the actual and predicted outcome together, we have four possible cases: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). The relative counts of these four groups can be used to define several intuitive measures of fairness. We next provide a brief overview of such definitions[1] (see [31] for more details).

Overall accuracy equality A classifier satisfies this definition if all groups have similar prediction accuracy.

Predictive parity The fraction of correct positive prediction $\frac{TP}{TP+FP}$ should be similar for all groups.

False positive error rate balance (predictive equality) The probability of a subject in the actual negative class to have a positive predictive value $FPR = \frac{FP}{FP+TN}$ is similar for all groups.

False negative error rate balance (equal opportunity) Similar to the above, but considers the probability of falsely classifying subject in the positive class as negative $FNR = \frac{FN}{TP+FN}$.

Equalized odds Combine the previous two definitions. All groups should have both similar false positive error rate $\frac{FP}{FP+TN}$ and false negative error rate $\frac{FN}{TP+FN}$.

Conditional use accuracy equality All groups should have similar probability of subjects to be accurately predicted as positive $\frac{TP}{TP+FP}$ and accurately predicted as negative $\frac{TN}{TN+FN}$.

Treatment equality This definition considers the ratio of error. A classifier satisfies it if all groups have similar ratio of false negatives and false positives.

Different definitions can lead to different outcomes, as we next demonstrate.

Example 2.1. Continuing with our running example, we have already discussed how accuracy in prediction was lower for students from the GP school with an urban address. Another fairness measure, known as equal opportunity, focuses on the false positive error rate ($FPR = \frac{FP}{FP+TN}$). Intuitively, by this measure the classifier should give similar results for all students who fail the exam. In this case a high FPR indicates fairness issues. In our example male students with a single previous failure have a high FPR (0.5) compared with the overall FPR (0.125).

Group fairness in ranking. The problem of fairness in ML was also studied in the context of ranking [27, 28, 32]. A fundamental definition of fairness in ranking presented in [13] uses constraints over the representations of different groups in the top-k ranked items. They use an upper bound U_{kl} and a lower bound L_{kl} over the number of items with the property l (i.e., a protected group) in the top-k positions of the ranking. Then a ranking algorithm is fair by definition of [13], if the number of selected items from the protected group in the top-k lies within the given boundaries.

Example 2.2. Consider again the dataset given in Figure 1 and the ranker whose result is presented in the Rank column. Consider a lower bound of 2 over the number of students from each school for $k = 5$ (i.e., $L_{5,school=MS} = L_{5,school=GP} = 2$). In this case, among the top 5 students, only one is from the GP school, thus the ranker does not satisfy the constraints.

Another prominent class of fairness measure in ranking considers the proportional representation of different groups in the top-k ranked items (see, e.g [32]). Intuitively, these definitions can be seen as variants of the fairness definition of [13] such that for each group g, and each k, the bounds on the number of occurrences of items from g in the top-k ranked items are defined with respect to the size of g in the dataset.

Example 2.3. Continuing Example 2.2, the total number of students from each school (MS and GP) is 8. The total dataset size is 16, thus a proportionate representation of each school in the top-5 items should be roughly $5 \cdot \frac{8}{16} \approx 2$.

3 PROBLEM DEFINITION

We assume the data is represented using a single relational database, and that the relation's attribute values are categorical. Where attribute values are drawn from a continuous domain, we render them categorical by bucketizing them into ranges: very commonly done in practice to present aggregate results. In fact, we may even group categorical attributes into fewer buckets where the number of individual categories is very large. We first define the notion of a pattern that is used to represent groups in the data.

Definition 3.1 (Patterns). Let D be a database with attributes $\mathcal{A} = \{A_1, \ldots, A_n\}$ and let $Dom(A_i)$ be the active domain of A_i for $i \in [1..n]$. A *pattern* p over D is the set of $\{A_{i_1} = a_1, \ldots, A_{i_k} = a_k\}$

[1]The definitions given in [31] consider two groups (protected/unprotected groups defined using a sensitive attribute). We generalize the definitions to fit our problem of detecting unfairly treated groups.

where $\{A_{i_1}, \ldots, A_{i_k}\} \subseteq \mathcal{A}$ and $a_j \in Dom(A_{i_j})$ for each A_{i_j} in p. We use $Attr(p)$ to denote the set of attributes in p.

We say that a tuple $t \in D$ *satisfies* a pattern p if $t.A_i = a_i$ for each A_i appearing in p. The *size* $s_D(p)$ of a pattern p is then the number of tuples in D that satisfy p.

Example 3.2. Consider the dataset given in Figure 1. p = {School = GP, Address = U}, is an example of a pattern. Tuples 3, 4, 12, 13, 15 and 16 satisfy p and thus $s_D(p) = 6$.

Unfairness in classification. We now define the fairness measure of a pattern p (i.e., of a group of tuples represented by p). For this, we consider the fairness definitions given in Section 2 and define the fairness measure of a pattern with respect to a classifier M.

Definition 3.3. Let f be a fairness definition, M be a classifier, and D be a dataset. We use $f_M(D)$ to denote the fairness value f of M with respect to D. Given a pattern p, the fairness measure of p is then defined as $f_M(\{t \in D \mid t \text{ satisfies } p\})$. Overloading notation, we use $f_M(p)$ to denote the fairness measure of p.

Example 3.4. For the overall accuracy fairness measure f, $f_M(D)$ is the overall accuracy of M over D. In our running example $f_M(D) = 0.875$. For the pattern p = {School = GP, Address = U} we get $f_M(p) = \frac{4}{6} = 0.67$, since tuples 3 and 13 are misclassified (out of 6 that satisfy p).

Our goal is to detect groups that are treated unfairly (according to a given definition) by a given classifier. Groups are represented using patterns and we can characterize each group in the data using two parameters, its *size* and *degree of unfairness*. The degree of unfairness denotes the difference between the overall fairness measure and the group's fairness measure. Namely, given a dataset D, a classifier M, and a fairness measure f, for a group represented by a pattern p, the degree of unfairness is defined as $\Delta_f(p) = |f_M(D) - f_M(p)|$. A high $\Delta_f(p)$ value indicates that tuples satisfying p are treated differently by M. Given a model M and a threshold over the degree of unfairness $\Delta\tau_f$, we say that p is a *most general* pattern such that $|f_M(D) - f_M(p)| \geqslant \Delta\tau_f$ if $\forall p' \subsetneq p$, $|f_M(D) - f_M(p')| < \Delta\tau_f$. We are now ready to formally define our problem.

Problem 3.5 (Unfairly Treated Patterns). *Given a database D, a classifier M, a fairness definition f, a threshold over the degree of unfairness $\Delta\tau_f$, and a size threshold[2] τ_s, find all most general patterns with size $\geqslant \tau_s$ such that $|f_M(D) - f_M(p)| \geqslant \Delta\tau_f$.*

Intuitively, we wish to avoid reporting "very specific" description of tuples that are mis-classified, and provide the user with a concise set of properties that characterise meaningful groups that are treated unfairly. This information allows users to gain a high level description of the fairness properties for different groups in the data, and indication for potential pitfalls of the model. We demonstrate this intuition using real dataset in Section 6.4. The problem is defined with respect to a generic classifier M, making it model agnostic. While there are typically far fewer most general patterns that are treated unfairly than the set of all patterns that are treated unfairly, in the worst case, their numbers can be exponential.

[2]We use an absolute value as the size threshold. Equivalently it may be defined as a fraction of the dataset size.

Theorem 3.6. *Given a dataset and a classifier, no polynomial time algorithm can guarantee the enumeration of the set of all most general unfairly treated patterns.*

The proof is by construction of an example with exponential number of unfairly treated patterns. See details in Appendix A.

Unfairness in Ranking. In a similar fashion, we define the problem of unfairness in ranking. The fairness definition of [13] restricts the count of different patterns (i.e., groups) in the top-k using upper and lower bounds. According to this definition, unfairness occurs either when the size of a pattern l exceeds the upper bound U_{kl} or falls below the lower bound L_{kl} among the top-k tuples for some k. We consider two variants of unfairness in ranking. The first, simply utilizes the definition of [13]. As in the classification problem, we eliminate the requirement to define l in advance, and use U_k and L_k to denote the upper and lower bound respectively, on every pattern in the top-k ranked tuples of a given ranker. Our goal is then to report the patterns that their count does not lie within the given bounds for a given range of possible k's. In what follows we use $s_{R^k(D)}(p)$ to denote the size of p in the top-k result of R on D.

Problem 3.7 (Top-k Ranking Bounds Unfairness). *Given a database D, a ranker R, a size threshold τ_s, a range $[k_{min}, k_{max}]$ and lower bounds L_k and upper bounds U_k for each $k_{min} \leqslant k \leqslant k_{max}$, find for each $k_{min} \leqslant k \leqslant k_{max}$, all most general patterns p with size $\geqslant \tau_s$ such that $s_{R^k(D)}(p) < L_k$ or $s_{R^k(D)}(p) > U_k$.*

Following the line of works on proportional representation, we consider another problem definition by further refining the above definition to account for the groups' sizes in the dataset. Intuitively, the number of items from each group in the top-k ranked items should be proportionate to the group's representation in the data.

