< Application and extension of machine learning fair algorithms for ranking>

Christos Georgitsis

Thesis

Supervisor: Prof. Evaggelia Pitoura

Ioannina, <July> <2020>

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Acknowledgments

After having completed my dissertation, I would like to thank everyone that devised to it. I would like to thank deeply my supervisor, Prof. Pitoura Evangelia, for trusting me from the beginning, assigning me this particular topic, her scientific guidance, her suggestions and for being that supportive throughout the whole procedure until the end.

<July,2020>

<Christos Georgitsis>

Abstract

Algorithms for fair ranking, concerned a section of the scientific community, as scientists observed that the final ranking that emerged after data processing was usually biased in favor of one group. The purpose of this thesis was the study - analysis of two algorithms (DELTR, FA*IR) for fair classification. More specifically, in this thesis, after the form of the input data was studied, various databases were created in order to test the two algorithms. In addition, with the help of some metrics, we extracted useful information that helped us better understand algorithms and draw useful conclusions.

Subsequently we were expanding the function of the FA*IR algorithm to work for more than two groups. Through this expansion, we have made it possible to use any database, as the initial restriction that allowed us to have only two groups is no longer valid. This extension gave us the ability to export fair results regardless of the number of groups contained in the database. Then we created various databases in order to prove that the extension we created has the same results as the FA*IR algorithm, that is, the final ranking should be classified as fair. In addition, again with the help of some metrics, we drew conclusions about whether the expansion of the algorithm we created works properly. Finally, based on the metrics, we created graphs that show the difference between the initial ranking and the resulting ranking after running the FA*IR algorithm.

Keywords: <fair ranking>, <database>

Περίληψη (extended abstract in Greek)

Η εύρεση αλγορίθμων για δίκαιη κατάταξη των αποτελεσμάτων απασχόλησε έντονα ένα μέρος της επιστημονικής κοινότητας, καθώς είχε παρατηρηθεί ότι η τελική κατάταξη που προέκυπτε μετά την επεξεργασία των δεδομένων, συνήθως μεροληπτούσε υπέρ μίας ομάδας. Σκοπός της συγκεκριμένης διπλωματικής εργασίας ήταν η μελέτη - ανάλυση δύο αλγορίθμων (DELTR, FA*IR) δίκαιης ταξινόμησης. Πιο συγκεκριμένα στην παρούσα διπλωματική αφού μελετήθηκε η μορφή των δεδομένων εισαγωγής, δημιουργήθηκαν διάφορες βάσεις δεδομένων με σκοπό το τεστάρισμα των δύο αλγορίθμων. Επιπρόσθετα με την βοήθεια ορισμένων μετρικών αντλήσαμε χρήσιμες πληροφορίες που μας βοήθησαν στην καλύτερη κατανόηση των αλγορίθμων και την εξαγωγή χρήσιμων συμπερασμάτων.

Στην συνέχεια επεκτείναμε την λειτουργία του αλγορίθμου FA*IR ώστε να λειτουργεί για περισσότερες από δύο ομάδες. Μέσω αυτής της επέκτασης, καταστήσαμε εφικτό να μπορούμε να χρησιμοποιήσουμε οποιαδήποτε βάση, καθώς πλέον ο αρχικός περιορισμός που μας επέτρεπε να έχουμε μόνο δύο ομάδες είχε καμφθεί, δινοντάς μας την δυνατότητα να εξάγουμε δίκαια αποτελέσματα ανεξάρτητα από των αριθμό των ομάδων που περιέχονται στην βάση δεδομένων. Στην συνέχεια δημιουργήσαμε διάφορες βάσεις δεδομένων με σκοπό να αποδείξουμε πως η επέκταση που δημιουργήσαμε έχει τα ίδια αποτελέσματα με τον αλγόριθμο FA*IR, δηλαδή η τελική κατάταξη να χαρακτηρίζεται ως δίκαιη. Επιπρόσθετα, πάλι με την βοήθεια ορισμένων μετρικών εξάγαμε συμπεράσματα για το αν λειτουργεί ορθά η επέκταση του αλγορίθμου που δημιουργήσαμε. Τέλος βασιζόμενοι στα metrics δημιουργήσαμε γραφήματα που απεικονίζουνε την διαφορά της αρχικής κατάταξης και της κατάταξης που προκύπτει αφού τρέξουμε τον αλγόριθμο FA*IR.

Λέξεις Κλειδιά: <δίκαιη κατάταξη>, <Βάσεις δεδομένων>

Table of Contents

Chapter 1. Introduction

This chapter refers to a brief description of the subject of this diploma -thesis, as well as a description of the next sections. A database is an organized collection of data that is usually organized to model various aspects of it and facilitate their questioning. Today, the size of data that is stored in the databases is huge so there has been a lot of researches about databases. The rapid growth in the number and size of databases creates a need for tools and techniques for data analyzing and more specifically for algorithms which will help us to rank their results. With the volume of information increasing at a frenetic pace, ranked search results have become the main mechanism by which we try to find content we are interested in. Ranked search results provide us with useful information where we can rely on to make some decisions.

1.1 Subject of thesis

The increased variety of information makes it critical to retrieve documents that are not only relevant but also broad enough to cover as many different aspects of a certain topic as possible. Search engines today are used to rank many different types of items, including items that represent people. Job recruiting search engines, marketplaces for consulting and other services, dating apps, automate data-driven processes in education, recruitment, banking, and judiciary systems etc. have at its core the idea of ranking/ordering people from most relevant to less relevant, which often means from “best” to “worst”.

In recent years, the scientific community has been studying databases with systematic biases. A system discriminates unfairly if it denies an opportunity or if it assigns an undesirable outcome to an individual or a group of individuals on grounds that are unreasonable or inappropriate. Accordingly, a database is biased if it systematically and unfairly discriminates against certain individuals or groups of individuals in favor of others. On a ranking, the desired good for an individual is to appear in the result and to

be ranked amongst the top-k positions. The outcome is unfair if members of a protected group are systematically ranked lower than those of a privileged group. This can happen if a ranking decision is based fully or partially on the protected feature then the ranking algorithm discriminates unfairly.

In this section, we investigate the problem of discrimination in various systematic biases and we are experimenting with two fair algorithms (DELTR, FA*IR), intending to make final results-rankings fairer. Following long-standing empirical observations showing that users of information retrieval systems rarely look past the first few results, researchers measure these problems in terms of discrepancies in the average group exposure (Namely, in the final rankings some teams occupy much higher positions than others, but this difference could not emerge from the given data). This led to the need to create algorithms that would produce fair results. In algorithmic bias, an important concept is that of a protected group, which is a category of individuals protected by law, voluntary commitments, or other reasons. Different researchers have used different notions of algorithmic fairness. Fairness is also referred to as statistical parity. It is a requirement that the protected groups should be treated similarly to the advantaged group or the populations as a whole.

The subject of this thesis is to create different databases for testing the given algorithms, to see how they work and to expand FA*IR algorithm to be able to operate with more than two groups of individuals. The main problem which we faced was the difficulty of building fair databases that are suitable for these algorithms, mostly for DELTR.

As shown in earlier research, a machine learning model trained on datasets incorporating preexisting bias will embody this bias and therefore produce biased results, potentially increasing any disadvantage further, reinforcing existing bias. Based on this observation, we study two algorithms:

- The first one is called DELTR, a learning-to-rank framework that addresses potential issues of discrimination and unequal opportunity in rankings. This method neither makes assumptions about biases in the training data nor does it ignore relevance scores of items and thus can reduce discrimination and inequality of opportunity without having to introduce large distortions in ranking relevance. DELTR extends list-wise learning to rank method with an objective that reduces the extent to which non-protected elements receive less exposure than protected elements. Specifically, researchers define their notion of group exposure as the average probability of items from a legally protected social group to be ranked at the top position. With this, researchers design a

ranker that optimizes search results in terms of relevance, while at the same time reducing potential discrimination or inequality of opportunity. More specific DELTR is a supervised learning-to-rank algorithm that simultaneously seeks to minimize ranking errors with respect to training data and to reduce disadvantages experienced by the protected group in terms of exposure. Additionally, DELTR constitutes an in-processing method. This means that contain fairness objectives into their loss functions. Researchers show that their in-processing method performs better in terms of relevance and equality of exposure than pre-processing and post-processing across all tested scenarios.

- The second one is called FA*IR and studies the problem of producing a fair ranking given one legally-protected attribute, a ranking in which the representation of the minority group does not fall below a minimum proportion p at any point in the ranking, while the utility of the ranking is maintained as high as possible. Researchers provided an algorithmic solution to the Fair Top-k Ranking problem over a single binary type attribute. Concerning the group fairness criteria, they provided a greedy algorithm that selects a subset of top k candidates from a large pool while ensuring maximal utility. FA*IR approach yields small distortions concerning rankings that maximize utility without considering fairness criteria. It's also remarkable that FA*IR is the first algorithm grounded in statistical tests that can mitigate biases in the representation of an under-represented group along with a ranked list.

1.2 Organization of the volume

This thesis consists of 6 chapters, which are discussed in the following paragraphs.

Chapter 2 describes the related work for fair ranking in databases. Also, a clearer description of the purpose of the specific task is given, describing the subject in more detail.

The first part of chapter 3 gives some useful definitions related to fair ranking. More specific we would mention some algorithms – definitions that formed the basis for the creation of DELTR and FA*IR. In the second part, we will represent the mathematical formula of our two algorithms and algorithmic description of the methods relatively their operation. Moreover, a description is given of the new method which we have created based on FA*IR algorithm, in order to work with databases that have as feature m groups of individuals.

Chapter 4 explains in detail the methodologies - metrics we used in our experiments and quotes in detail the results of our experiments. Moreover, we give some details about the databases we use for testing and how databases have emerged.

The first part of Chapter 5 contains the conclusions that emerged after the experiments which we ran and the second part includes some of the extensions that could be done as future work.

In Chapter 6 reference is made to the literature that has been used to implement this thesis.

Finally, Chapter 7 contains some useful functions that were used to extend the FA*IR algorithm so, it can handle basis that contains more than two groups of individuals.

Chapter 2. Description

2.1 Related research - work

This section describes related work, available technologies as well as theoretical models that can be used as a basis for completing this thesis.

Several scientists are concerned with learning to rank, which is to construct a model or a function for ranking objects. Learning to rank is useful for document retrieval, collaborative filtering, and many other applications. There is one major approach to learning to rank, referred to as the pairwise approach. In the pairwise approach, the learning task is formalized as classification of object pairs into two categories (correctly ranked and incorrectly ranked). Herbrich et al., [GHBO99] proposed employing the approach and using the SVM techniques to build the classification model. The method is referred to as Ranking SVM. In [BSR+05], the authors also adopted the pairwise approach and developed a method called RankNet. They employed Cross Entropy as loss function and Gradient Descent as algorithm to train a Neural Network model. To be more specific, authors applied RankNet to large scale web search [CXL+06] adapted Ranking SVM to document retrieval by modifying the loss function. Learning to rank, particularly the pairwise approach, has been successively applied to information retrieval. For instance, Joachims [JOAC02] applied ranking SVM to document retrieval. The author developed a method of deriving document pairs for training, from users' clicks-through data.

Although the pairwise approach offers advantages such as:

- existing methodologies on classification can be directly applied and
- the training instances of document pairs can be easily obtained in certain scenarios,

the method has several disadvantages that outweigh its advantages. The main ones are listed below:

- pairwise approach ignores that ranking is a prediction task on list of objects.
- Furthermore, the objective of learning is formalized as minimizing errors in classification of document pairs, rather than minimizing errors in ranking of documents.
- Moreover, the training process is computationally costly, as the number of document pairs is very large.
- Finally, the number of generated document pairs varies largely from query to query, which will result in training a model biased toward queries with more document pairs.

For all these reasons, scientists postulate that learning to rank should adopt the listwise approach in which lists of objects are used as 'instances' in learning. Researchers [CQL+07] employ a new learning method called ListNet, for optimizing the listwise loss function based on top one probability, with Neural Network as model and Gradient Descent as optimization algorithm.

More specifically, researchers [CQL+07] propose a probabilistic method to calculate the listwise loss function. Authors transform both the scores of the documents assigned by a ranking function and the explicit or implicit judgments of the documents given by humans into probability distributions. Subsequently, authors utilize any metric between probability distributions as the loss function. In addition, researchers consider the uses of two models for the transformation; one is referred to as permutation probability and the other top one probability. Finally, author propose a learning to rank method using the list-wise loss function, with Neural Network as model and Gradient Descent as algorithm. We refer to it as ListNet.

Notice that ListNet is similar to RankNet. The only major difference lies in that:

- ListNet uses document lists as instances while RankNet uses document pairs as instances
- ListNet utilizes a listwise loss function while RankNet utilizes a pairwise loss function. Interestingly, when there are only two documents for each query, then the listwise loss function in ListNet becomes equivalent to the pairwise loss function in RankNet.

Experimental results on information retrieval show that the proposed listwise approach performs better than the pairwise approach.

An additional study from Radlinski and Dumais [AGHL09] observe that the top k results of a query might not be diverse enough to contain representative documents corresponding to various interpretations of the query, thus limiting the usefulness of personalized client-side reranking of results. Researchers study the question of generating related queries in order to yield a more diverse set of documents. Researchers work is related with the algorithms we are working on as both algorithms (DELTR, FA*IR) try to create a fairer ranking by trying to go up the protected documents in the ranking. In particular, the algorithms we are working with attempt to deliver justice, parity and diversity.

Another related work is a tool named COMPAS (Correctional Offender Management Profiling for Alternative Sanctions). Criminologists have long tried to predict which criminals are more dangerous before deciding whether they should be released. Race, nationality and skin color were often used in making such predictions until about the 1970s, when it became politically unacceptable, according to a survey of risk assessment tools by Columbia University law professor Bernard Harcourt. COMPAS is a case management and decision support tool developed and owned by Northpointe (now Equivant) used by U.S. courts to assess the likelihood of a defendant becoming a recidivist. The COMPAS software uses an algorithm to assess potential recidivism risk. Northpointe created risk scales for general and violent recidivism, and for pretrial misconduct. According to the COMPAS Practitioner's Guide, the scales were designed using behavioral and psychological constructs of very high relevance to recidivism and criminal careers. Northpointe’s core product is a set of scores derived from 137 questions that are either answered by defendants or pulled from criminal records. In 2012, the Wisconsin Department of Corrections launched the use of the software throughout the state. It is used at each step in the prison system, from sentencing to parole.