Problem 3.8 (Top-k Ranking Proportional Representation Unfairness). *Given a database D, a ranker R, a size threshold τ_s, a range $[k_{min}, k_{max}]$, find for each $k_{min} \leqslant k \leqslant k_{max}$, all most general patterns p with size $\geqslant \tau_s$ such that $s_{R^k(D)}(p) < \alpha \cdot s_D(p)\frac{k}{|D|}$ or $s_{R^k(D)}(p) > \beta \cdot s_D(p)\frac{k}{|D|}$ for $\alpha < \beta \in \mathbb{R}$.*

Similarly to the classification case, the number of most general patterns that are treated unfairly may be exponential. This is true even if we consider only the lower bounds (e.g., if $U_k = |D|$).

Theorem 3.9. *Given a dataset D and a ranking algorithm R, no polynomial time algorithm can guarantee the enumeration of the set of all unfairly treated patterns by R.*

The proof is by construction, details are in Appendix B.

Upper bounds. The notion of the most general patterns is motivated by the utility of the information they provide. For example, for the case where the bounds are fixed (not proportionate), if the number of females in the top-k is less than the lower bound, then clearly the number of black females is below the bound. Unlike the most general patterns for the lower bound, in the case of upper bound, the most specific patterns are more informative. For instance, if the number of black females is above the upper bound, then so is the number of blacks and the number of females. A plausible problem definition may account for the *most specific substantial patterns.* Analogously to the definition of the most general patterns, a pattern p is a *most specific substantial pattern* if the size of p is above a given threshold τ_s and for every pattern p' such that $p \subsetneq p'$, the size of p' is below the threshold τ_s. The goal is then to find the most general patterns that do not satisfy the lower bound and the most specific substantial patterns that exceed the upper bound. For ease of presentation, in the rest of the paper we will focus on the solution for Problems 3.7 and 3.8 considering only the lower bounds. We note that our solutions can be adjusted to support such problem definition (and other definitions such as most general for upper bound, and the most specific for lower bound).

4 UNFAIRNESS IN CLASSIFICATION

In this section we present our approach for the problem of finding the set of all most general unfairly treated patterns in classification. A naive algorithm for the problem would perform the following: Given a dataset D, a classifier M, a fairness definition f, a threshold over the degree of unfairness $\Delta\tau_f$, and a size threshold τ_s, traverse the set of all possible patterns by the number of attributes they contain (i.e., $|Attr(p)|$), starting from the most general patterns ($|Attr(p)| = 1$). The algorithm maintains a result set R and for each pattern p computes the size $s_D(p)$ and fairness measure $f_M(p)$. If $s_D(p) \geqslant \tau_s$ and $|f_M(D) - f_M(p)| \geqslant \Delta\tau_f$, it is added to the result set. The algorithm terminates when the size of all the patterns in a given iteration ($|Attr(p)| = i$) is lower than the size threshold. We next present a more efficient solution. We start by considering the case where the fairness definition f is overall accuracy and then show how to generalize it for other definitions.

Low accuracy detection. When the fairness measure f is set to overall accuracy, we aim at detecting significant groups that have low accuracy compared to the overall classifier accuracy. More formally, given a database D, a classifier M, and the thresholds τ_s and $\Delta\tau_f$, the goal is to find the most general patterns p with $f_M(p) < f_M(D) - \Delta\tau_f$ and $s_D(p) \geqslant \tau_s$. The high level idea of the algorithm is as follows. First compute the set of misclassified tuples D^M_{mis} (tuples in D classified incorrectly by M). Note that given a pattern p, the accuracy of p can be computed as $f_M(p) = 1 - \frac{s_{D^M_{mis}}(p)}{s_D(p)}$. We traverse the set of possible patterns (starting with the most general ones) and compute the accuracy for each pattern.

To do so, we use the notion of *pattern graph* presented in [8]. Briefly, the nodes in the graph are the set of all possible patterns, and there is an edge between a pair of patterns p and p' if $p \subset p'$ and p' can be obtained from p by adding a single attribute value pair. In this case, we say that p (p') is a parent (child) of p' (p). As shown in [8], the pattern graph can be traversed in a top-down fashion, while visiting each pattern at most once. Namely, traversing a spanning tree of the pattern graph, we denote as the *search tree.*

Pruning the search space. The search tree can be pruned using the size threshold. Clearly, if $s_D(p) < \tau_s$ then for every p', a descendent of p in the search tree, it holds that $s_D(p') < \tau_s$. Based on the following, the search space can be further pruned using the accuracy threshold as well.

Proposition 4.1. *If* $s_{D^M_{mis}}(p) < (1 - \tau_f) \cdot \tau_s$ *then* p *can not be a most general unfairly treated pattern with size* $\geqslant \tau_s$

Algorithm 1: Low accuracy patterns detection

```
input  : A dataset $D$, a model $M$, a size threshold $\tau_s$ and an
         unfairness degree $\Delta\tau_f$
output : $\mathcal{P} = \{p_1, \dots, p_n\}$ s.t. $\forall p_i \in \mathcal{P}$ $p_i$ is a most general
         pattern with $s_D(p_i) \geq \tau_s$ and $f_M(D) - f_M(p_i) \geqslant \Delta\tau_f$
1  $mis_M \leftarrow$ misCalssified($D, M$)
2  $\tau_f \leftarrow f_M(D) - \Delta\tau_f$
3  $\mathcal{S} \leftarrow$ {generateChildren({})}
4  $\mathcal{P} \leftarrow \emptyset$
5  while $\mathcal{S}$ is not empty do
6      $p = \mathcal{S}.pop()$
7      $mis\_count \leftarrow$ patternSize ($p, D^M_{mis}$)
8      if $mis\_count > (1 - \tau_f) \cdot \tau_s$ then
9          $p\_count \leftarrow$ patternSize ($p, D$)
10         if $p\_count > \tau_s$ then
11             $acc \leftarrow (p\_count - mis\_count)/p\_count$
12             if $acc \geqslant \tau_f$ then
13                 $\mathcal{S}.push$(generateChildren(p))
14             else
15                 update ($\mathcal{P}, p$)
16 return $\mathcal{P}$
```

Proof. Note that the accuracy of a pattern p is lower than τ_f if $f_M(p) = 1 - \frac{s_{D^M_{mis}}(p)}{s_D(p)} \leqslant \tau_f$, i.e., $s_D(p)(1 - \tau_f) \leqslant s_{D^M_{mis}}(p)$. Since we are interested only in pattern p with $s_D(p) \geqslant \tau_s$ we can get $\tau_s \cdot (1 - \tau_f) \leqslant s_{D^M_{mis}}(p)$. Thus, if $s_{D^M_{mis}}(p) < (1 - \tau_f) \cdot \tau_s$ then p can not be a most general unfairly treated pattern with size $\geqslant \tau_s$ □

Based on the above proposition, patterns (and their descendants) with $s_{D^M_{mis}}(p)$ less than $\tau_s \cdot (1 - \tau_f)$ can be pruned.

Algorithm 1 detects patterns with low accuracy. It traverses the search tree of the pattern graph (top-down), using a queue $\mathcal{S}$ and maintains the set of identified low accuracy patterns $\mathcal{P}$. First using M and D the algorithm computes the set of misclassified tuples D^M_{mis} (line 1) and $\tau_f = f_M(D) - \Delta\tau_f$ (line 2). Then it initializes $\mathcal{S}$ to contain the children of the most general (empty) pattern (line 3), and sets the result set $\mathcal{P}$ to $\emptyset$ (line 4). While the queue $\mathcal{S}$ is not empty (lines 5 – 15), the algorithm extracts the first pattern in the queue p (line 6), and computes its size in D^M_{mis} (line 7). Next, based on Proposition 4.1, the algorithm uses the size of p in D^M_{mis} to prune the search, and considers only patterns that can lead to a valid output (line 8). For each such pattern, the algorithm computes its size in D (line 9). For patterns with size greater than τ_s (line 10), the accuracy of p is calculated using the computed sizes of p in D and D^M_{mis} (line 11). If the accuracy exceeds the threshold τ_f (line 12), the children of p are added to the queue (line 13). Otherwise, $\mathcal{P}$ is updated using the procedure `update` (line 15). The procedure `update` checks whether any ancestor of p in the pattern graph is already in $\mathcal{P}$ (this could happen because the algorithm traverses the search tree and not the patterns graph). If not, p is added to the result set. Finally, $\mathcal{P}$ is returned (line 16).