Subsampling from a large data set is another basic algorithmic task that is central in machine learning. Subsamples are used both as an end-goal in data summarization and to train algorithms (where biases in the samples are often a source of bias in the resulting model). A crucial requirement for either task is that the sample be diverse in the feature space, this is important both to provide a comprehensive viewpoint if the sample is the end-goal, and to make algorithms trained on such samples robust. However, diversity may not guarantee fairness on sensitive attributes and may propagate biases, leading to broken models and algorithmic prejudice.

Another related research presents an algorithmic framework that allows a user to integrate both notions of diversity (Combinatorial diversity, Geometric diversity), and experimentally demonstrate a marked improvement in fairness without compromising geometric diversity by much – resulting in the best solution. Conceptually, researchers [CDKV16] propose a novel generalization of k DPPs which they call P-DPP.

Algorithmically, a polynomial time algorithm for sampling k-DPPs generalizes, albeit nontrivially, to P-DPPs with a constant number of disjoint partitions, making researchers approach feasible. After researches, authors notice that P-DPP outperforms or matches other approaches with respect to both combinatorial measure of diversity and a geometric measure of diversity in three different scenarios:

- when perfect fairness is ensured,
- when some sensitive attributes remain hidden and
- when the underlying dataset is biased.

This experiment demonstrates that P-DPP can match or outperform the other approaches with respect to both fairness and diversity. Hence, this experiment demonstrates that P-DPP can match or outperform the other approaches with respect to both fairness and diversity, even when some of the underlying attributes are unknown. Despite all the advantages of P-DPP, the only difference between P-DPP and the algorithms we work on is that P-DPP takes only a subsample of the database contrary to FA*IR and DELTR which they work on the whole database.

2.2 Our contribution

This thesis is aimed at analyzing two algorithms from the fair ranking literature, namely FA*IR [ZBC+17] and DELTR [ZeDC19]. Particularly, this thesis focusses on testing these algorithms with different databases. Moreover, some interesting metrics are presented, which depict how much the initial ranking differs from the final one (which is fairer than the initial one). Finally, in this thesis is represented one expansion for FA*IR algorithm, which will give us the opportunity to test it with four groups of individuals (e.t Africans, Americans, Europeans, Asians). This expansion allows us to test a variety of databases, more complicated than authors have proposed. This gives us the advantage of extending our testing and monitoring the behavior of FA * IR in more complicated databases since in the real world usually, databases have more than two groups of individuals.

Chapter 3. Design & Implementation

3.1 Useful definitions

Before we get started it would be nice to have some definitions related to fair ranking. At the same time, we would mention some algorithms that formed the basis for the creation of FA * IR and DELTR.

At this point we should mention that there are three main approaches to build fair algorithms:

- a **pre-processing** of the data, to reduce the difference between the protected class and the non-protected one. In this way, an algorithm is supposed to avoid learning possible discriminations coming from the data.
- by operating **in-processing**, i.e., by plugging the fairness measures inside the algorithms, making them operate fairly by design.
- or to build **post-processing**, which usually re-ranks the unfair results produced by an algorithm, thus mitigating the unfairness coming from it.

3.1.1 DELTR: a learning to rank approach

3.1.1.1 Useful observations about how LISTNET works:

In Learning to Rank, there is a ranking function, that is responsible for assigning the score value. This function is learning in the training phase, where is provided with a collection of queries, documents and target scores. Learning to rank, when applied to document retrieval, is a task as follows. Assume that there is a collection of documents. In retrieval (i.e., ranking), given a query, the ranking function assigns a score to each document and ranks the documents in descending order of the scores. The ranking

order represents the relative relevance of documents concerning the query. Describe Learning to Rank in 3 simple steps:

- a number of queries are provided,
- where each query is associated with a perfect ranking list of documents,
- Then a ranking function is created using the training data, such that the model can precisely predict the ranking lists in the training data.

As we have mentioned before DELTR extends LISTNET. More specifically, for each Query Q (consider a set of queries Q and a set of documents D) the list of documents is associated with a list of judgments. For a clearer distinction between different judgment sets, we call $y^{(q)}$ the judgments of the training data and $\hat{y}^{(q)}$ the judgments predicted by the model. From each document d(q) we can derive a feature vector x(q). Each list of feature vectors x(q) and the corresponding list of judgments $y^{(q)}$ form an instance of the training set T. The standard learning-to-rank objective then is to learn a ranking function f that outputs a new judgment $\hat{y}^{(q)}$ for each feature vector x(q) which forms the second list of judgments $\hat{y}^{(q)}$. Function f should be such that the sum of the differences L between the training judgments $y^{(q)}$ and the predicted judgments $\hat{y}^{(q)}$. In list-wise learning to rank, training elements are processed as lists of elements (not as individual elements having scores, which corresponds to point-wise learning to rank, or as pairs of elements, which corresponds to pair-wise learning to rank). We call the learning problem described above as the listwise approach to learning to rank.

3.1.2 FA*IR: a fair top-k ranking algorithm

Two basic frameworks have been adopted in recent studies on algorithmic discrimination:

- individual fairness, a requirement that individuals should be treated consistently and
- group fairness, also known as statistical parity, a requirement that the protected groups should be treated similarly to the advantaged group or the populations as a whole

A definition of group fairness has been proposed by Yang and Stoyanovich. Essentially, authors compare the distributions of protected and non-protected candidates on dissimilar prefixes of the list (e.g., top-10, top-20, top-30) and finally take the average of these differences. Researchers use the synthetic ranking generation procedure of Yang and Stoyanovich to adjust FA*IR and advance the utility ranked group fairness. In other

words, unlike, researchers connect the creation of the ranking with the metric used for assessing fairness.

3.1.2.1 Top-K ranking problem:

Authors want to determine a subset of k candidates from a large pool of n >> k candidates, in a way that maximizes utility (selects the "best" candidates), subject to group fairness criteria. The utility objective is operationalized in two ways.

- First, by a criterion called **selection utility**, whereby, the final ranking will obtain elements with the highest score values (more qualified) or elements in which the difference in their scores are infinitesimal.
- Second, by a criterion, called ordering utility, whereby, the top-k result of ranking includes candidates classified in descending order based on their score, and in every pair of candidates either the better candidate proceeds the worst candidate or the difference in their qualifications-score is small.

Finally, researchers propose an efficient algorithm, named FA*IR, for producing a top-k ranking that maximizes utility while satisfying ranked group fairness, with the sole prerequisite that there are "enough" protected candidates to achieve the desired minimum proportion.

3.1.2.2 How FA*IR treats discrimination as a matter of group or individual:

Group utility:

Researchers use the notion of utility because it reflects their desire to select candidates that are potentially better qualified and to rank them as high as possible. In contrast with other algorithms, in FA*IR there are no assumptions about the contribution of having a given candidate at a particular position, but instead, the utility is calculated based on losses which are created due to non-monotonicity. The qualifications may have been even proven to be biased against a protected group, but FA*IR can bound the effect of that bias because the utility maximization is subject to the ranked group fairness constraint.

Individual utility:

The more specific notion of utility is centered on individuals, for instance by taking the minima instead of averaging. While other choices are possible, this has the advantage that we can trace loss of utility to specific individuals. These are the people who are ranked below a less qualified candidate, or excluded from the ranking, due to the ranked group fairness constraint. This is connected to the notion of individual fairness, which

requires people to be treated consistently. Under this interpretation, a consistent treatment should require that two people with the same qualifications be treated equally, and any deviation from this is in our framework a utility loss. This allows trade-offs to be made explicit.

3.2 Detailed description of DELTR and FA*IR

3.2.1 An in-processing method called DELTR

3.2.1.1 Mathematical formula of DELTR:

Generally, in traditional learning-to-rank systems, a ranking function f is learned by minimizing a loss function L, which measures the error between predictions $\hat{y}$ made by f and the training judgments y. Function f is trained by solving a minimization problem with respect to a loss function L_{DELTR}. In this point we should mention that L_{DELTR} combines the information from the training data L and a measure U that shows how unfair is the generated rankings in term of exposure.

Categorically, in DELTR we extend the loss function of LISTNET (DELTR is using LISTNET as a base algorithm) by adding a well-known LTR by a term U, which measures the "unfairness" of a predicted ranking.

$$L_{DELTR} = L(y, \hat{y}) + \gamma U(\hat{y})$$

We define U to be a measure of disparate exposure across different social groups in a probabilistic ranking $P_{\hat{y}}$. In this term, we can now introduce an unfairness criterion measured in terms of disparate exposure. This means we measure discrepancies in the probability to appear at the top position, received by items of the protected group G1 vs items of the non-protected group G0 (Items in the protected group have a certain protected attribute, such as belonging to an underprivileged group):

$$U(\hat{y}^{(q)}) = \max\left(0, \text{Exposure}(G_0|P_{\hat{y}^{(q)}}) - \text{Exposure}(G_1|P_{\hat{y}^{(q)}})\right)^2$$

By the use of square, we achieve to have a differentiable loss function that prefers rankings in which the exposure of the protected group is not less than the exposure of the non-protected group. This means that our definition will optimize only relevance in

cases where the protected group already receives as much exposure as the non-protected group. Finally, it's worth mentioning that U only depends on prediction $\hat{y}$ and consequently is not directly affected by biases in the training procedure, which is a great advantage compared to the naïve approach.

Term U has a coefficient called gamma (γ). Depending on the value of gamma we have 2 different cases:

- When we prefer larger γ value we expect the final result-ranking to be quite different from the original one, by reducing disparate exposure for the protected group and
- When we prefer smaller γ value we expect output of the ranking algorithm to be quite similar with the original one.

In conclusion, we should mention that parameter γ depends on desired trade-offs between ranking utility and disparate exposure that are application-dependent.

3.2.1.2 How algorithm works:

As we have referred before ranking function (which infers to document judgments) uses a Linear function $f_\omega(x_i^{(q)}) = \langle \omega \cdot x_i^{(q)} \rangle$, and Gradient Descent to find an ideal solution for LDELTR. To use Gradient Descent, we need the derivative of LDELTR ($y^{(q)}$,$\hat{y}^{(q)}$) which in turn consists of the derivatives of the disparate exposure and accuracy metric respectively.

At training time, we are given an annotated set consisting of queries and ordered lists of items for each query. Training documents are ordered by decreasing scores hence the top element is the one that has the highest score. The algorithm learns from training data by minimizing a loss function. At testing time, we provide a query and a document collection, and expect as output a set of top-k items from the collection that should be relevant for the query, and additionally should not exhibit disparate exposure. We should emphasize that scores $x_{i,j}$ are distributed uniformly at random over two non-overlapping intervals.

3.2.1.3 Two different operations of DELTR algorithm

- Case where there is a difference between a standard learning to rank algorithm and DELTR:

  Let's consider a scenario in which all protected elements have strictly smaller scores than all non-protected elements. A standard learning to rank algorithm, in final result-ranking, places all non-protected elements above all protected

elements, giving them a larger exposure. Instead, DELTR with increasing values of γ reduces the disparate exposure, while still considering the discrepancy in the score values. At this point, however, we should emphasize that as we increase the variable gamma, the more protected elements will go up in the final ranking. Of course, this will have the effect of greatly reducing the relevance, as the final result will be quite different from the original one.

- Case where DELTR acts like a standard Learning to rank algorithm:
  If the protected elements already receive larger exposure than the non-protected elements (i.e. in a scenario where all protected elements have strictly larger scores than the nonprotected ones), the DELTR algorithm performs same results as the standard learning to rank approach. Experiments showed that enhancing for fairer results does not necessarily come with a trade-off in relevance.

3.2.2 A post-processing method called FA*IR

3.2.2.1 Mathematical formula of FA*IR:

FA*IR is a post-processing method. In contrast to DELTR, FA*IR assumes that a ranking function has already been trained and that a ranked search result is available. In any given ranking of length k, the percentage of protected elements does not fall far below a given p (p ranges between 0.02 and 0.98) at any ranking position. The above procedure is succeeded by ranked group fairness constraint, which is used by the FA * IR algorithm as a statistical significance test. In more detail FA*IR uses the binomial cumulative distribution function F with parameters p, k and α. The result of this function will be considering as fair if the following inequality is respected.

$$F(\tau_p;k,p)>\alpha$$

where τ_p is the actual number of protected items in the ranking under test. Based on this constraint emerges the minimum percentage of protected groups at each ranking position such that the constraint holds.

Authors ranked group fairness definition extends group fairness using the standard notion of protected groups and is based on ensuring that the proportion of protected candidates in every prefix of the top-k ranking remains statistically above or indistinguishable from a given minimum. The ranked group fairness criterion compares

the number of protected elements in every prefix of the ranking with the expected number of protected elements if they were picked at random using Bernoulli trials with success probability p. We should emphasize that in order for the above comparison to take place we should use a statistical test. This test includes a significance parameter α which depicts the probability of rejecting a fair ranking. An adjusted significance which is called $\alpha_c = c(\alpha, k, p)$ is required because numerous hypothesis (k of them) is required to be tested, so adhere to ranked group fairness definition. If we use $\alpha_c = \alpha$, we might produce false negatives, rejecting fair rankings, at a rate larger than α.

3.2.2.2 How algorithm works:

Researchers based on Yang and Stoyanovich model propose a new algorithm called FA*IR. FA*IR creates a ranking that we will consider fair by:

- starting with an empty list, and
- incrementally adding the best available protected candidate with probability p, or the best available non-protected candidate with probability 1−p.

Algorithm FA*IR finds a ranking that maximizes utility subject to in-group monotonicity and ranked group fairness constraints. FA*IR expects the following variables as arguments-input:

- variable k, which depicts the top documents of final ranking that will be returned,
- the qualifications q_i, namely the score of each document,
- variable g_i, where indicator i shows the category of each document (protected or nonprotected),
- variable p, the minimum proportion of protected elements that should be included in top-k documents and
- the adjusted significance a_c.

First, the algorithm uses q_i to create two priority queues with up to k candidates each:

- P0 for the non-protected candidates and
- P1 for the protected candidates.