Example 4.2. Consider again D and M from the running example. Given the thresholds $\tau_s = 5$ and $\Delta\tau_f = 0.2$, the algorithm first computes τ_f and the set of miscalssified tuples D^M_{mis}. In this case, the overall accuracy of the classifier is 0.875, thus $\tau_f = 0.675$

and D^M_{mis} = {3, 13}. The most general patterns are p_1 ={Gender = M}, p_2 ={Gender = F}, p_3 ={School = MS}, p_4 ={School = GP}, p_5 = {Address = U}, p_6 = {Address = R}, p_7 = {Failures = 0}, p_8 = {Failures = 1} and p_9 = {Failures = 2}. The algorithm first considers p_1. Since $s_{D^M_{mis}}(p_1) = 1 < (1 - \tau_f) * \tau_s = 1.625$ this pattern and all of its descendants can be pruned. Similarly p_2 and p_3 and their descendants are pruned. Next, the algorithm considers p_4. There are two students in the GP school with incorrect predicted grades, thus $s_{D^M_{mis}}(p_4) = 2 > 1.625$. Since the total number of students in GP is 8, the accuracy of p_4 is $acc_M(p_4) = 0.75 > \tau_f = 0.675$, thus its children are generated. Similarly the children of p_5 are generated. Patterns p_6 through p_9 (and their descendants) are pruned as well. The patterns p_6 and p_9 are pruned due to their low number of miscalssified tuples. For the patterns p_7 and p_8, $s_D(p_7) = s_D(p_8) = 4 < \tau_s = 5$, thus they (and their descendants) are pruned. Finally, the pattern p_{10} ={School = GP, Address = U} (generated as a child of p_4) is considered. Since $s_D(p_{10}) = 6$ and $s_{D^M_{mis}}(p_{10}) = 2$, the accuracy of p_{10} is $0.67 < 0.675$ and p_{10} is added to the result set. In this example, this is the only pattern with low accuracy.

Generalized algorithm. Our solution can be generalized to capture the aforementioned fairness definition (and any other definition that relies on the values TP, FP, FN and TN). The overall idea of the generalized algorithm is similar to Algorithm 1 with the following modifications. First, instead of computing D^M_{mis} (line 1), the algorithm uses D and M to compute four datasets: D^M_{TP}, D^M_{FP}, D^M_{FN} and D^M_{TN} that contain the tuples from D that are TP, FP, FN and TN respectively. The algorithm traverses the search tree of the pattern graph in a top-down fashion, and for each pattern it computes the size, and the fairness measure using D^M_{TP}, D^M_{FP}, D^M_{FN} and D^M_{TN}. Note that the pruning based on $s_{D^M_{mis}}$ for low accuracy pattern detection is not applicable in general, however other methods can be apply such as pruning based on the numerator value.

Example 4.3. Assume the user wish to find groups that are treated unfairly according to the predictive equality measure (i.e., groups with high positive error rate $FPR = \frac{FP}{FP+TN}$). Given the thresholds $\tau_s = 4$ and $\Delta\tau_f = 0.25$, the generalized algorithm first computes the datasets $D^M_{TP} = \{1, 2, 5, 9, 11, 12, 14\}$, $D^M_{FP} = \{3\}$, $D^M_{FN} = \{13\}$ and $D^M_{TN} = \{4, 6, 7, 8, 10, 15, 16\}$. In this case, the overall FPR is $\frac{1}{8} = 0.125$, thus the algorithm should return groups with FPR of 0.375 or above. The algorithm starts from the most general patterns listed in Example 4.2. For patterns p_2 ={Gender = F}, p_3 ={School = MS}, p_6 = {Address = R}, p_7 = {Failures = 0} and p_9 = {Failures = 2}, the size in D^M_{FP} is 0 (i.e., the numerator of the FPR), thus their FPR is 0 and so is the FPR of their descendants, and the search can be pruned. For p_1 ={Gender = M}, , p_4 ={School = GP}, p_5 = {Address = U}, and p_8 = {Failures = 1} the FPR is below 0.375 (0.25, 0.16, 0.2 and 0.2 resp.). The algorithm then generates their children, out of which the pattern {Gender = M, Failures = 1} is the only one with a high FNR of $0.5 > 0.375$, thus it is added to the result set. In the next iteration the search terminates as all the patterns generated are below the size threshold.

5 UNFAIRNESS IN RANKING

We next turn to the problem of fairness in ranking. We first present our solution to the problem where unfairness is defined by specified bounds over the representation of groups in the data, and then, in Section 5.2, we show how to modify the solution in order to account for proportional representation of groups in the top-k ranked items.

5.1 Unfair Representation Using Bounds

Recall that the fairness definition of [13] restricts to the count of different patterns in the top-k tuples using upper and lower bounds. The problem of detecting unfairly treated groups in ranking is then to find patterns that their count does not lie within the given bounds in the top-k ranked tuples, for a given range of possible k's. Given a dataset D and a ranking algorithm R, a size threshold τ_s, a range $[k_{min}, k_{max}]$ and lower bounds L_k for each $k_{min} \leqslant k \leqslant k_{max}$[3], a simple solution would be to apply a top-down search, similar to the search in Algorithm 1, for each $k_{min} \leq k \leq k_{max}$. Then, in each iteration report the patterns that are treated unfairly by R. We next propose a more efficient algorithm for the problem.

The key observation is that when the lower bound remains the same for k and $k + 1$, the search spaces for patterns that are treated unfairly by R for k and $k + 1$ are typically very similar. This is because the set of top-k and top-$(k + 1)$ tuples differ by a single tuple. Namely, increasing k implies only local modifications to the search space. Let T_k be the search tree generated to find the set of unfairly treated patterns in the top-k tuples, and $R(D)[k + 1]$ the $(k + 1)$ element in the result of ranking D using R. We can bound the number of nodes in T_k whose size is affected by increasing k.

Proposition 5.1. *$R(D)[k + 1]$ can satisfy at most $\frac{|(T_k)|}{2}$ patterns (nodes) in T_k, where $|T_k|$ denotes the number of nodes in T_k.*

Proof. (Sketch) The basic idea of the proof, is that whenever a pattern p is generated during the search, at least one pattern p' with the same set of attributes ($Attr(p) = Attr(p')$) that differ from p in the value assignment of a single attribute is generated as well. This is true under the assumption that every attribute has at least 2 values. For instance, the patterns {Gender = M, School = MS} and {Gender = M, School = GP} are both children of the pattern {Gender = M} and are generated when the procedure `generateChilder`({Gender = M}) is invoked. Let A_i be the attribute such that $A_i \in Attr(p)$ and $\{A_i = a_i\} \subseteq p$ while $\{A_i = a'_i\} \subseteq p'$. $R(D)[k+1]$ can satisfy at most one of the patterns, as the value of A_i in $R(D)[k+1]$ is either a_i or a'_i (or possibly other value, if $|Dom(A_i)| > 2$). Since this is true for every node generated during the search, $R(D)[k + 1]$ can satisfy at most $\frac{|(T_k)|}{2}$ patterns (nodes) in T_k. □

In that case, by starting the search for $k + 1$ from the endpoint of the search for k, we significantly reduce the search space (and as a result, the runtime, see Section 6.2) .

In more details, Algorithm 2 starts by initializing the result set map Res (line 1). It then preforms a top-down search for the case where $k = k_{min}$. This is done using the procedure `TopDownSearch`. Similarly to the search for unfairly treated patterns in classification,

[3]We assume $L_k \leqslant L_{k+1} \forall k\ k_{min} \leqslant k < k_{max}$. This is a reasonable assumption since as k increase, so is the number of tuples in the top-k, thus it is only logical the bounds increase as well.

Algorithm 2: Detecting Unfair Representation Using Bounds in Ranking

```
input : A dataset $D$, a ranker $R$, a size threshold $\tau_s$, a range
        $[k_{min}, k_{max}]$ and lower bounds $L_k$ for each
        $k_{min} \leqslant k \leqslant k_{max}$
output: $Res$ s.t. for each $k_{min} \leqslant k \leqslant k_{max}$
        $Res[k] = \{p_1, \ldots, p_n\}$ where $\forall p_i \in Res[k]\ s_D(p_i) \geq \tau_s$
        and $p_i$ is a most general pattern with $s_{R^k(D)}(p) < L_k$
1  $Res \leftarrow \emptyset$
2  $Res[k_{min}], DRes \leftarrow$ TopDownSearch($D, R^{k_{min}}(D), \tau_s, L_{k_{min}}$)
3  for $k = k_{min} + 1$ to $k_{max}$ do
4      if $L_{k-1} < L_k$ then
5          $Res[k], DRes \leftarrow$ TopDownSearch($D, R^k(D), \tau_s, L_k$)
6      else
7          $Res[k] \leftarrow Res[k-1]$
8          for $b \in \{p \in Res[k-1] \mid R(D)[k]$ satisfies $p\} \cup DRes$
             do
9              searchFromNode ($b, Res[k], DRes$)
10 return $Res$
   Procedure TopDownSearch($D, R^k(D), \tau_s, L_k$)
1      $\mathcal{S} \leftarrow$ {generateChildren({})}
2      $\mathcal{P} \leftarrow \emptyset$
3      $DRes \leftarrow \emptyset$
4      while $\mathcal{S}$ is not empty do
5          $p = \mathcal{S}.pop()$
6          if patternSize ($p, D$) $> \tau_s$ then
7              $top\text{-}k\_c \leftarrow$ patternSize ($p, R^k(D)$)
8              if $top\text{-}k\_c < L_k$ then
9                  if update ($\mathcal{P}, p$) is False then
10                     $DRes \leftarrow DRes \cup \{p\}$
11             else
12                 $\mathcal{S}.push$(generateChildren($p$))
13     return $\mathcal{P}, DRes$
```

it uses a queue $\mathcal{S}$ (line 1 of the procedure) and maintains the set of unfairly treated patterns $\mathcal{P}$ for the given k (line 2). Additionally, it maintains the set $DRes$ (line 3). $DRes$ contains patterns p reached during the search (lines 4–12), with size below the lower bound, that are not part of the result set since it already contains an ancestor of p. When k increases (and L_k kept intact), the algorithm will utilize this set to initiate a local search in the pattern graph. TopDownSearch returns both, the result set $\mathcal{P}$ and the set $DRes$ (line 13).

Next, the algorithm preforms the search for each k from $k = k_{min} + 1$ through $k = k_{max}$ (lines 3–9). For each k, if the bound increases, TopDownSearch is used to preform a new top-down search. Otherwise, The algorithm considers only patterns from $DRes$ and patterns from $res[k-1]$ that the newly inserted tuple $R(D)[k]$ satisfies (line 8). This is because only their sizes in the top-k are affected by the new tuple. For each such pattern, the algorithm applies the procedure searchFromNode (line 9) to resume the search in the relevant parts of the graph. This search updates $Res[k]$ and $DRes$. Finally, Res is returned (line 10).

Example 5.2. Consider again D and R from the running example. Assume we are given the size threshold $\tau_s = 4, k_{min} = 4, k_{max} = 5$, and the lower bounds $L_4 = L_5 = 2$. At the end of the top-down search for $k = 4$, the result set $Res[4]$ contains (among others) the patterns {Address = U} and {Failures = 1}, that appears only once in the top-4 tuples (namely, below the lower bound). $DRes$ contains, for instance, the patterns {Gender = F, Address = U}, {Gender = M, Address = U}, {Gender = F, Failures = 1} and {Address = R, Failures = 1}. These patterns were generated during the top-down search and have an ancestor in $Res[4]$ ({Address = U} and {Failures = 1}). Next, the algorithm turns to compute the unfairly treated patterns for $k = 5$. The new tuple in the top-5 is tuple 14. It matches the patterns {Address = U} and {Failures = 1} in $Res[4]$. Thus the algorithm preforms the search starting from those nodes. Their size in the top-5 exceeds the lower bound. In this search, these two patterns are extracted from the result set and the pattern {Address = U, Failures = 1} is added. From the set $DRes$, the patterns {Gender = F, Address = U}, {Gender = M, Address = U}, {Gender = F, Failures = 1} and {Address = R, Failures = 1} are added to the result set $Res[5]$, as their sizes in the top-5 tuples is still below the threshold L_5 and their respective ancestors are removed from the result set.

5.2 Unfair Proportional Representation

We next consider the problem of (un)fair proportional representation in the top-k elements generated by a ranker R as depicted in Problem 3.8. The inputs are a dataset D, a ranker R, a range $[k_{min}, k_{max}]$, a size threshold τ_s and $\alpha \in \mathbb{R}$. The objective is to report the patterns p with adequate size but insufficient representation in the top-k tuples $R^k(D)$, where the representation in $R^k(D)$ should be proportional to the representation in D. More formally, we want to report the set of patterns p such that $s_D(p) > \tau_s$ and $s_{R^k(D)}(p) < \alpha \cdot s_D(p) \cdot \frac{k}{D}$ for each $k_{min} \leqslant k \leqslant k_{max}$.

The simple solution presented at the beginning of Section 5.1 can be used to solve this problem as well: apply a top-down search for each $k \in [k_{min}, k_{max}]$ and report patterns that are treated unfairly in each iteration. We next present a more efficient algorithm for the problem. First, note that the optimized solution presented for the case where fair representation in the top-k is defined by bounds L_k is not applicable here. Recall that Algorithm 2 aims at reducing the search space by avoid searching areas in the pattern graph that were not changed between consecutive iterations. When the bound remains unchanged ($L_k = L_{k+1}$), patterns which the $(k+1)$ tuple in the ranking does not satisfy are not affected, and can be eliminated from the search. This is not the case for proportional representation, as the bound for each pattern depends on k as well.

The goal is to find patterns p such that $s_{R^k(D)}(p) < \alpha \cdot s_D(p) \cdot \frac{k}{|D|}$, and note that α and $s_D(p)$ do not change during the computations. Thus, the inequality holds depending on the value of k. Given R, D, α, a pattern p and a value k such that $s_{R^k(D)}(p) \geqslant \alpha \cdot s_D(p) \cdot \frac{k}{|D|}$, we denote by $\tilde{k}$ the minimal value for k such that the inequality does not hold when fixing the value of $s_{R^k(D)}$. Namely, the minimal value such that $s_{R^k(D)}(p) < \alpha \cdot s_D(p) \cdot \frac{\tilde{k}}{|D|}$.

Example 5.3. Let $\alpha = 0.9$. For D and R from our running example, p ={Gender = F} satisfies the inequality for $k = 4$ since $s_{R^k(D)}(p) = 2 > 1.8 = 0.9 \cdot 8 \cdot \frac{4}{16}$. $\tilde{k} = 5$ is this case since $0.9 \cdot 8 \cdot \frac{5}{16} = 2.25$.

Intuitively, if k is increased up to $\tilde{k}$ but the number of patterns satisfying p remains the same, then p is treated unfairly. If there is no ancestor of p in the result set, then p should be added to it. To this end, the optimized algorithm computes for each pattern in the search tree its corresponding $\tilde{k}$ value. It maintains a set $\mathcal{K}$ which indicates patterns that potentially, if they do not satisfy the $\tilde{k}$ element in the ranking, should be added to the result set. $\mathcal{K}$ contains patterns in a branch of the search tree whose $\tilde{k}$ values are monotonically decreasing. A pattern p in $\mathcal{K}$ with the corresponding $\tilde{k}$ value should be extracted from $\mathcal{K}$ and added to the result set when the computation reaches $k = \tilde{k}$ if $s_{R^k(D)}(p) = s_{R^{k-1}(D)}(p)$ (i.e., p does not satisfy $R(D)[k]$). Our optimized algorithm operates as follows. It applies a top-down search for k_{min}, similarly to the search applied by the procedure `TopDownSerach` in Algorithm 2, but instead of the set $DRes$ it maintains the set $\mathcal{K}$. Then, the algorithm iterates over the values of k from $k_{min}+1$ to k_{max}. In each iteration, it utilizes $\mathcal{K}$ to determine the changes to the result set $Res[k]$ for patterns that are not affected by $R(D)[k]$. For patterns that are affect by $R(D)[k]$, the algorithm applies a search from the root. This search ignores the areas in the tree that are not affected $R(D)[k]$ (at least half of the tree, based on Proposition 4.1).

Example 5.4. Consider again the dataset given in Figure 1 and the ranker whose results are presented in the Rank column. Assume we are given the size threshold $\tau_s = 5, k_{min} = 4, k_{max} = 5$, and $\alpha = 0.9$. At the end of the top-down search for $k = 4$ the result set $Res[4]$ consists of the patterns {School = GP}, {Address = U} and {Failures = 1}. For each one of them, $s_D(p) = 8$, and thus the bound on $s_{R^k(D)}(p)$ is $\alpha \cdot s_D(p) \cdot \frac{k}{|D|} = 0.9 \cdot 8 \cdot \frac{4}{16} = 1.8$, but $s_{R^k(D)}(p)$ is only 1. The set $\mathcal{K}$ consists of {Gender = M} and {Gender = F}, both with $\tilde{k}$ of 5, and {School = MS} and {Address = R} with $\tilde{k} = 7$. Note that the pattern {School = MS, Address = R} was generated in the first top-down search, but was not added to $\mathcal{K}$ since its $\tilde{k}$ value is 9, higher than it's parent in the search tree {School = MS}. When k is increased to 5, the algorithm reexamines only the patterns {Gender = M}, {School = MS}, {Address = U} and {Failures = 1} that are affected by the $R(D)[5]$. The patterns {Address = U} and {Failures = 1} remain in the result set for $k = 5$ even-though their size in the top-5 is larger, since the bound for $k = 5$ increases as well. Finally, the pattern {Gender = F} is added to $Res[5]$ based on the information from $\mathcal{K}$.