Subsequently, the algorithm with the use of a ranked group fairness table and the given parameters p, k and α_c, calculates the minimum proportion of protected elements that should be include in each position so the final ranking remains fair. Then, FA*IR greedily constructs a ranking subject to candidate qualifications, and minimum protected elements required, for the case of a single protected attribute. The elements are selected on the basis of the above-mentioned table with the following modification:

If the computed table m demands a protected candidate at the current position, the algorithm appends the best candidate from P1 to the ranking; otherwise, it appends the best candidate from P0 ∪ P1. FA*IR has running time O (n + k ∗ log k), which includes building the two O(k) size priority queues from n items and processing them to obtain the ranking, where we assume k < O ($n/logn$). In case where we already have two ranked lists for both classes of elements, FA*IR can avoid the first step and obtain the top-k in O(k ∗ log k) time.

The method can work correctly since the percentage of protected groups required in the top-k results of the final ranking did not exceed the number of protected elements contained in the given database. In the case that the above limitation is not complied, the "head" of the ranking will satisfy the ranked group fairness constraint, but the "tail" of the ranking may not.

3.2.2.3 Algorithm correctness

By construction, a ranking τ generated by FA*IR satisfies in-group monotonicity, because protected and non-protected candidates are selected by decreasing qualifications. The following lemmas prove that optimal selection utility, and that it

maximum ordering utility can be achieved by τ. The proof of those lemmas is given in [ZBC+17] paper.

- If a ranking satisfies the in-group monotonicity constraint, then the utility loss (ordering or selection utility different from zero) can only happen across protected/non-protected groups.
- The optimal selection utility among rankings satisfying in-group monotonicity and ranked group fairness is either zero or is due to a non-protected candidate ranked below a less qualified protected candidate.
- Given two rankings ρ, τ satisfying in-group monotonicity, if they have the same number of protected elements $\rho_p = \tau_p$, then both rankings contain the same k elements (possibly in a different order), and hence both rankings have the same selection utility.
- Algorithm FA*IR achieves optimal selection utility among rankings satisfying in-group monotonicity and ranked group fairness.
- Algorithm FA*IR maximizes ordering utility among rankings satisfying in-group monotonicity, ranked group fairness, and achieving optimal selection utility.

3.2.2.4 Fair top-k ranking criteria:

Ranked group fairness Criterion: τ should fairly represent the protected group

In order to decide if a ranking is fair, the algorithm computes the number of protected elements in every prefix of the ranking. Afterwards these numbers are compared with the expected number of protected elements if they were picked at random using Bernoulli trials. The criterion is based on a statistical test, and a significance parameter α corresponding to the probability of rejecting a fair ranking is included. A larger α means a larger probability of declaring a fair ranking as unfair. The ranked group fairness condition can be used to create a ranked group fairness measure. For a ranking τ and probability p, the ranked group fairness measure is the maximum α ∈ [0,1] for which τ satisfies the ranked group fairness condition. Larger values of parameter α means that more protected elements are required in each position of the ranking.

Selection utility Criterion: τ should contain the most qualified candidates

The selection utility criterion states that rankings in which the more qualified candidates are included, and the less qualified excluded, are preferred. This criterion makes use of the ranked utility. The ranked individual utility associated with a candidate (i) in a ranking τ, compares it against the least qualified candidate ranked above it. By this definition, the maximum ranked individual utility that can be attained by an element is zero.

Ordering utility Criterion: Ordering utility τ should be ordered by decreasing qualification

The ordering utility states that the top-k result of ranking includes candidates classified in descending order based on their score, and in every pair of candidates either the better candidate proceeds the worst candidate or the difference in their qualifications-score is small. The ordering utility of a ranking is only concerned with the candidate attaining the worst (minimum) ranked individual utility. Instead, the in-group monotonicity constraints refer to all elements and specify that both protected and non-protected candidates, independently, must be sorted by decreasing qualifications.

3.2.2.5 How FA*IR decides if the given ranking is not fair:

First way: (intractable)

The probability of considering this ranking of k elements (m−1(k)blocks) unfair, is:

$$1 - \sum_{v \in I_{m^{-1}(k)}} \prod_{j=1}^{m^{-1}(k)} f(v_j; b(j), p)$$

Where $f(x; b(j), p) = Pr(X = x)$ is the probability density function of a binomially-distributed variable X ~Bin(b(j),p).

Second way:

With algorithm ADJUSTSIGNIFICANCE, which takes as input the size of the ranking to produce (k), the expected proportion of protected elements (p) and the significance for each individual test (a_c). Adjust - Significance give us as output the probability of rejecting a fair ranking.

3.2.3 Extension of FA*IR functionality

In this thesis, we have introduced a new method based on FA*IR algorithm. In particular, we extended FA*IR functionality, to handle databases that consist of m groups of individuals. Initially, the FA*IR algorithm worked only for databases that consisted of two groups of individuals (for example nonprotected = men and protected = women). So, we wanted to extend the functionality of the FA * IR algorithm in order to work for m groups of individuals (three protected groups and one nonprotected).

In the beginning, we divided the problem into m-1 sub-problems (which are essentially the categories of protected groups). For each sub-problem, we follow the same procedure:

- We assume as protected group one of our m-1 protected groups of individuals (x) and the rest of them as nonprotected. (with the only condition x belongs in the protected categories that we have set in the beginning). Then, we run the FA * IR algorithm and keep top-k results.

The above procedure will occur in total of m-1 times (one for each sub-problem), therefore, m-1 new top-k rankings will emerge. At this point we will merge the m-1 lists that have been produced by the previous procedure in one new list which will contain the top-k results of the m-1 rankings. To keep the algorithm fair, we should take into account the value p, which determines the minimum percentage of protected groups in the final top-k ranking. It should be mentioned here that value p depends on AdjustMTable, which is responsible for keeping the algorithm fair. The algorithm uses m-1 tables type AdjustMTable, one for each subcategory of the protected groups. At the time of merging, we import at least as many documents from each protected category as the tables above indicate. If the final ranking does not obtain k documents, we will fill in with the best documents from the m-1 lists that have not been used.

Standard proof for the correct operation of our extension for FA*IR algorithm:

The above algorithm ensures that the three requested protected categories exist in the final ranking - result. This is ensured during the merge process when we imported results from each category. The minimum required number of documents from each protected category is given by the AdjustedMTable. The proof of the proper function of the above table (AdjustedMTable), that contains the minimum required documents in order to be fair the final ranking has been already given by the authors [ZBC+17] of FA*IR algorithm. Since the sum of the percentages of the three categories does not exceed the unit, the documents entered in the final ranking are less than or equal to k. Therefore, if the final results are less than k, more documents should be inserted. New documents to be added are selected among the best-ranked documents each time. This means that we get the best possible score based on the initial ranking and since we have ensured that there is an adequate number of protected documents from each category. Based on the foregoing, a category may have more documents than required but this is not a problem as the classification to be fair to cover the minimum number. As for the ranking, the algorithm presents the results in descending order based on the score, as when a new document is inserted the list remains sorted.

Proof by contradiction:

Assume that the ranking is not fair to one protected category. This means it has fewer documents in the output than needed. Invalid, as the final results include at least as many documents from each protected category so that the final ranking is fair, considering only two categories, the protected one and the rest as non-protected. So, the ranking is fair in this category and therefore in all.

Suppose there is a higher-scoring document that has not entered the results. Suppose it belongs to a protected category and is among the first documents to qualify the ranking as fair. Invalid, because we put the first documents in descending order by score. Suppose now that the document belongs to an unprotected category or belongs to one of the three protected categories and is ranked lower than the minimum number of documents that should be included in the final result that are needed to be fair the final ranking. Invalid, as top-k results are completed by getting the highest scoring document each time. So, there is no document with a higher score that should be in the results.

Chapter 4. Experimental Evaluation

4.1 Detailed presentation of DELTR results

SAT Database: SAT corresponds to scores in the US Scholastic Assessment Test, a standardized test used for college admissions in the US. We create this data, sampled from the actual distribution of SAT results from 2014. We modify the database in order to collect male and women students with unique test scores on the SAT. The algorithm is divided into 2 phases (training, testing):

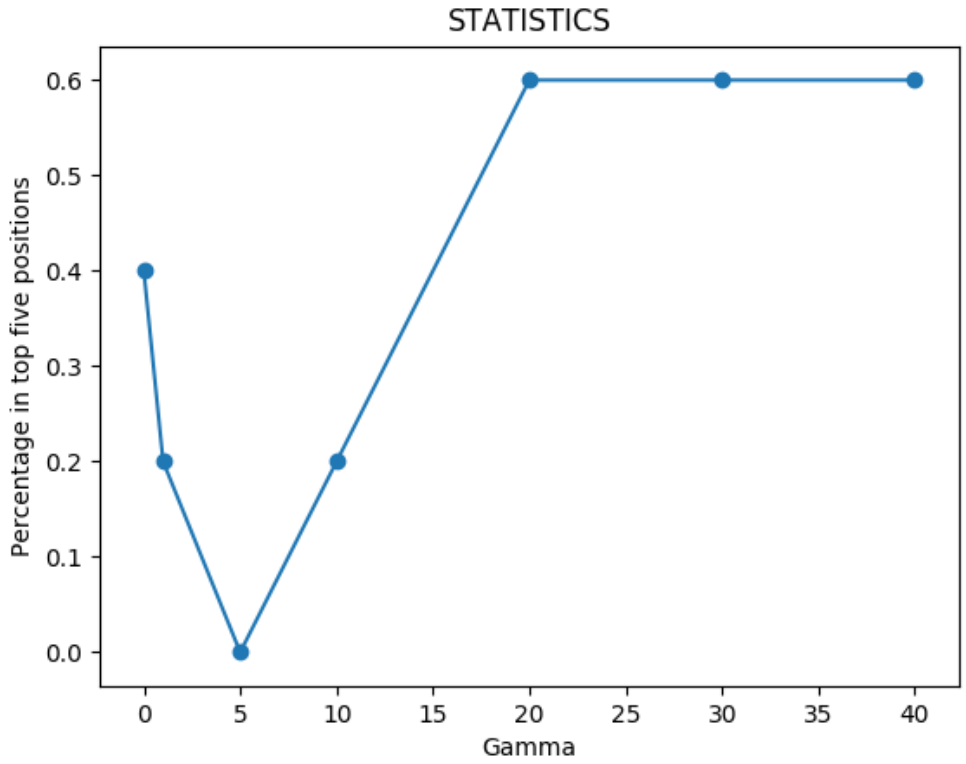
- At training time, we are given an annotated set consisting of queries and ordered lists of items for each query. Training documents are ordered by decreasing scores hence the top element is the one that has the highest score. Moreover, DELTR needs: a query_id = 1, doc_id (counting of documents in ascending order), gender (1 for womens, 0 for men) and judgment in descending order.
- At testing time, we provide a query and a document collection, and expect as output a set of top-k items from the collection that should be relevant for the query, and additionally should not exhibit disparate exposure. For testing phase, we need four features: query_id = 1, doc_id (counting of documents in ascending order), score in descending order and gender (0 for womens, 1 for men).

The purpose of DELTR is to rank protected group (in our case the protected group are women) higher than the original ranking. The rise of women in rankings is directly related with parameter gamma. When we prefer larger γ value we express a preference for solutions that reduce disparate exposure for the protected group and when we prefer smaller γ value we express a preference for solutions that reduce the differences between the training data and the output of the ranking algorithm.

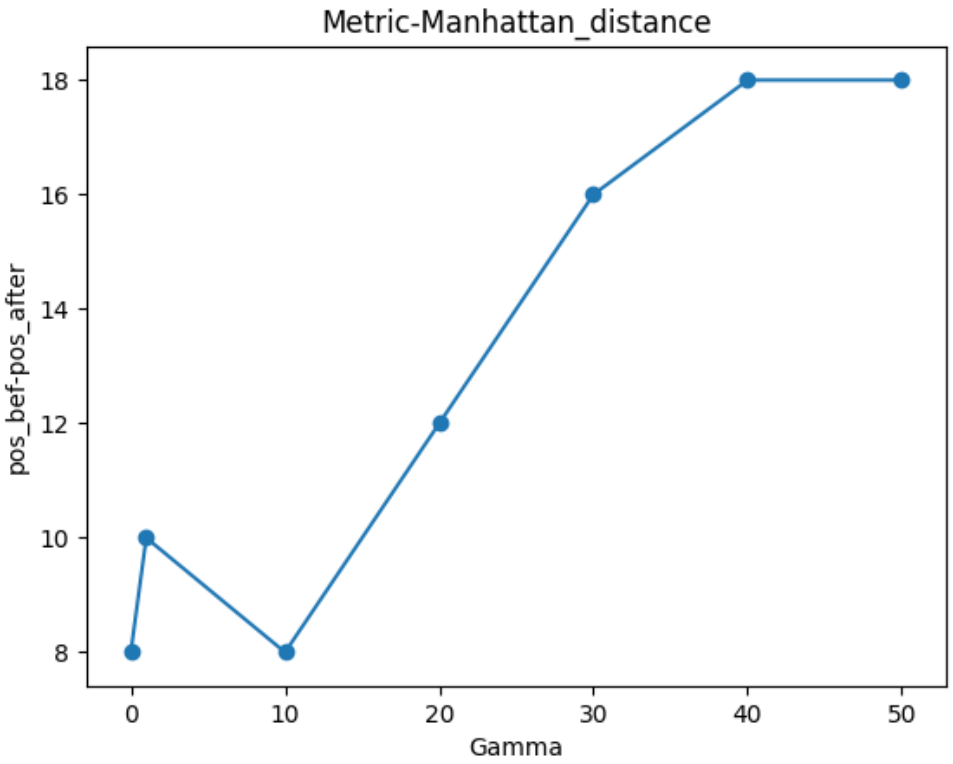
SAT_DATABASE: SAT corresponds to scores in the US Scholastic Assessment Test, a standardized test used for college admissions in the US. We create this data, sampled from the actual distribution of SAT results from 2014. The protected group is women (protected = 20) and the nonprotected group is men (nonprotected = 46).

General observation on how the increase in the gamma parameter affects the final ranking: In the beginning of our test women hold the last position. We observer that when gamma parameter is increased women ranked higher than the original ranking.

In this graph, we can observe what percentage of the protected group is in the top 5 positions in relation to the gamma parameter. As we can see when parameter gamma >= 20 all the protected features are in top – 5 positions. With this metric we depict the degree of fairness of the DELTR algorithm.