6 EXPERIMENTS

We experimentally examine the proposed solutions, demonstrating the usefulness of our approach in detection of unfairly treated groups and the effectiveness of the proposed optimizations, using multiple real-life datasets. We start with our setup and then present a quantitative experimental study whose goal is to assess the scalability of our algorithms. In particular, we examine our algorithms' performance for the different fairness definitions as a function of the number of attributes, pattern's size threshold, degree of unfairness (for classification) and range of k (for ranking). We follow this with two case studies, whose goal is twofold. First is to show our solution can be used to reproduce results from previous studies [6], and even enrich them. The second aim is to demonstrate the benefits of eliminating the need to have the protected groups defined a priori. We conclude with a comparison of our proposed solution to existing methods in the context of classification.

6.1 Experiment Setup

Datasets. We used five real datasets with different numbers of tuples and attributes as follows.

The COMPAS dataset[4] that was collected and published by ProPublica as part of their investigation into racial bias in criminal risk assessment software. It contains the demographics, recidivism scores produced by the COMPAS software, and criminal offense information for 6,889 individuals. We used up to 16 attributes eliminating attributes such as names, ids, dates, etc.

The Default of Credit Card Clients Dataset (Credit card dataset)[5], which contains 23 attributes with information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. The dataset consist of $30K$ tuples. We used up to 15 attributes.

The Adult dataset[6] with information about 48,842 individual's annual income based on the census data. It was collected and used to explore the possibility in predicting income level (over/under $50K$ a year) based on the individual's personal information. We used 12 out of 14 attributes, omitting the fnlwgt and the native-country attribute (a categorical attribute with 41 values).

Student Performance Data Set (Student dataset)[7] shows the performance of students in secondary education of two Portuguese schools as described in Example 1.2. We considered in the experiment the data fragment with information regrading the Math exam (which includes 395 tuples and 33 attributes).

Medical Expenditure Panel Survey (Medical dataset)[8] shows the cost and use of health care and health insurance coverage. This dataset consists of 41 attributes and 7,915 tuples and was used by AI Fairness 360[9] to predict whether a person has a "high" utilization (the total number of trips requiring some sort of medical care). We used up to 16 attributes.

Compared algorithms. We evaluate the performance, in terms of running time of our proposed algorithms, and compare the result to baseline solutions for each problem as follows.

Detecting Unfairness in Classification (DUC). The algorithm for detecting unfairness in classification as depicted in Algorithm 1, and its generalized version described in Section 4.

Unfair Representation using Bounds (URB). Algorithm 2 for detecting unfair representation in the result of a ranking algorithm using bounds as described in Section 5.1.

Unfair Proportional Representation (UPR). The algorithm for detecting unfair proportional representation in the result of a ranking algorithm as described in Section 5.2.

Naive. The naive approach for detecting unfairness in classification described at the beginning of Sections 4.

[4]https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis

[5]https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

[6]https://archive.ics.uci.edu/ml/datasets/adult

[7]https://archive.ics.uci.edu/ml/datasets/student+performance

[8]https://nbviewer.org/github/IBM/AIF360/blob/master/examples/tutorial_medical_expenditure.ipynb#2.-Data-used

IterTD. The simple solution for detection of unfairness in ranking, which iteratively applies a top-down search as depicted in the beginning of Section 5.1.

Parameters setting. The number of attributes was set to be the maximal number the baseline solution could handle. We examined different parameters setting and observed similar trends. For space constraints, unless stated otherwise, we report the result for the following set of default parameters. For classification we use $\tau_s = 50$ and $\Delta\tau_f = 0.2$ (where in most experiments, the result set is neither empty nor too large, i.e., between 1 to 100). For ranking, $k_{min} = 10$, $k_{max} = 49$, and the lower bounds are 10 for $k = 10 \sim 19$, 20 for $k = 20 \sim 29$, 30 for $k = 30 \sim 39$ and 40 for $k = 40 \sim 49$ for ranking with bounds, and $alpha = 0.1$ for proportional representation. Similar trends are observed for other values.

ML models. The COMPAS, Credit card, Adults and Medical datasets are used for classification, thus we use them to examine the performance of the algorithms for classification. We use a decision tree to predict is_recid in the COMPAS dataset, default payment next month in the Credit card dataset, income in the Adult dataset, and directly use the classification results from AI Fairness 360 in Medical dataset. For ranking, we used the Student and COMPAS datasets, as they have natural properties for ranking (and used in such manner in previews studies, see, e.g., [7]). The Student dataset has an attribute G3 showing the student's math grades, which we use to rank all the tuples. To use COMPAS dataset in ranking, we preformed similar ranking method as in [7]: We used normalized attribute values c_days_from_compas, juv_other_count, days_b_screening_arrest, start, end, age, and priors_count as scoring attributes. Values are normalized as $(val - min)/(max - min)$. Higher values correspond to higher scores, except age. Tuples are ranked descendingly according to their scores.

All experiments were performed on a MacOS machine with a 2.8 GHz Quad-Core Intel Core i7 CPU and 8GB memory. The algorithms were implemented using Python3.7.

6.2 Experimental Results

Number of attributes. The first set of experiments aims at studying the effect of the number of attributes on the running time. To this end, we varied the number of attributes in the datasets from 3 to $|\mathcal{A}|$ where $\mathcal{A}$ is the set of all attributes in the dataset. The number of attributes (along with their cardinality) determines the number of possible patterns, and as a result, the size of the search space. Thus, as the number of attributes increases, we expect to see a steep growth in the running time. The results are presented in Figures 2–4. Indeed, in all cases we observed a rapid increase in the running time, particularly for the naive approach, which become infeasible beyond 7 to 9 attributes for classification and 13 to 31 attributes for ranking (using a 10-minute timeout).

Size threshold. In the next set of experiments, we assessed the effect of the size threshold τ_s on the running time. To this end, we varied the size threshold from 10 to 100. We did not observe any changes in the running time of the naive algorithm. Its running times were 8, 47, 24 and 80 seconds for the Adults, COMPAS, Credit card and Medical datasets, respectively, when using the overall accuracy fairness measure. In the case of false positive error rate the running times were 8 seconds for the Adult dataset, 6 seconds for the COMPAS dataset, 23 seconds for the Credit card dataset and 18 seconds for Medical dataset. As the size threshold increases, the number of patterns in the data with size above the bound decreases, namely, the DUC is able to prune the search more and reduce the search space and the search time accordingly. The results are presented in Figures 5. To better show this trend, we omit the naive algorithm from the plot. For ranking, IterTD utilizes pruning methods similar to DUC, thus we expect to see a moderate decrease in the running time for all algorithms (IterTD, URB and UPR). The results are presented in Figure 6 and 7.

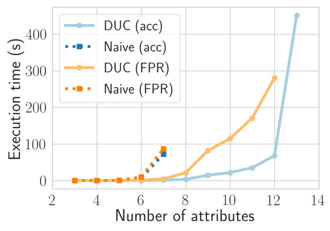
Degree of unfairness threshold (unfairness in classification). Figures 8 depicts the effect of the threshold over the degree of unfairness $\Delta\tau_f$ on the performance of algorithms from $\Delta\tau_f = 0.05$ to $\Delta\tau_f = 0.3$. We consider this a reasonable range because the fairness values of the datasets lies within the range of 0.5 ± 0.3 (e.g., the original COMPAS dataset has accuracy 0.64 and FPR 0.36). Since $\Delta\tau_f$ does not affect the running time of the naive algorithm, we show in the plots only the results for DUC.

Interestingly, in the case of the overall accuracy fairness measure, we observed an increase in the running time up to a certain point: 0.2 for the Adult dataset (green solid line), 0.2 in COMPAS (red solid line) and then a decrease. A similar trend was observed for the Credit card (yellow solid line) at 0.25 however the changes were in small scale (less than 0.1 seconds) and thus are not shown in the graph. This behavior could be explained by our pruning method. Recall that Algorithm 1 prunes the search space based on the values of τ_f (line 8). As $\Delta\tau_f$ increases, τ_f decreases, and the expression in the if condition $((1 - \tau_f) * \tau_s)$ increases, reducing the search space. However, at the same time, the if condition at line 12 is more likely to be satisfied, expanding the search space. We can observe that the condition in line 12 dominates the search space up to the turning points in the graph, when the pruning in line 8 comes into effect. This trend is not shown in the Medical dataset (purple solid line), as the turning point in this dataset occurs before $\tau_f = 0.05$.

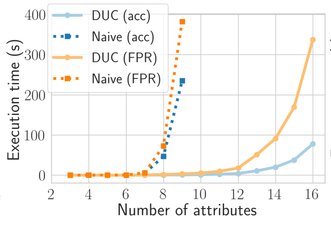
Range of k (unfairness in ranking). We examine the scalability with respect to the range of k considered by the algorithm. We varied the range from 40 (40) to 990 (340) by setting k_{min} to be 10, and increasing k_{max} from 50 (50) to 1000 (350) for COMPAS (Student) dataset and observed the effect on the running time. We set different maximum range of k due to different size of the datasets (6889 for COMPAS and 395 for Student). The results are presented in Figure 9 and 10. In both cases the optimized algorithms outperform IterTD, which illustrates the efficiency reduction in the search space.