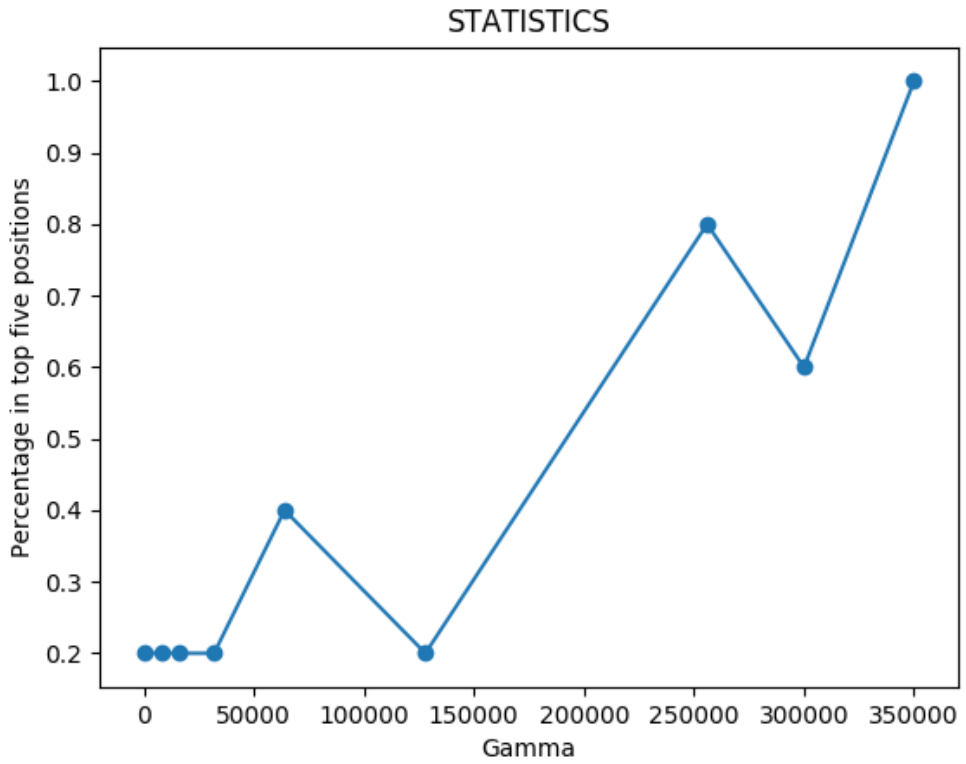


In this graph, we implement the Manhattan metric. The aim of this metric is to help us understand the differences between the original ranking and the resulting rankings (with different values of gamma parameter).

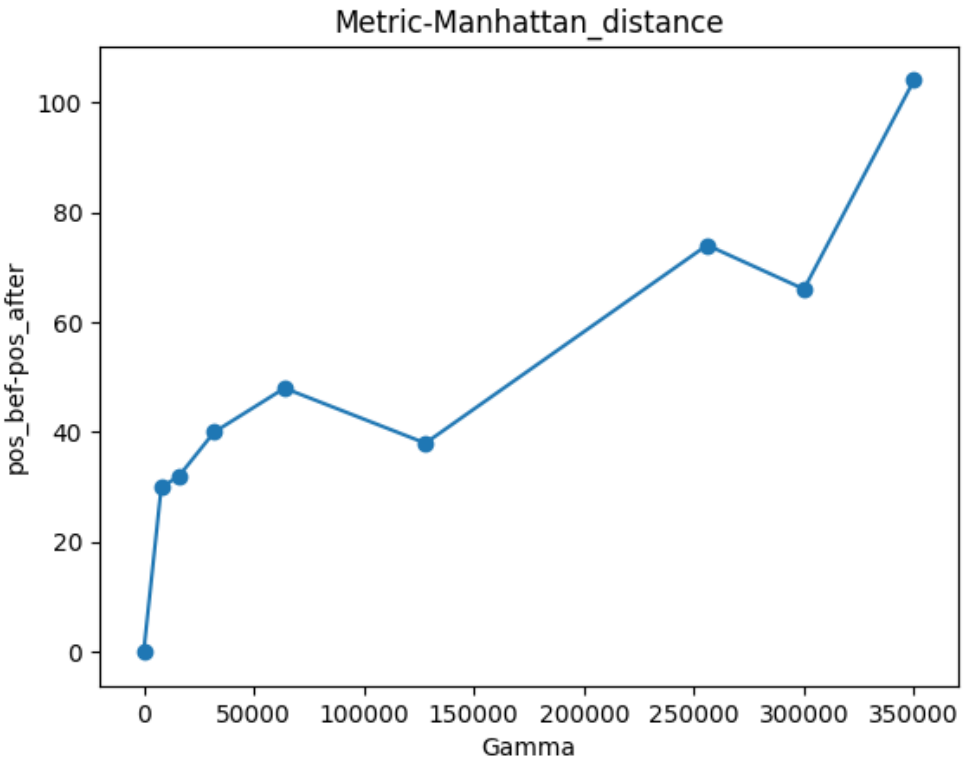


SAT_DATABASE (variation): The protected group is women (protected = 38) and the nonprotected group is men (nonprotected = 62).

In this graph, we can observe what percentage of the protected group is in the top 5 positions in relation to the gamma parameter. As we can see when parameter gamma >= 20 all the protected features are in top – 5 positions. With this metric we depict the degree of fairness of the DELTR algorithm.



In this graph, we implement the Manhattan metric. The aim of this metric is to help us understand the differences between the original ranking and the resulting rankings (with different values of gamma parameter). When the gamma parameter = 350000 then all the protected elements are at the top of the ranking.



4.2 Detailed presentation of FA*IR results

Test for our new database (Predict whether income exceeds $50K/year and race = Asian-Pac-Islander based on census data) with NDCG and UTILITY metrics. Below we quote the list of tests we ran in order to test how FA*IR behaves.

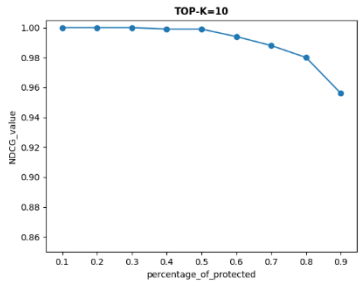
Protected (true) = 44// Nonprotected (false) = 232

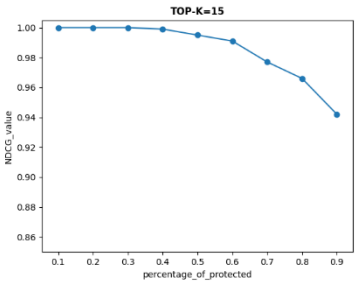
Link for database: https://archive.ics.uci.edu/ml/datasets/Adult.

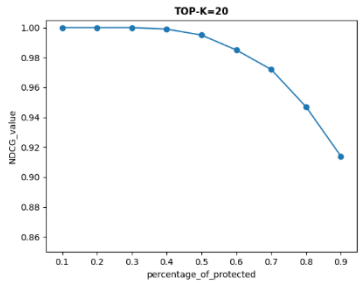
NDCG	K=10	K=15	K=20	K=25	K=30	K=35	K=40
P=0.1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.2	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.3	1.0	1.0	1.0	0.999	0.999	0.999	0.999
P=0.4	0.999	0.999	0.999	0.997	0.997	0.994	0.993
P=0.5	0.999	0.995	0.995	0.991	0.986	0.984	0.981
P=0.6	0.994	0.991	0.985	0.975	0.972	0.965	0.961
P=0.7	0.988	0.977	0.972	0.962	0.953	0.943	0.934
P=0.8	0.98	0.966	0.947	0.937	0.919	0.909	0.893
P=0.9	0.956	0.942	0.914	0.893	0.874	0.858	0.839

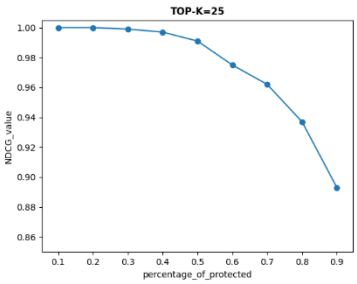
utility	K=10	K=15	K=20	K=25	K=30	K=35	K=40
P=0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.3	0.0	0.0	0.0	0.021	0.046	0.056	0.09
P=0.4	0.018	0.012	0.05	0.137	0.247	0.503	0.678
P=0.5	0.037	0.15	0.222	0.475	0.794	1.141	1.51
P=0.6	0.112	0.265	0.553	1.053	1.451	2.073	2.673
P=0.7	0.209	0.575	0.969	1.551	2.232	3.118	4.118
P=0.8	0.309	0.803	1.588	2.333	3.521	4.619	6.032
P=0.9	0.616	1.235	2.345	3.596	5.041	6.536	8.431

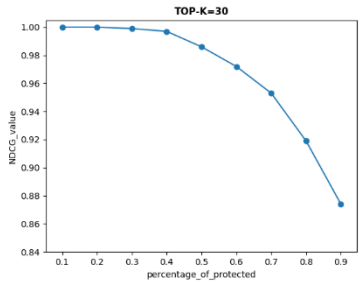
NDCG METRIC: We report a normalized weighted summation of the quality of the elements in the ranking, $\sum_{i=1}^{k} w_i q(\tau_i)$, in which the weights are chosen to have a logarithmic discount in the position: $w_i = 1/\log2(i+1)$. This is a standard measure to evaluate search rankings. This is normalized so that the maximum value is 1.0.

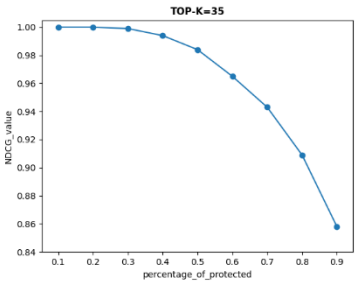


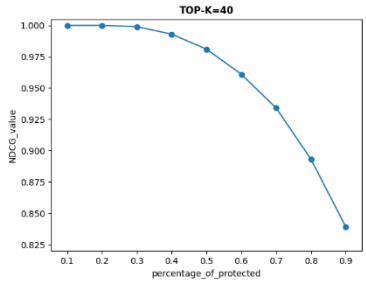






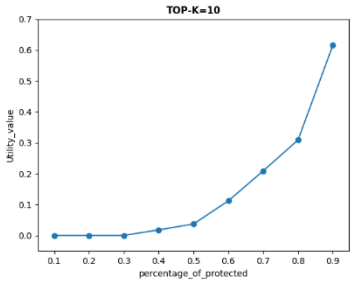


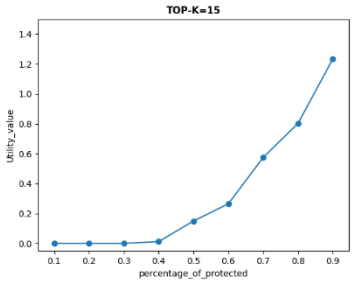


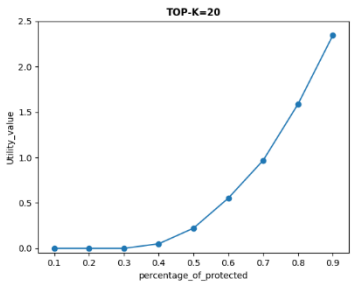


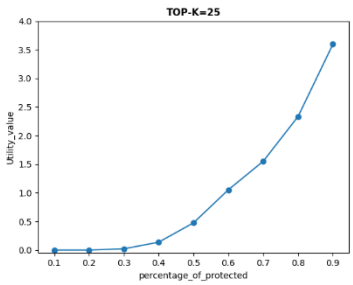
UTILITY METRIC: We report the loss in ranked utility after score normalization, in which all qi are normalized to be within [0,1]. We also report the maximum rank drop, i.e., the number of positions lost by the candidate that realizes the maximum ordering utility loss.

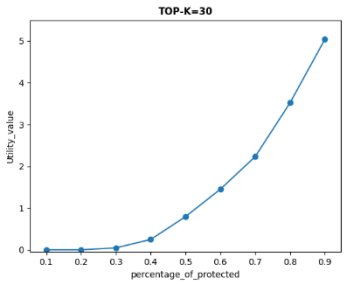
Observation: Fair wants the new ranking to be as close as possible to the original ranking. The more the final ranking differs from the original one, the higher the Utility_loss. Utility is calculated based on losses caused by non-monotony.

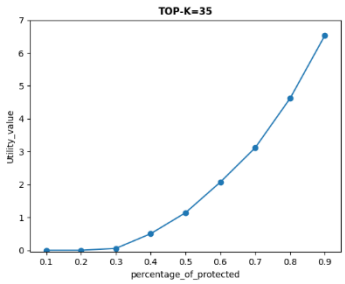


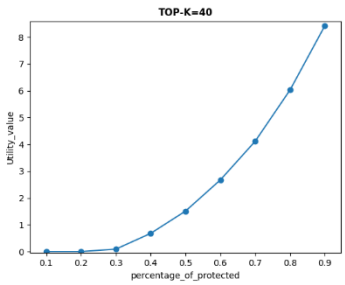












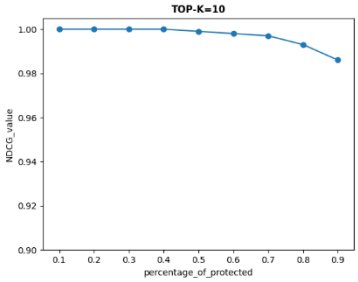
SAT_DATABASE: SAT corresponds to scores in the US Scholastic Assessment Test, a standardized test used for college admissions in the US. We create this data, sampled from the actual distribution of SAT results from 2014. The qualification attribute is set to be the achieved SAT score, and the protected group is women(protected=89).

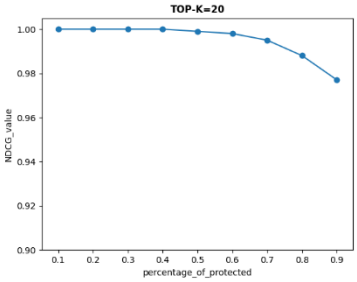
http://www.fairness-measures.org/Pages/Datasets/SAT.html

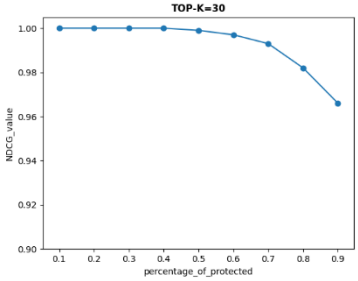
NDCG	K=10	K=20	K=30	K=40	K=50	K=60	K=70	K=80
P=0.1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.5	0.999	0.999	0.999	0.999	0.999	0.998	0.998	0.998
P=0.6	0.998	0.998	0.997	0.996	0.995	0.992	0.991	0.989
P=0.7	0.997	0.995	0.993	0.989	0.984	0.98	0.977	0.97
P=0.8	0.993	0.988	0.982	0.976	0.969	0.959	0.952	0.942
P=0.9	0.986	0.977	0.966	0.956	0.941	0.93	0.914	0.903

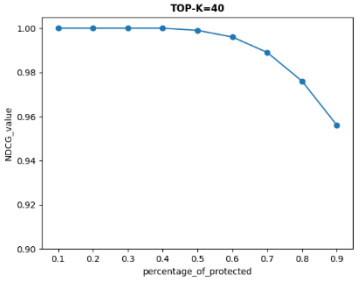
utility	K=10	K=20	K=30	K=40	K=50	K=60	K=70	K=80
P=0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.5	0.01	0.025	0.055	0.097	0.187	0.338	0.503	0.736
P=0.6	0.032	0.127	0.288	0.521	0.869	1.35	1.909	2.571
P=0.7	0.065	0.253	0.553	1.065	1.774	2.621	3.631	4.877
P=0.8	0.11	0.428	0.977	1.771	2.791	4.177	5.714	7.576
P=0.9	0.192	0.669	1.506	2.629	4.203	6.015	8.28	10.709

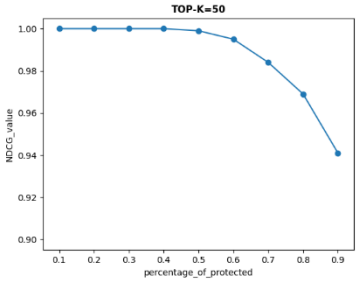
NDCG METRIC:

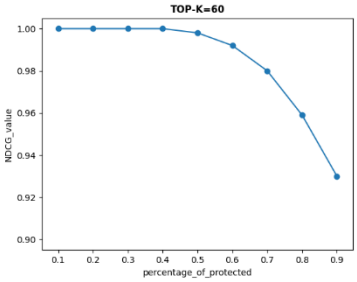

TOP-K=10
NDCG_value
percentage_of_protected

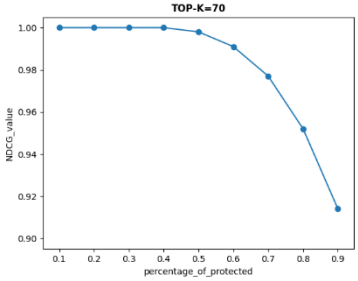

TOP-K=20
NDCG_value
percentage_of_protected

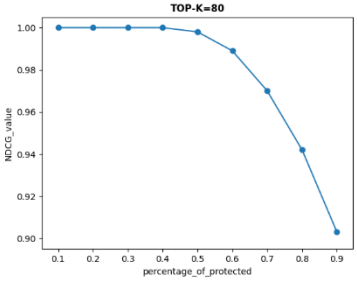

TOP-K=30
NDCG_value
percentage_of_protected


TOP-K=40
NDCG_value
percentage_of_protected

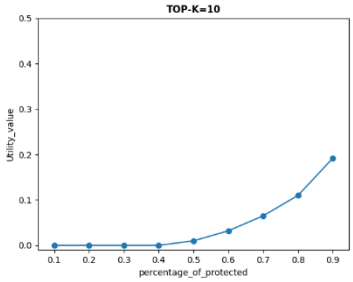

TOP-K=50
NDCG_value
percentage_of_protected

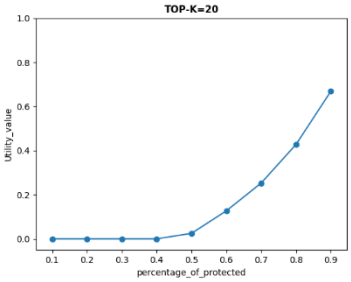

TOP-K=60
NDCG_value
percentage_of_protected

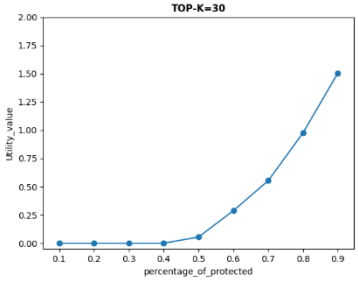

TOP-K=70
NDCG_value
percentage_of_protected

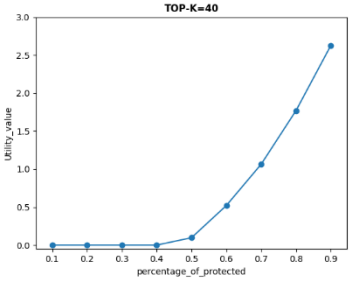

TOP-K=80
NDCG_value
percentage_of_protected

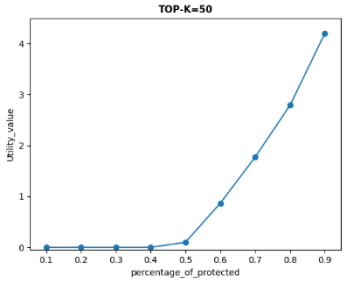
Utility metric:

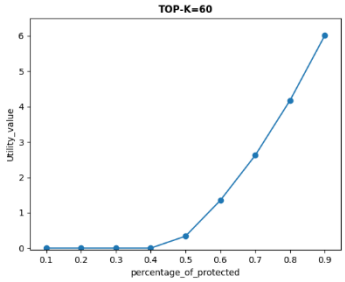


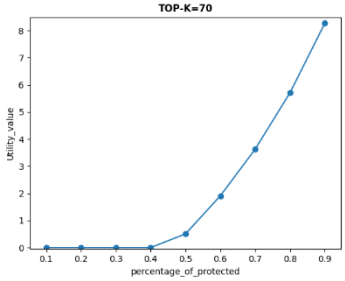


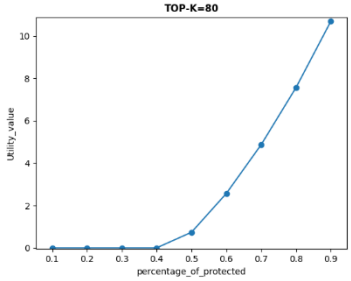












Credit Card Clients DATABASE: Test for our new database (includes the available amount, each customer has in their bank accounts) with NDCG and UTILITY metrics. Below we quote the list of tests we ran in order to test how FA*IR behaves.

Protected (true - women) = 285 // Nonprotected (false - men) = 215

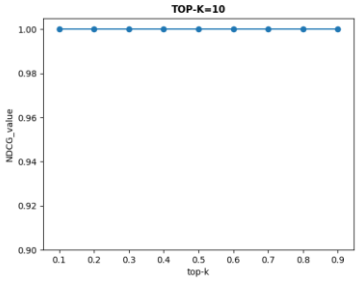
Link for database:

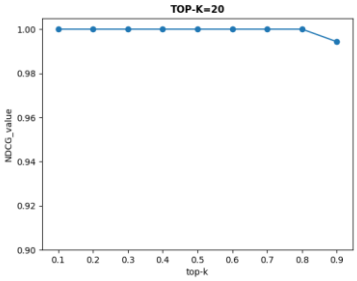
https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

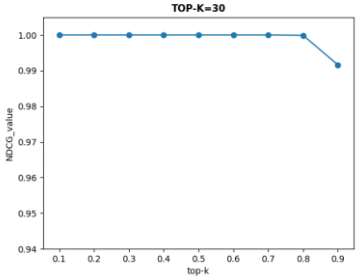
NDCG	K=10	K=20	K=30	K=40	K=50	K=60	K=70	K=80
P=0.1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P=0.7	1.0	1.0	1.0	1.0	1.0	1.0	0.9999	0.995
P=0.8	1.0	1.0	0.9999	1.0	0.999	0.995	0.9934	0.9904
P=0.9	1.0	0.9943	0.9905	0.9916	0.989	0.9832	0.9749	0.9694

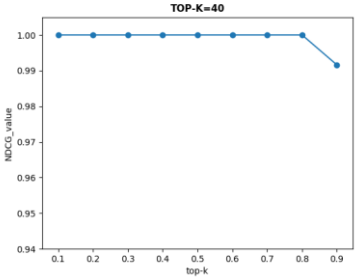
utility	K=10	K=20	K=30	K=40	K=50	K=60	K=70	K=80
P=0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P=0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0317	0.1269
P=0.8	0.0	0.0	0.0317	0.0	0.0952	0.3492	0.7619	1.222
P=0.9	0.0	0.1428	0.5396	1.031	1.444	2.1904	3.015	3.9206

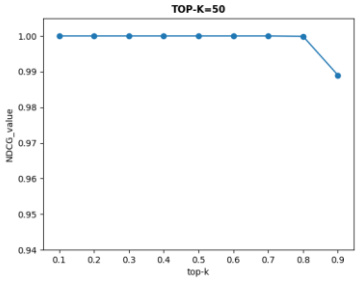
NDCG METRIC:

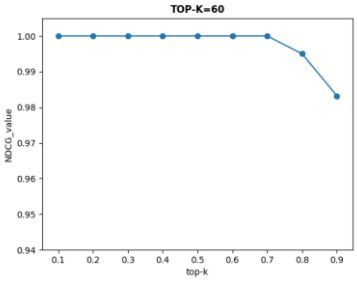

TOP-K=10
NDCG_value
top-k

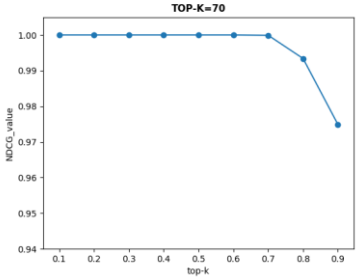

TOP-K=20
NDCG_value
top-k

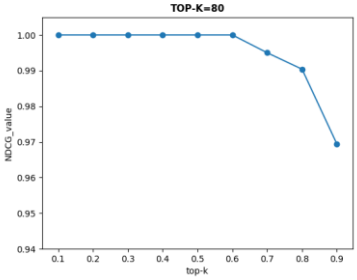

TOP-K=30
NDCG_value
top-k


TOP-K=40
NDCG_value
top-k

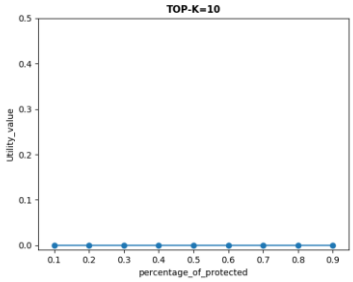

TOP-K=50
NDCG_value
top-k

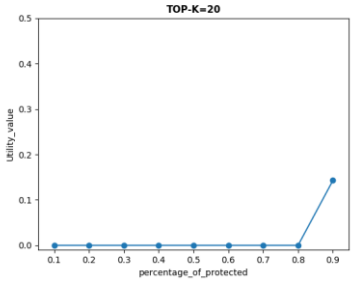

TOP-K=60
NDCG_value
top-k

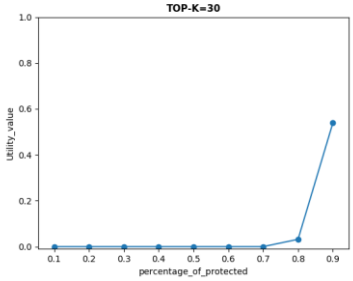

TOP-K=70
NDCG_value
top-k

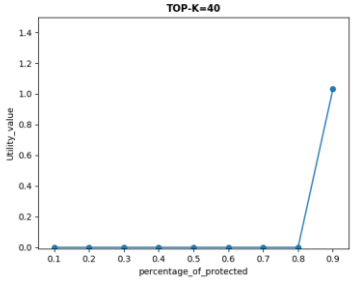

TOP-K=80
NDCG_value
top-k

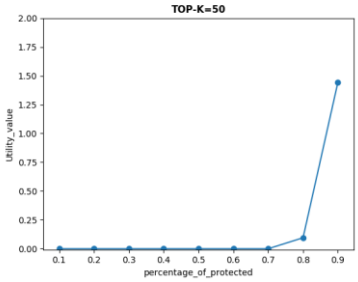
Utility metric:

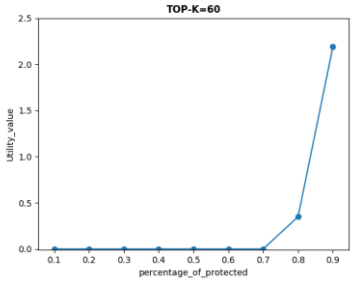


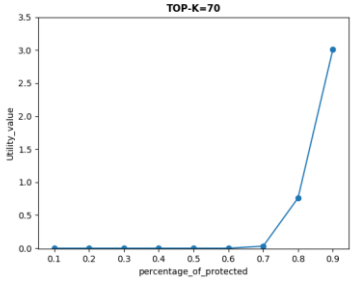


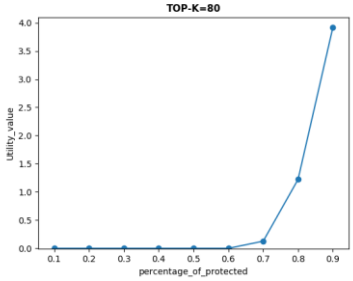












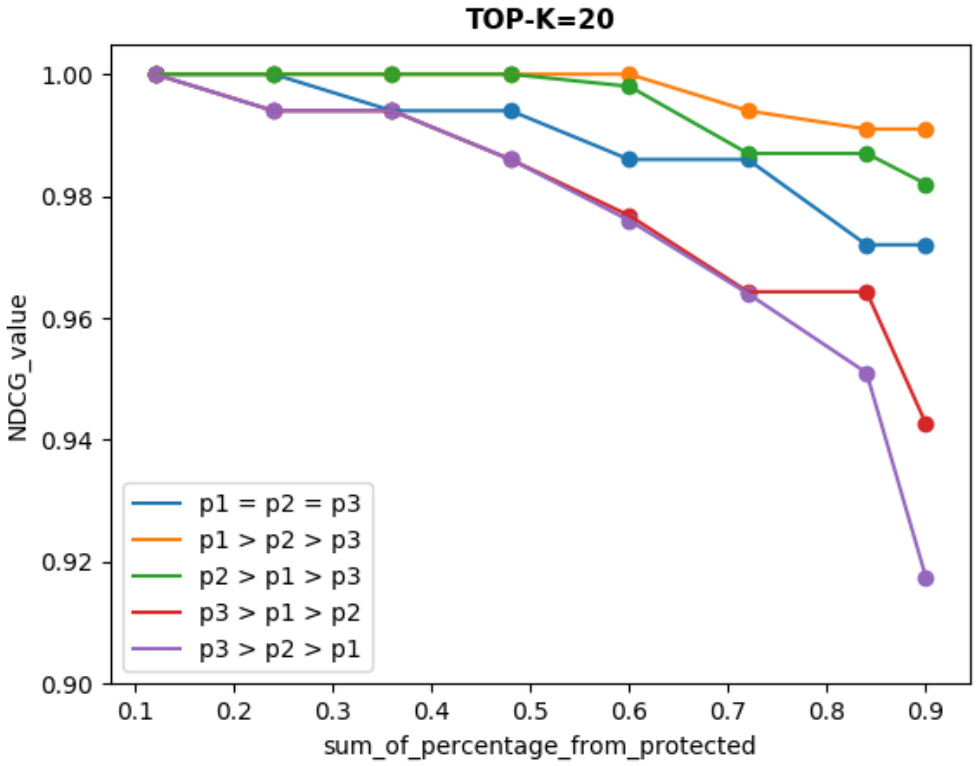
4 groups database: Test for our new database (Predict whether income exceeds $50K/year based on census data) with NDCG and UTILITY metrics. Below we quote the list of tests we ran in order to test how the extension of FA*IR behaves. We have one nonprotected category (race = White) and three protected categories (race = Asian-Pac-Islander, race = Black, race = American-Indian-Eskimo).