Dataset	Number of patterns	# of patterns examined	% of patterns examined
Adult	78335	5372	6.86
COMPAS	1306799	16577	1.27
Credit Cards	414719	17943	4.22
Medical	1781999	34503	1.94

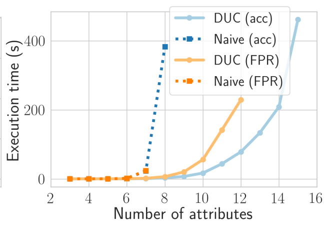
Table 1: Effect of optimization for unfairness in classification



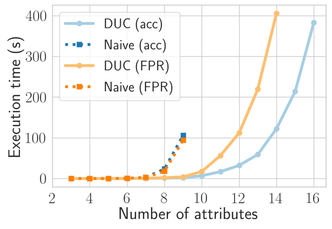
(a) Adult dataset



(b) COMPAS dataset

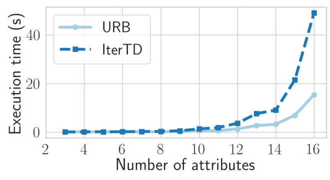


(c) Credit Card dataset

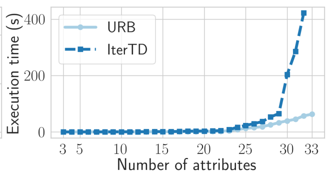


(d) Medical dataset

Figure 2: Running time as a function of number of attributes

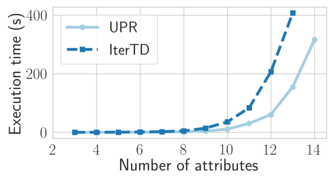


(a) COMPAS dataset

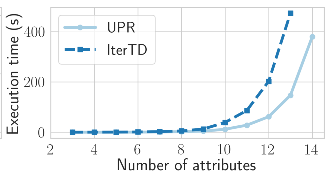


(b) Students dataset

Figure 3: Running time as a function of number of attributes - Ranking with bounds



(a) COMPAS dataset



(b) Student dataset

Figure 4: Running time as a function of number of attributes - Ranking with proportional representation

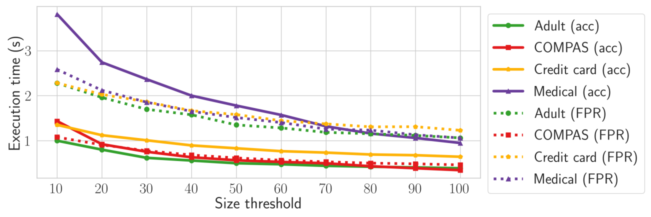
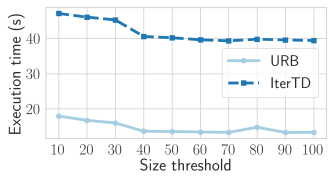
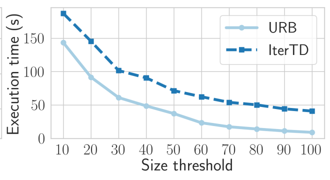


Figure 5: Running time of DUC as a function of the size threshold τ_s



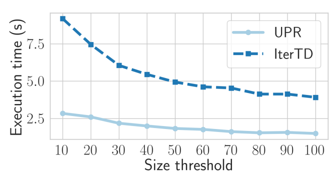
(a) COMPAS dataset



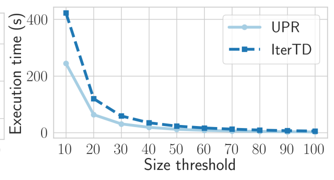
(b) Students dataset

Figure 6: Running time as a function of the size threshold τ_s - Ranking with bounds

Effect of optimization. Recall that DUC optimizes the number of patterns examined during the search. To quantify the usefulness of the optimization we compared the number of patterns examined during the search DUC and the naive approach. The results for the classification case are summarized in Table 1. We observed a gain of up to 93.14% in the number of pattern examined for the Adult dataset, 98.73% for the COMPAS dataset, 95.78% for the credit card and 98.06% for the Medical dataset.



(a) COMPAS dataset



(b) Students dataset

Figure 7: Running time as a function of the size threshold τ_s - Ranking with proportional representation

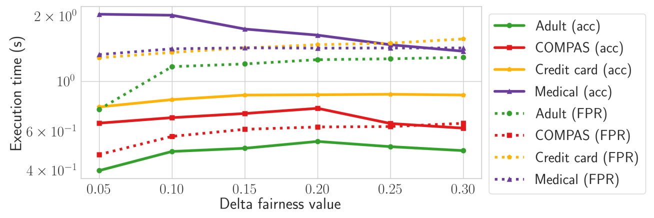
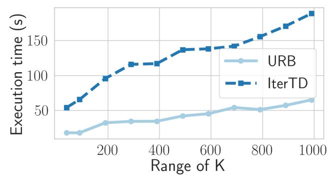
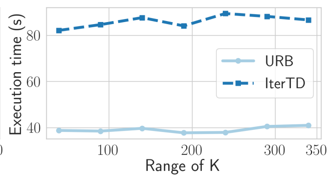


Figure 8: Running time as a function of the degree of unfairness $\Delta\tau_f$

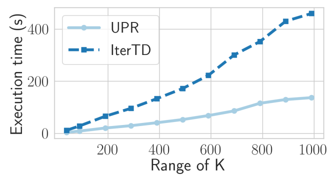


(a) COMPAS dataset

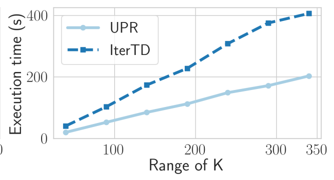


(b) Students dataset

Figure 9: Running time as a function of the range of k- Ranking with bounds



(a) COMPAS dataset



(b) Students dataset

Figure 10: Running time as a function of the range of k - Ranking with proportional representation

In the context of Ranking, URB and UPR optimize the search space compared to IterTD by utilizing the search result of the iteration for k in order to compute the result set for $k + 1$. Thus, as the range of k increase, we expect to see a grater improvement in the performance of the optimized algorithm compared to the naive approach. This trend was shown in Figure 9 and 10. To further demonstrate the usefulness of the approach, we compared the number of patterns examined during the search for each one of the algorithms. When the naive approach was feasible, the observed gain was up to 28.26% in the COMPAS dataset and 72.23% in the student dataset for ranking with bounds, and 42.17% and 20.63% respectively for ranking with proportional representation.

6.3 Case Studies

We next demonstrate the usefulness of our approach through two case studies. We first show the result of applying our algorithm on the COMPAS dataset, reproducing the discrimination reported in the study of [6]. As we next report, our algorithm was also able to detect other groups that were treated unfairly but were not reported as part of the result in [6]. Then, we demonstrate the benefit in eliminating the requirement to pre-defined the protected attributes. We show that the number of different attributes involved in the definition of an unfairly treated group may be large, and that different fairness definitions may vary to a considerable extent in the set of sensitive attributes.

Case study 1: reproducing the result of [6]. In the result of their study on the COMPAS system, ProPublica reported that black defendants were scored at greater risk of re-offending than the actual re-offending rates, while whites were scored at lower risk than actual rates [6]. Namely, blacks had a high false positive rate, and whites had a high false negative rate. We followed [6] to reproduce these results with the same setting, using our algorithm for detecting unfairly treated patterns. For the former, we applied the algorithm with the fairness measure of false positive error rate balance, and for the latter we used false negative error rate balance. We set the degree of unfairness to be $\Delta_f = 0.1$ since according their report, the false positive rate of blacks is 0.1 higher than the overall, and considered different size threshold. We report the results of Whelch's t-test for detected group, which can be used to determine their statistical significance.

For false positive error rate balance, with a degree of unfairness of $\Delta_f = 0.1$, starting from $\tau_s = 1$ and up to $\tau_s = 3696$, we were able to find the pattern {race = African-American} as a pattern with a high false positive error rate (44.85% compared the overall false positive rate of 32.35%, with t-value 9.00). We further found that for the same threshold over the degree of unfairness, when setting τ_s to 20, the pattern {age = Less than 25} is an unfairly treated pattern by this fairness definition, with a false positive rate of 54.14% (with t-value 10.53). Namely, we were able to reproduce the results reported by ProPublica, and found, in addition, that young people where treated unfairly by COMPAS. When using false negative error rate, we observed, similarly, that whites had high false negative error rate: while the overall false negative error rate was 37.40%, the false negative error rate of whites was 47.72% (with t-value 5.68). In addition we found the following other patterns with a high false negative rate: {race = Hispanic}, {race = Other} and {age = Greater than 45 } (with t-values 5.42, 7.30, 8.36, respectively).

Case study 2: sensitive attributes. Detecting unfairly treated groups is not a trivial task. As we show next, the detected groups can be defined by a large number of attributes, and these attributes may significantly vary between fairness definitions. We demonstrate this using the Medical dataset with the same pre-processing methods from AIF360 and note that we have similar observations when using other datasets. We consider the fairness measures false positive error rate balance (FPR) and false negative error rate balance (FNR), using the size threshold to be $\tau_s = 150$ and the degree of unfairness to be $\Delta_f = 0.1$.