Whites = 170 (the nonprotected category) ||

Black = 80 (are illustrated by the symbol p1 in the graph below) ||

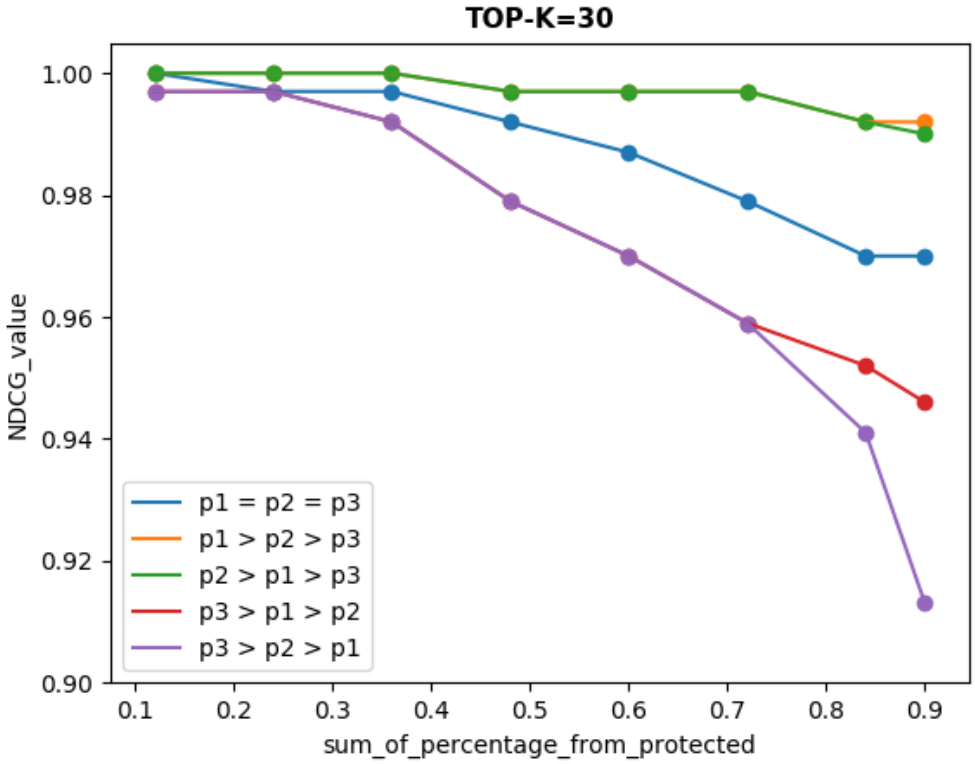
Asian-Pac-Islander = 120 (are illustrated by the symbol p2 in the graph below) || American-Indian-Eskimo= 30 (are illustrated by the symbol p3 in the graph below)

NDCG metric:

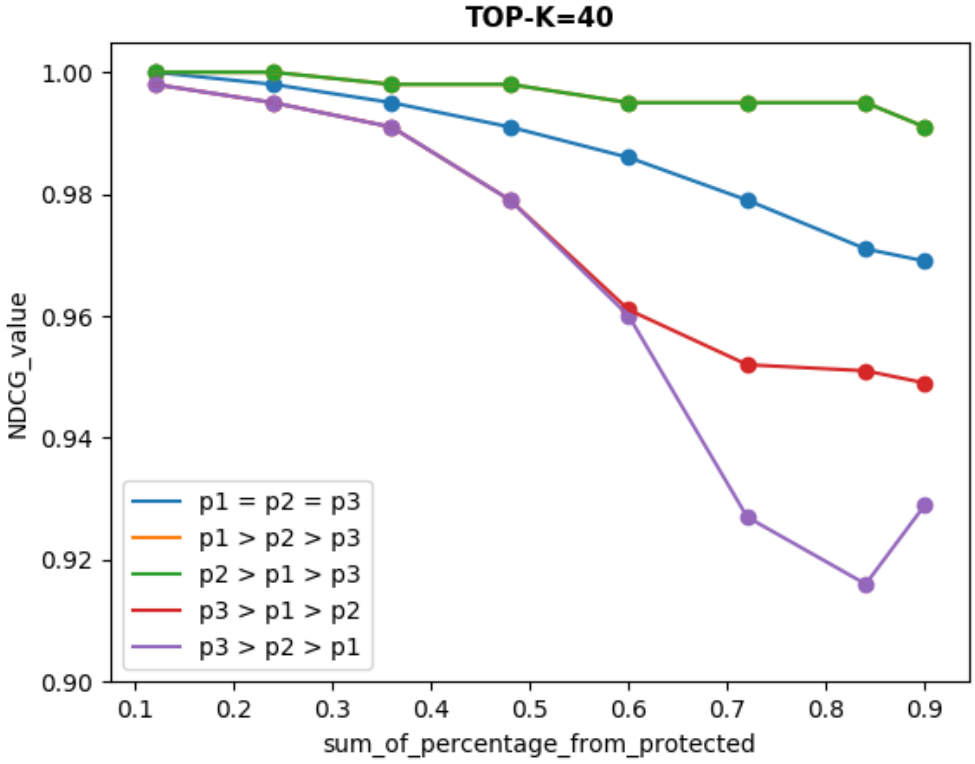


Based on this metric we can arise the following conclusions:

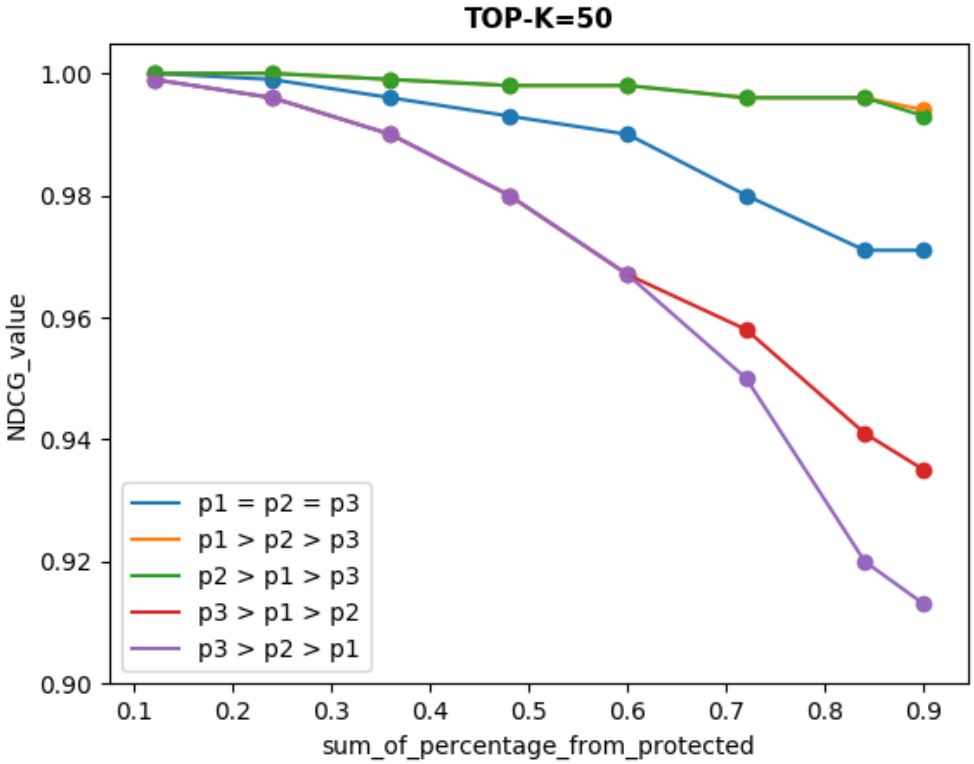
- When p1 has the highest value and p2 the second-highest then the resulting ranking is similar to the original one.
- When p3 has the highest value and p2 the second-highest then the resulting ranking is quite different from the original one.



In this metric we should mention that yellow and green line accept same values, as a result yellow line is not obvious in the graph.

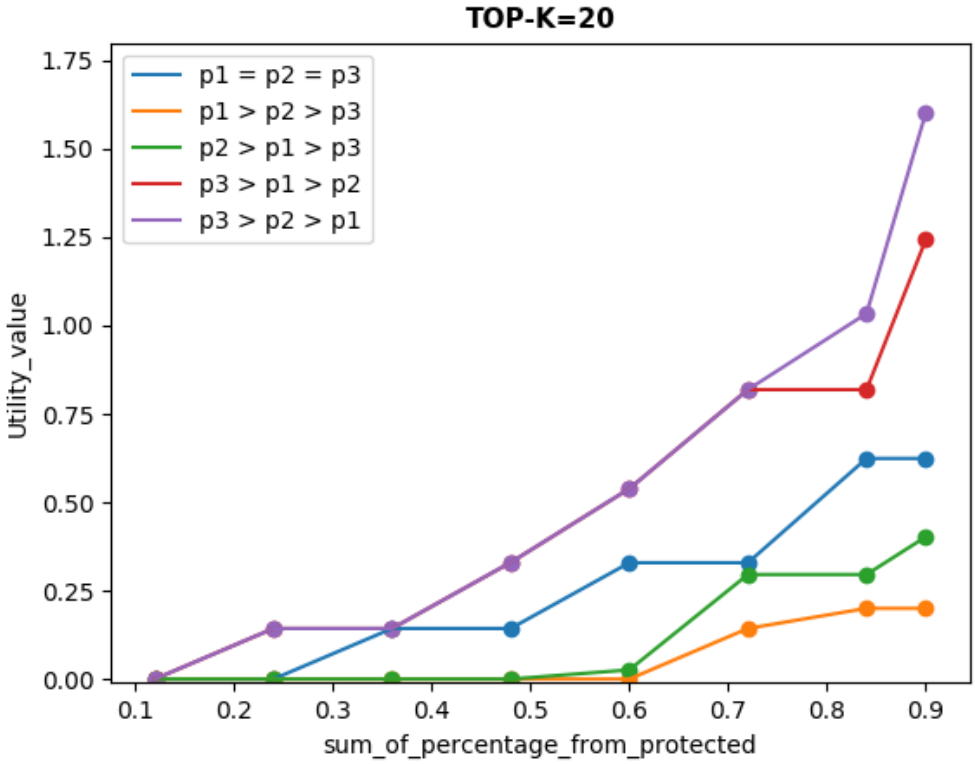


In the above metric we should mention that yellow and green line accept same values, as a result yellow line is not obvious in the graph. Moreover, purple line receives the lowest value when the sum of percentages is equal to 0.8 (instead of 0.9 as expected). This algorithm behavior is justified if we take a closer look to the initial and final ranking. This is happening because the non-protected group is not included the top positions of the original ranking.



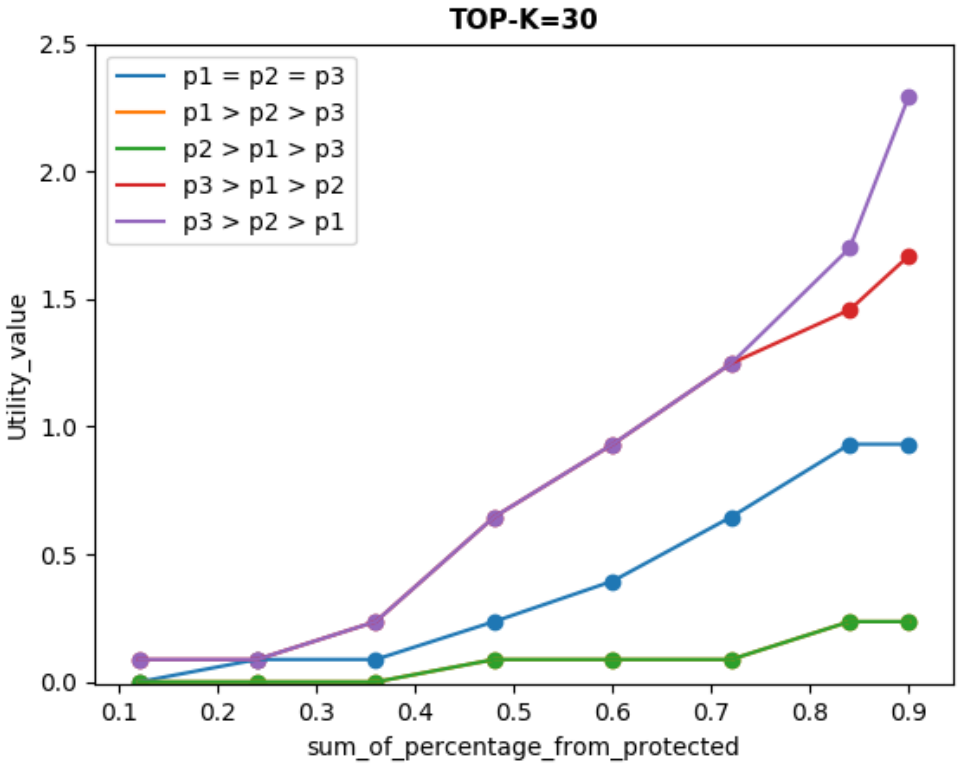
In this metric we should mention that yellow and green line accept almost same values and as a result yellow line is not obvious in the graph.