The result set of FPR consists of three patterns using six sensitive attributes in total: {sex = Female, region = Northeast, Perceived mental health status = Good, Heart attack diag = No}, {sex = Female, region = Northeast, Angina diagnosis = No, Heart attack diag = No}, {sex = Female, region = Northeast, Perceived health status = Good}. The result set of FNR was largely diverse and significantly differ from the three patterns observed in the case of FPR. For instance, it consist of the patterns {Coronary hrt disease diag = Yes} and {sex = Female, region = Midwest, Student status = Inapplicable, Military full-time active duty = No, High blood pressure diag = No} (which also have no attribute overlap with each other).

6.4 Comparison with Existing Solutions for Classification

Recent works have studied the problem of automatic detecting "problematic" or biased subgroups in the data without the need to specify the protected attributes a priori [12, 15, 23, 26]. Closest to our work are DivExplorer [26] and SliceFinder [15], which both present alternative definitions and consider the problem only in the context of classification.

SliceFinder [15], defines the notion of problematic slices (patterns) using statistical techniques which measures the significance and magnitude of the performance for different slices. It returns the top-k patterns with statistically significant difference in classifier loss compared to its counterpart. In a similar vein to our definition of most general patterns, it is aimed at finding interpretable slices, i.e., slices with small number of attributes. SliceFinder presents two methods, using a decision tree and a lattice based search. The former finds only non-overlapping slices and the latter, employ a similar approach to our solution, searching the lattice graph. The differences in metric used to define the problematic groups, which is designed for a broader application domain (beyond fairness) leads to different results as we next show.

In [26], the authors introduced the notion of divergence to measure the difference in the behavior of a classification on data subgroups. They leverage existing algorithms for frequent itemsets mining and augment them to compute the divergence. In contrast to our solution, their goal is to report *all* subgroups with size above a given threshold and high divergence. This may result in an extremely large output. While comprehensive and informative, it may be less interpretable for users. In [25] they extend their framework to ranking, where they consider the average value in the ranking of an instance in each group as a measure of the group's outcome.

To demonstrate the differences between the solutions, we used the COMPAS dataset as used in the study of ProPuclica [6]. We consider false positive error rate balance, with a degree of unfairness $\Delta_f = 0.1$, and size threshold of $\tau_s = 72$ (support [25] of 0.01, i.e.,1% of the data). The result of our algorithm, as reported in case study 1

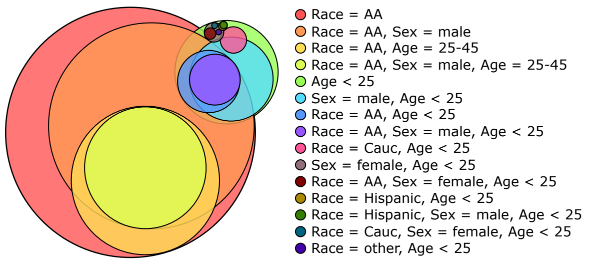


Figure 11: An illustration of the groups returned by DivExplorer [26] on the COMPAS dataset as used by ProPublica [6]. Each reported group is represented by a circle, where the circle's diameter is proportional to the group's size in the data. AA and Cauc are used as shorthand for African-American and Caucasian respectively. Our proposed method returns the union of the groups using only 2 patterns.

(Section 6.3) consists of two patterns: {race = African-American} and {age = Less than 25}. For SliceFinder, we used the default parameters setting, with degree set to be 3 (since there are 3 attributes in the dataset). The two patterns returned by our algorithm were not part of the results provided by SliceFinder. On the other hand, SliceFinder reported very small patterns such as {age = Greater than 45, race = Native American} and {age = Greater than 45, race = Native American, sex = male} (with size 3 and 1 respectively). This is because SliceFinder does not consider a size threshold (and reports the top-k results).

DivExplorer returns 15 groups (illustrated in Figure 11), which include the two groups (patterns) returned by our algorithm and additional subgroups of them. While informative, the result returned by DivExplorer can be redundant. For example, it contains the patterns {age = Less than 25, sex = male}, {age = Less than 25, sex = female} and the patterns that correspond to the union of them {age = Less than 25}. Another example of redundancy in the result set is that the patterns {race = African-American, sex = male}, {race = African-American, age = 24-45} and their intersection {race = African-American, sex = male, age = 24-45} are all part of the result set. Our solution returns a more concise result, the union of all the groups returned by DivExplorer using only two patterns, which provides the user a high-level understanding of the problematic parts of the data (in terms of fairness).

To further demonstrate the differences in the result set size, we compared the result of the two solutions on the COMPAS dataset as processed in [26] (which consists of 6 attributes). We used false positive error rate balance as the fairness measure and set the degree of unfairness to be $\Delta_f = 0.1$, and size threshold to $\tau_s = 61$ (support of 0.01). Our solution returned 10 patterns, while DivExplorer reported the same 10 patterns and additional 186 patterns, all of which are subgroups of the 10 patterns returned by our solution. Namely, a total number of 196 patterns, which can be overwhelming for a user to process manually.

7 RELATED WORK

The notion of fairness in ML was well studied in recent years, and various fairness measures have been proposed. In the context of classification, we focus on group fairness definitions that are based on the prediction and actual outcomes [10, 14, 20]. Several recently developed tools allow users to assess fairness of ML systems (see, e.g., [9, 11]). While those tools further assist users in mitigating the bias and provide explanations, they focus on investigating only user-specific protected groups. In [19], the authors studied the problem of evaluating the fairness of a system with respect to a given set of attributes, using the notion of casual discrimination. Our problem definitions, which eliminates the need to define the sensitive attributes, introduce a new and noticeable challenge: the need to explore a combinatorial search space in terms of data attributes.

Recent works have studied the problem of automatic detecting "problematic" or biased subgroups in the data without the need to specify the protected attributes a priori [12, 15, 23, 25, 26]. Closest to our work are SliceFinder [15] and DivExplorer [26] to which we have extensively compared our solution in Section 6.4. The work of [26] was extended to the context of ranking in [25]. Unlike our notation of unfairly treated groups, which relies on representation fairness measures for ranking, [25] builds on the average values in the ranking of instances in each group as a measure of the group's outcome. The interactive system MithraCoverage [23] investigates population bias in intersectional groups. The notion coverage is introduced to identify intersectional subgroups with inadequate representation in the dataset. Differently, in our work, we only report patterns with adequate representation. FairVIS [12] uses user-defined protected groups to audit the fairness of ML models and propose additional subgroups identified using a cluster-based technique. Unlike FairVIS, our methods do not relay on a priori defined set of protected groups, and goes beyond classification.

Several recent works have studied the notion of fairness in rankings while introducing different fairness definition [13, 22, 28, 30, 32, 33]. These definition typically focus on top-k positions, as those are usually the most important positions. Our definition of unfairness is derived from the [13]. We also consider a refined definition which aligns with the notion of proportional representation.

Technically, the algorithm for the case of classification presented in Section 4 built on the algorithm presented in [8], which in turns shares similar ideas to the Apriori algorithm [4], the Set-Enumeration Tree for enumerating sets in a best-first fashion [29], discovering functional dependencies [21, 24] and frequent item-sets and association rule mining [5]. Similarly, to [8], the key difference from our work lies in the structure of the graph traversed in the solution: the pattern graph (in our case) compared to the power-set lattice in the other works. On top of that, the algorithm in [8] alone is not sufficient for our needs, and some additional steps are required, such as extracting subsets of the dataset (e.g., D^M_{mis}) and traversing them (in addition to the given datasets) in parallel in order to compute the fairness measures. This difference provides us new pruning opportunities that were not applicable in [8].

8 CONCLUSION

In this paper, we have studied the problem of detecting groups in the data that are treated unfairly by a given machine learning model in the context of classification and ranking. Our work is compatible with a wide variety of (un)fairness metrics. There are many intriguing directions for future research, including the extension of the framework to support other fairness measures, in particular in the context of ranking, and further investigation on the automatic suggestion for the thresholds.

REFERENCES

[1] 2011. The Algorithm That Beats Your Bank Manager. https://www.forbes.com/sites/parmyolson/2011/03/15/the-algorithm-that-beats-your-bank-manager/?sh=4fbafadc1ae9.

[2] 2015. Can an Algorithm Hire Better Than a Human? https://www.nytimes.com/2015/06/26/upshot/can-an-algorithm-hire-better-than-a-human.html.

[3] 2020. When Algorithms Give Real Students Imaginary Grades. https://www.nytimes.com/2020/09/08/opinion/international-baccalaureate-algorithm-grades.html.

[4] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB*. Morgan Kaufmann.

[5] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*. Morgan Kaufmann, 487–499.

[6] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine bias. *ProPublica, May* 23, 2016 (2016).

[7] Abolfazl Asudeh, HV Jagadish, Julia Stoyanovich, and Gautam Das. 2019. Designing fair ranking schemes. In *Proceedings of the 2019 International Conference on Management of Data*. 1259–1276.

[8] Abolfazl Asudeh, Zhongjun Jin, and H. V. Jagadish. 2019. Assessing and Remedying Coverage for a Given Dataset. In *ICDE*.

[9] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John T. Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. *CoRR* abs/1810.01943 (2018). arXiv:1810.01943 http://arxiv.org/abs/1810.01943

[10] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. 2018. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research* (2018).

[11] Sarah Bird, Miro Dudík, Richard Edgar, Brandon Horn, Roman Lutz, Vanessa Milan, Mehrnoosh Sameki, Hanna Wallach, and Kathleen Walker. 2020. *Fairlearn: A toolkit for assessing and improving fairness in AI*. Technical Report MSR-TR-2020-32. Microsoft. https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai/

[12] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. 2019. Fairvis: Visual analytics for discovering intersectional bias in machine learning. In *VAST*.

[13] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. 2018. Ranking with Fairness Constraints. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic (LIPIcs)*, Vol. 107. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 28:1–28:15.

[14] Alexandra Chouldechova. 2017. Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data* 5, 2 (2017), 153–163.

[15] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. 2020. Automated Data Slicing for Model Validation: A Big Data - AI Integration Approach. *IEEE Trans. Knowl. Data Eng.* 32, 12 (2020), 2284–2296.

[16] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic Decision Making and the Cost of Fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 797–806.

[17] Paulo Cortez and Alice Maria Gonçalves Silva. 2008. Using data mining to predict secondary school student performance. (2008).

[18] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. 2012. Fairness through awareness. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*. ACM, 214–226.

[19] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*. ACM, 498–510.

[20] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of Opportunity in Supervised Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 3315–3323.

[21] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *Comput. J.* 42, 2 (1999), 100–111.

[22] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[23] Zhongjun Jin, Mengjing Xu, Chenkai Sun, Abolfazl Asudeh, and HV Jagadish. 2020. Mithracoverage: a system for investigating population bias for intersectional fairness. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2721–2724.

[24] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms. *Proc. VLDB Endow.* 8, 10 (2015), 1082–1093. https://doi.org/10.14778/2794367.2794377

[25] Eliana Pastor, Luca de Alfaro, and Elena Baralis. 2021. Identifying Biased Subgroups in Ranking and Classification. *arXiv preprint arXiv:2108.07450* (2021).

[26] Eliana Pastor, Luca de Alfaro, and Elena Baralis. 2021. Looking for Trouble: Analyzing Classifier Behavior via Pattern Divergence. In *Proceedings of the 2021 International Conference on Management of Data*. 1400–1412.

[27] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. 2021. Fairness in Rankings and Recommendations: An Overview. *CoRR* abs/2104.05994 (2021).

[28] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. 2021. Fairness in Rankings and Recommendations: An Overview. *CoRR* abs/2104.05994 (2021). arXiv:2104.05994 https://arxiv.org/abs/2104.05994

[29] Ron Rymon. 1992. Search through Systematic Set Enumeration. In *KR*. Morgan Kaufmann.

[30] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of Exposure in Rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 2219–2228.

[31] Sahil Verma and Julia Rubin. 2018. Fairness definitions explained. In *FairWare*.

[32] Ke Yang and Julia Stoyanovich. 2017. Measuring Fairness in Ranked Outputs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management, Chicago, IL, USA, June 27-29, 2017*. ACM, 22:1–22:6.

[33] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. 2017. Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1569–1578.

	A_1	A_2	A_3	$\cdots$	A_{n-1}	A_n	Outcome	Predication
t_1	1	0	0	$\cdots$	0	0	1	1
t_2	0	1	0	$\cdots$	0	0	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_n	0	0	0	$\cdots$	0	1	1	1
t_{n+1}	0	0	0	$\cdots$	0	0	1	0

Figure 12: Dataset D in the proof of Theorem 3.6 and 3.9

A PROOF FOR THEOREM 3.6

Proof. We prove the theorem by constructing an example with exponential number of unfairly treated patterns using the overall accuracy equality as the fairness measure. Consider a dataset D with $n+1$ (assuming $n \geqslant 2$) tuples $D = t_1, \ldots, t_n, t_{n+1}$ and n attributes, as depicted in Figure 12. In $t_1, \ldots, t_n$, for $A_1, \ldots, A_n$, only the values of elements on the diagonal are one and rest are zero. That is to say, $\forall i \in [1, n], t_i[A_i] = 1$, and $\forall j \neq i, t_i[A_j] = 0$. All the attributes of t_{n+1} are zero. The outcome attribute describes the actual outcome and the prediction attribute in the prediction of a classifier M. Namely, M correctly classifies all the tuples except for the tuple t_{n+1}. Thus the accuracy of M on D is $f_M(D) = \frac{n}{n+1}$. Let $\tau_s = 2$ be the size threshold and $\Delta_{\tau_f} = \frac{n-1}{(n+1)(n+3)}$ be the threshold over the degree of unfairness. Therefore, the goal is to find the set of all patterns p satisfying $s_D(p) \geqslant 2$ and $f_M(p) < f_M(D) - \Delta_{\tau_f} = \frac{n}{n+1} - \frac{n-1}{(n+1)(n+3)} = \frac{n+1}{n+3}$.

Consider a pattern p with $m \leqslant n$ attributes A_i such that $\{A_i{=}0\}$. Let I be the set of indices of those attribute. Among $t_1, \cdots, t_n$, the size of p is $n - m$, since $\forall i \in I, t_i[i] = 1, t_i$ does not satisfy p, and all the other tuples satisfy p. t_{n+1} also satisfies p, therefore $s_D(p) = n - m + 1$. Only t_{n+1} is mis-classified, thus the fairness measure of p is $f_M(p) = \frac{n-m}{n-m+1}$. Let $m = \frac{1}{2}n$, we have $f_M(p) = \frac{n/2}{n/2+1} = \frac{n}{n+2} < \tau_f = \frac{n+1}{n+3}$. To show p is the most general pattern to report, we examine its parent p' which has $m-1$ attributes with the value assignment 0. Similarly, size $s_D(p') = n - m + 2$, making the accuracy $f_M(p') = \frac{n-m+1}{n-m+2} = \frac{n/2+1}{n/2+2} = \frac{n+2}{n+4} > \tau_f = \frac{n+1}{n+3}$. As a result, every pattern with $\frac{1}{2}n$ attributes with values 0 are in the result set. The number of such patterns is $\binom{n}{n/2} > \sqrt{2}^n$, which is exponential. Therefore, any algorithm enumerating over these patterns is exponential. □

B PROOF FOR THEOREM 3.9

Proof. We prove the theorem by construction. Consider a dataset D with n attributes $\{A_1 \ldots, A_n\}$ (with a similar structure to the dataset depicted in Figure 12) and a ranker R where the top-k tuples in D are the tuples $t_1, \ldots, t_k$ in Figure 12 (i.e., only values of elements on the diagonal are one). Let $k_{min} = k_{max} = n, L_k = \frac{n}{2} + 1$ for Problem 3.7 (top-k ranking bounds unfairness), and $\alpha = \frac{n+3}{n+4}$ for Problem 3.8 (top-k ranking proportional representation unfairness), for some $n \geqslant 2$.

Consider a pattern p with $m \leqslant n$ attributes A_i with the value assignment $A_i = 0$. Similar to the proof of Theorem 3.6, among $t_1, \cdots, t_n$, the size of p is $n - m$. Let $m = \frac{n}{2}$, thus the size of p in top-k is $s_{R^k(D)}(p) = \frac{n}{2} < L_k = \frac{n}{2} + 1$. And for proportional representation, we have $s_{R^k(D)}(p) = \frac{n}{2} < \alpha \cdot s_D(p) \cdot \frac{k}{|D|} = \frac{n+3}{n+4}(\frac{n}{2} + 1)\frac{n}{n+1} = \frac{n}{2}\frac{(n+3)(n+2)}{(n+4)(n+1)}$. To show p is the most general pattern to report, we examine its parents p' which has $m - 1$ attribute with the value assignment 0. For ranking with bounds, we have $s_{R^k(D)}(p') = n - m + 1 = \frac{n}{2} + 1 = L_k$. For proportional representation, we have $s_{R^k(D)}(p') = \frac{n}{2} + 1 > \alpha \cdot s_D(p') \cdot \frac{k}{|D|} = \frac{n+3}{n+4}(\frac{n}{2} + 2)\frac{n}{n+1} = \frac{n+2}{2}\frac{(n+3)n}{(n+1)(n+2)}$. As a result, every pattern with $\frac{n}{2}$ attribute assigned to 0 should be in the result set. The number of such patterns is $\binom{n}{n/2} > \sqrt{2}^n$, which is exponential. Therefore, any algorithm enumerating over these patterns is exponential. □