Utility metric:

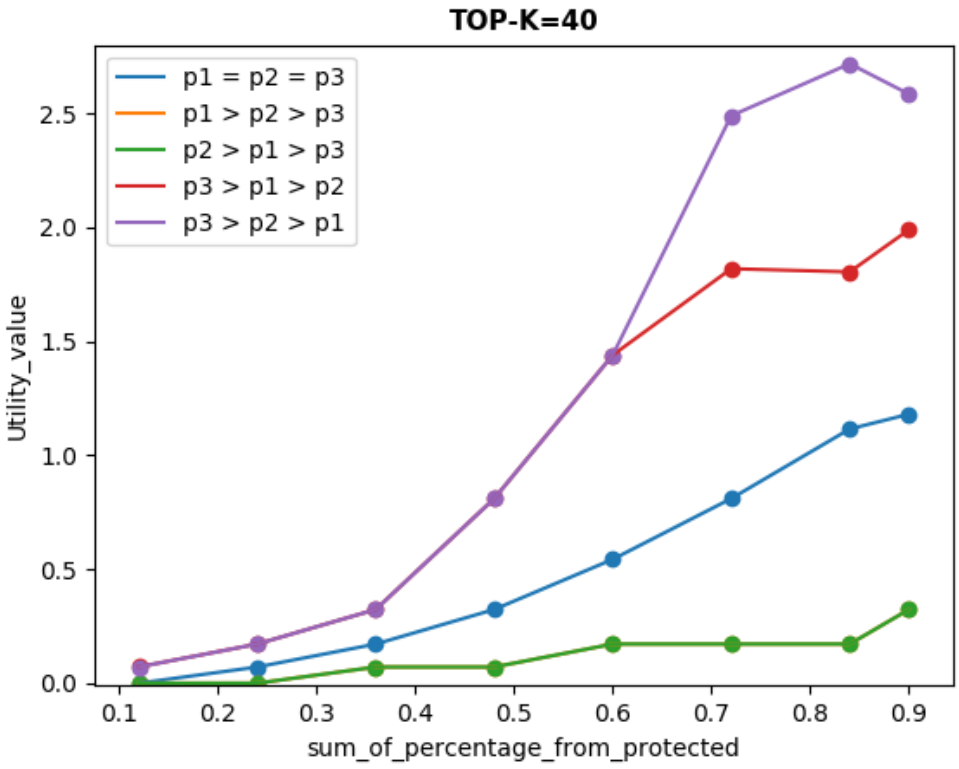


Based on this metric we can arise the following conclusions:

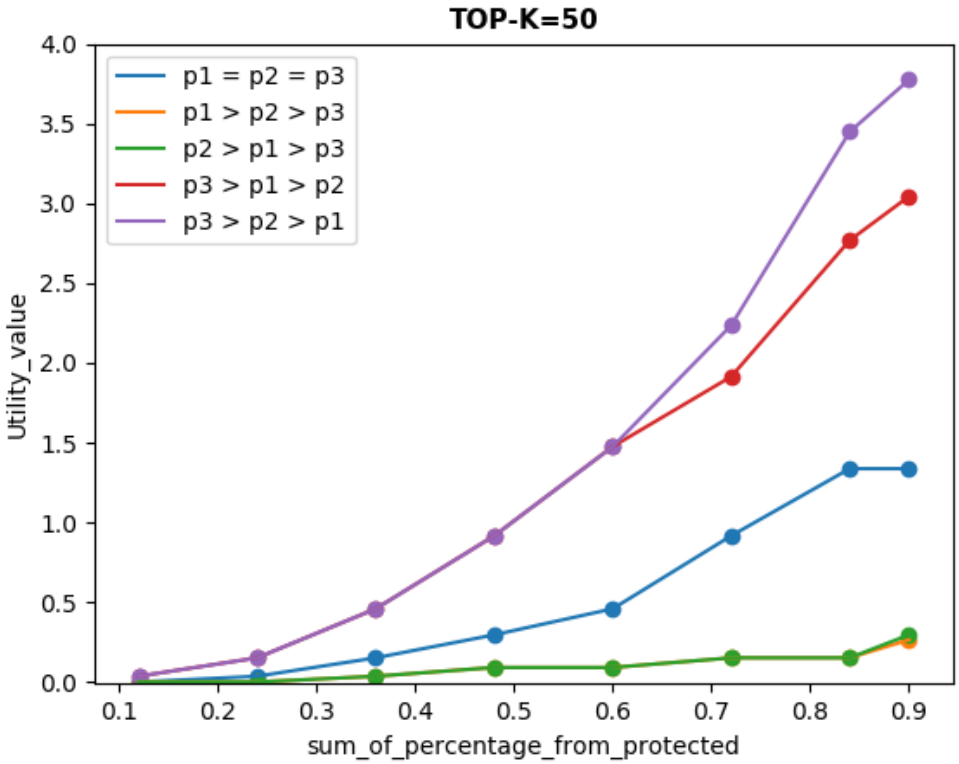
- When p1 has the highest value and p2 the second-highest then the resulting ranking is similar to the original one. As we can see the orange line takes small utility values.
- When p3 has the highest value and p2 the second-highest then the resulting ranking is quite different from the original one. As we can see the purple line takes the highest utility values.
- When the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph.



In this metric we should mention that purple and red line accept almost same values and as a result red line is not so obvious in the graph.



In the above metric we should mention that purple line receives the highest value when the sum of percentages is equal to 0.8 (instead of 0.9 as expected). This algorithm behavior is justified if we take a closer look to the initial and final ranking. This is happening because the non-protected group is not included the top positions of the original ranking.



In this metric we should mention that yellow and green line accept almost same values and as a result yellow line is not obvious in the graph.

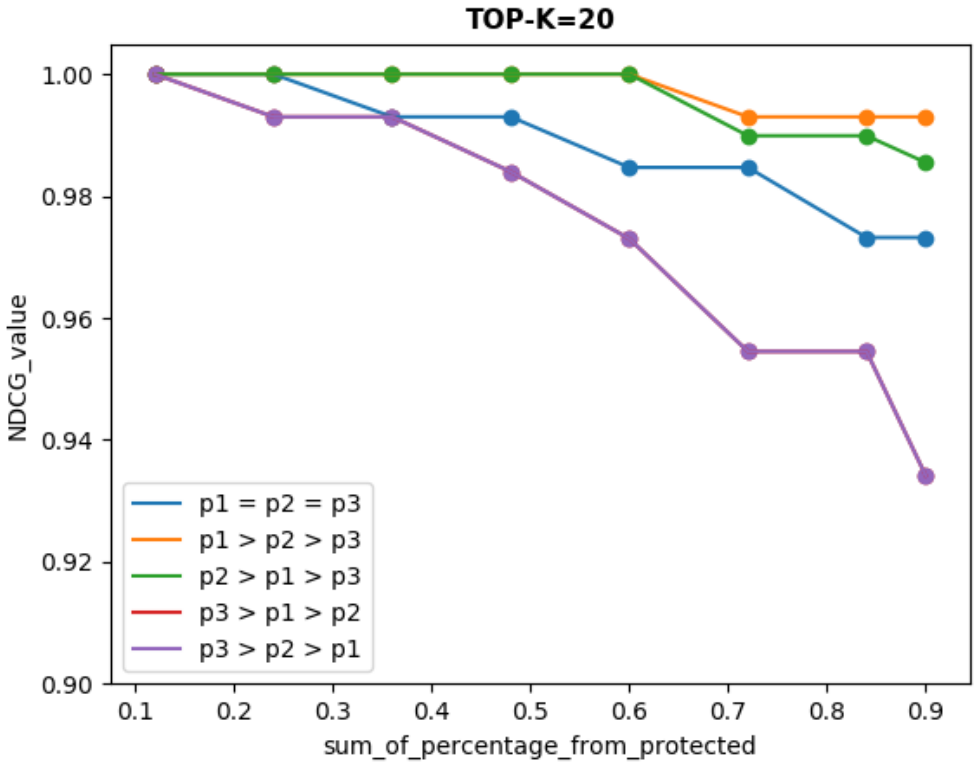
4 groups database (variation I): Test for our database (Predict whether income exceeds $50K/year based on census data) with NDCG and UTILITY metrics. Below we quote the list of tests we ran in order to test how the extension of FA*IR behaves. We have one nonprotected category (race = White) and three protected categories (race = Asian-Pac-Islander, race = Black, race = American-Indian-Eskimo).

Whites = 100 (the nonprotected category) ||

Black = 120 (are illustrated by the symbol p1 in the graph below) ||

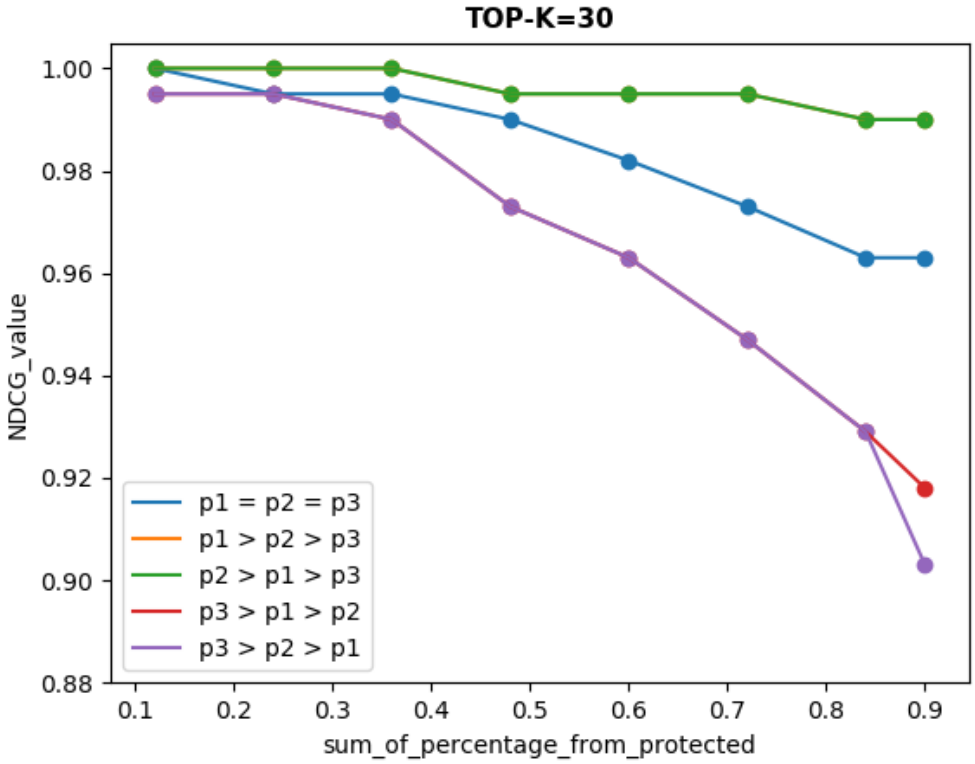
Asian-Pac-Islander = 160 (are illustrated by the symbol p2 in the graph below) || American-Indian-Eskimo= 20 (are illustrated by the symbol p3 in the graph below)

NDCG metric:

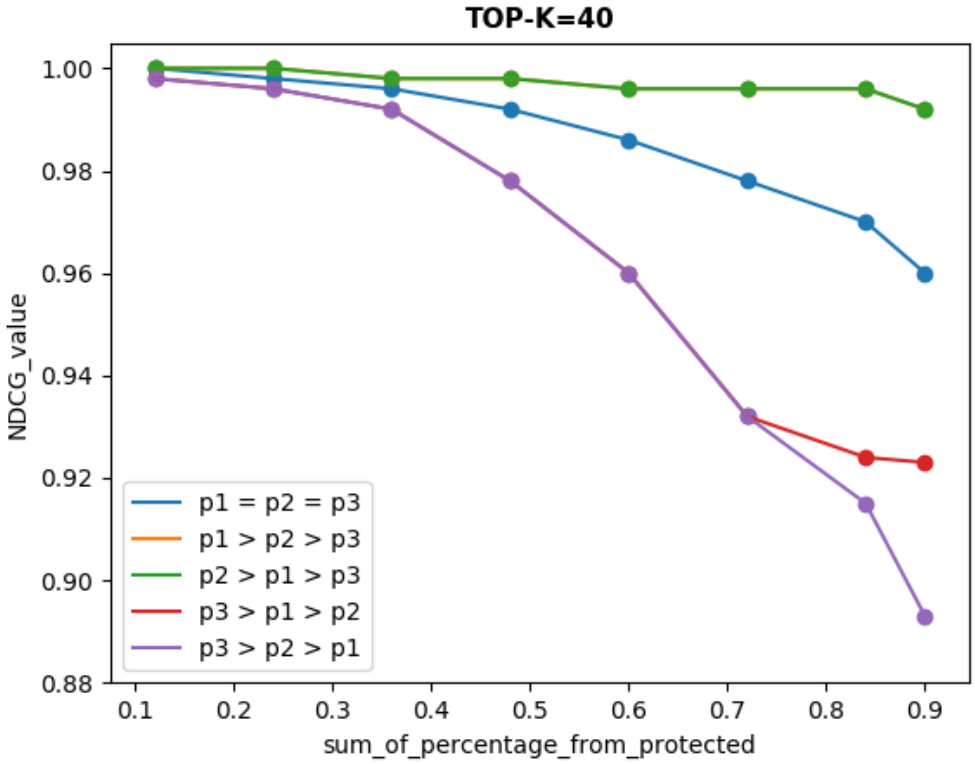


Based on this metric we can arise the following conclusions:

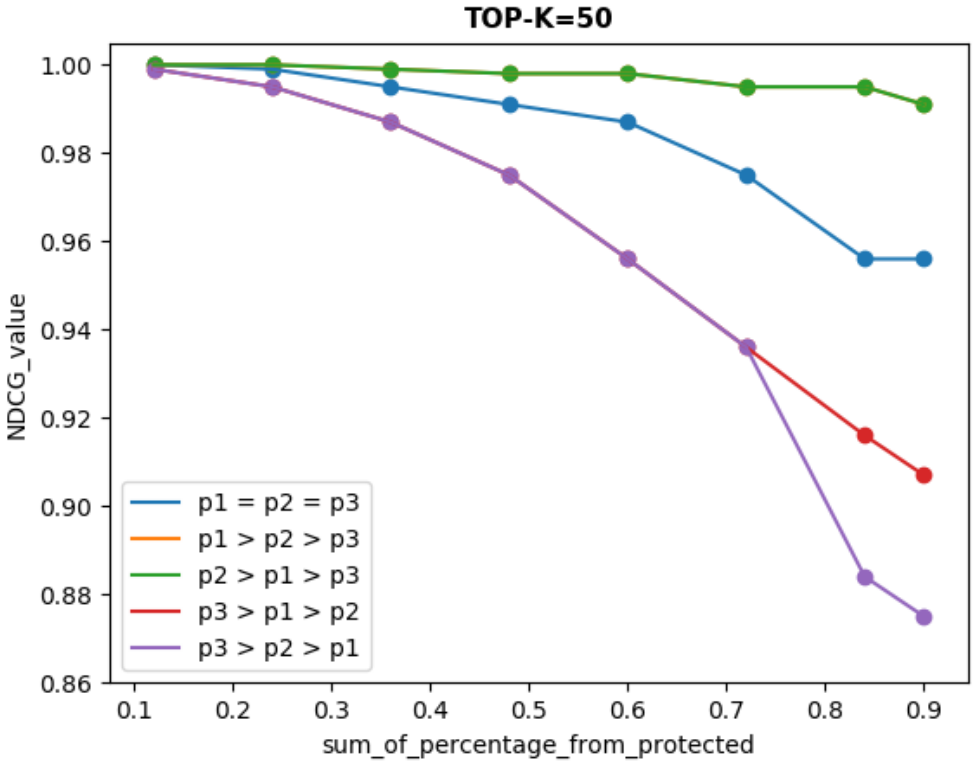
- When p1 has the highest value and p2 the second-highest then the resulting ranking is similar to the original one.
- When p3 has the highest value and p2 the second-highest then the resulting ranking is quite different from the original one.
- Finally, we should mention that red line (p3>p1>p2) and purple line (p3>p2>p1) have exactly the same results, so the red line is not appearing in the graph.



In this metric we should mention that yellow and green line accept exactly same values and as a result yellow line is not obvious in the graph. In addition, purple line accepts almost same values with red line. As a result, the red line appears slightly in the lower right. Finally, when the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph.

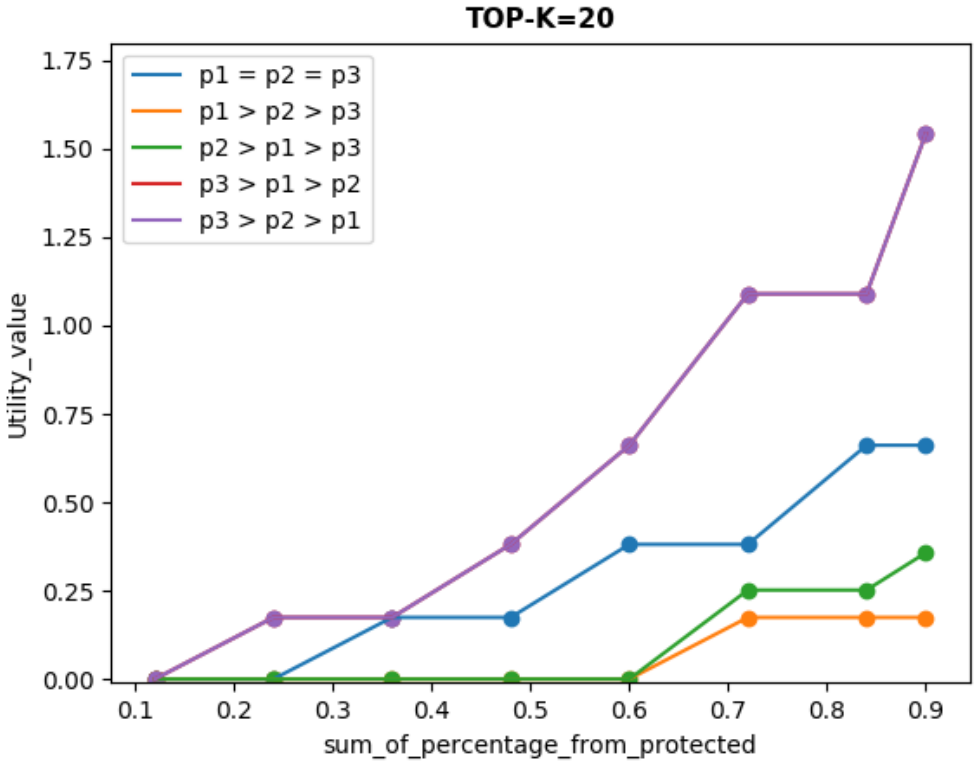


In this metric we should mention that yellow and green line accept exactly same values and as a result yellow line is not obvious in the graph. In addition, purple line accepts almost same values with red line. As a result, the red line appears slightly in the lower right. When the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph. Finally, we can note that the biggest difference between initial and final ranking, as expected, is shown by purple line.



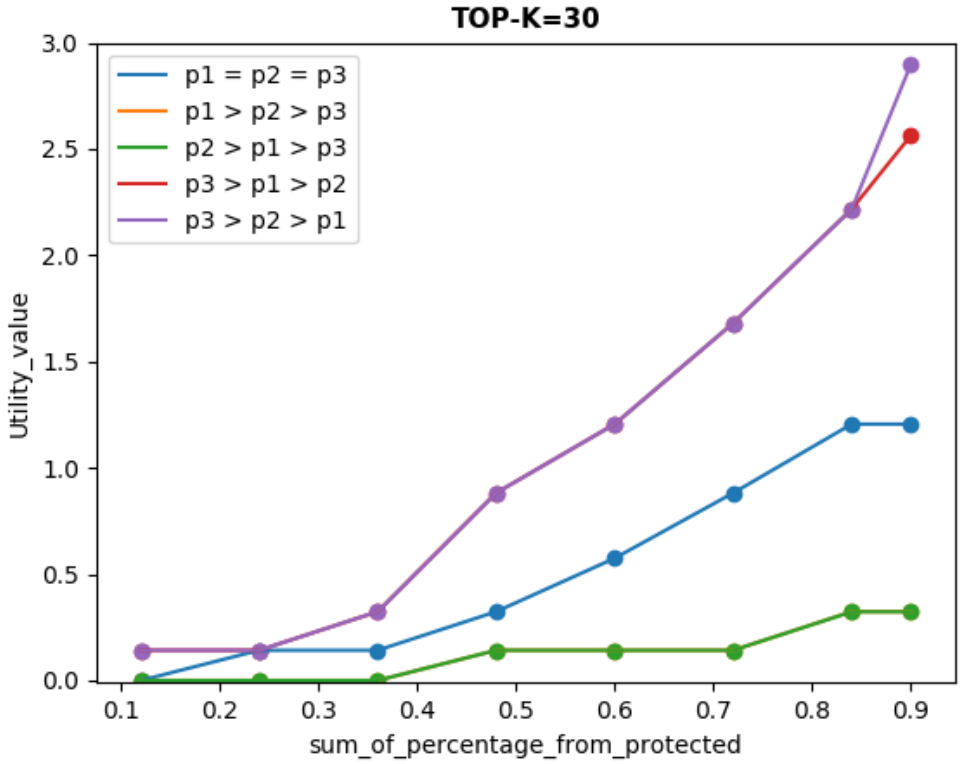
In this metric we should mention that yellow and green line accept exactly same values and as a result yellow line is not obvious in the graph. In addition, purple line accepts almost same values with red line. As a result, the red line appears slightly in the lower right. When the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph. The blue line reflects this observation in the graph. Finally, we can note that the biggest difference between initial and final ranking, as expected, is shown by purple line.

Utility metric:

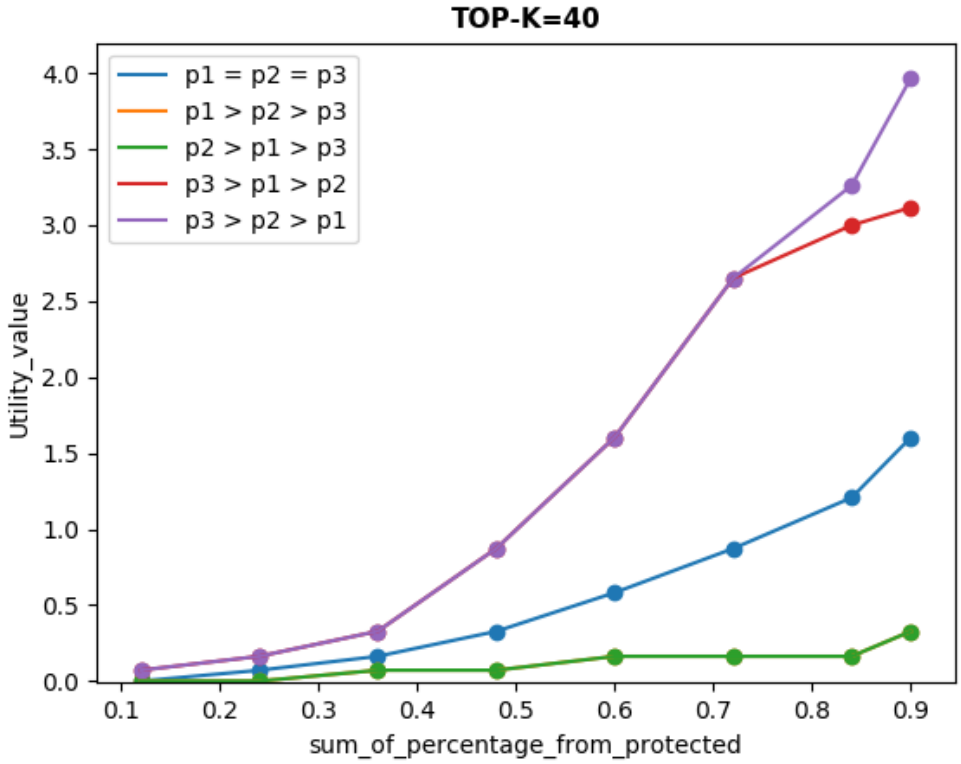


Based on this metric we can arise the following conclusions:

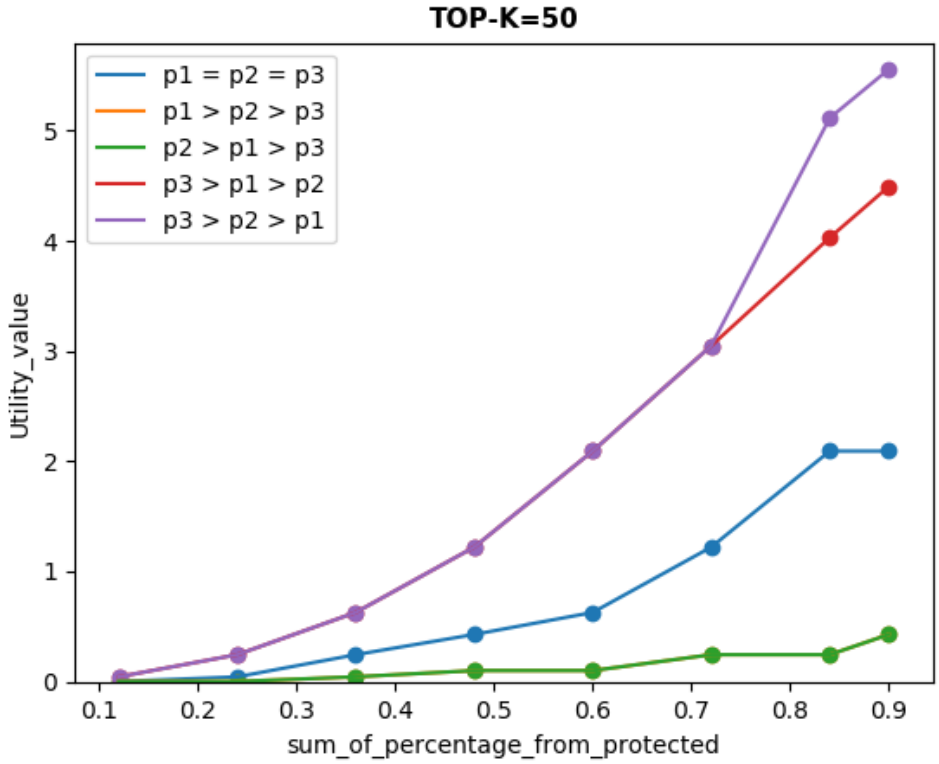
- When p1 has the highest value and p2 the second-highest then the resulting ranking is similar to the original one. As we can see the orange line takes small utility values.
- When p3 has the highest value and p2 the second-highest then the resulting ranking is quite different from the original one. As we can see the purple line takes the highest utility values.
- When the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph.
- Finally, we should mention that red line (p3>p1>p2) and purple line (p3>p2>p1) have exactly the same results, so the red line is not appearing in the graph.



In this metric we should mention that yellow and green line accept exactly same values and as a result yellow line is not obvious in the graph. In addition, purple line accepts almost same values with red line. As a result, the red line appears slightly in the lower right. When the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph. The blue line reflects this observation in the graph. Finally, we can note that the biggest difference between initial and final ranking, as expected, is shown by purple line.



In this metric we should mention that yellow and green line accept exactly same values and as a result yellow line is not obvious in the graph. In addition, purple line accepts almost same values with red line. As a result, the red line appears slightly in the lower right. When the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph. The blue line reflects this observation in the graph. Finally, we can note that the biggest difference between initial and final ranking, as expected, is shown by purple line.



In this metric we should mention that yellow and green line accept exactly same values and as a result yellow line is not obvious in the graph. In addition, purple line accepts almost same values with red line. As a result, the red line appears slightly in the lower right. When the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph. The blue line reflects this observation in the graph. Finally, we can note that the biggest difference between initial and final ranking, as expected, is shown by purple line.

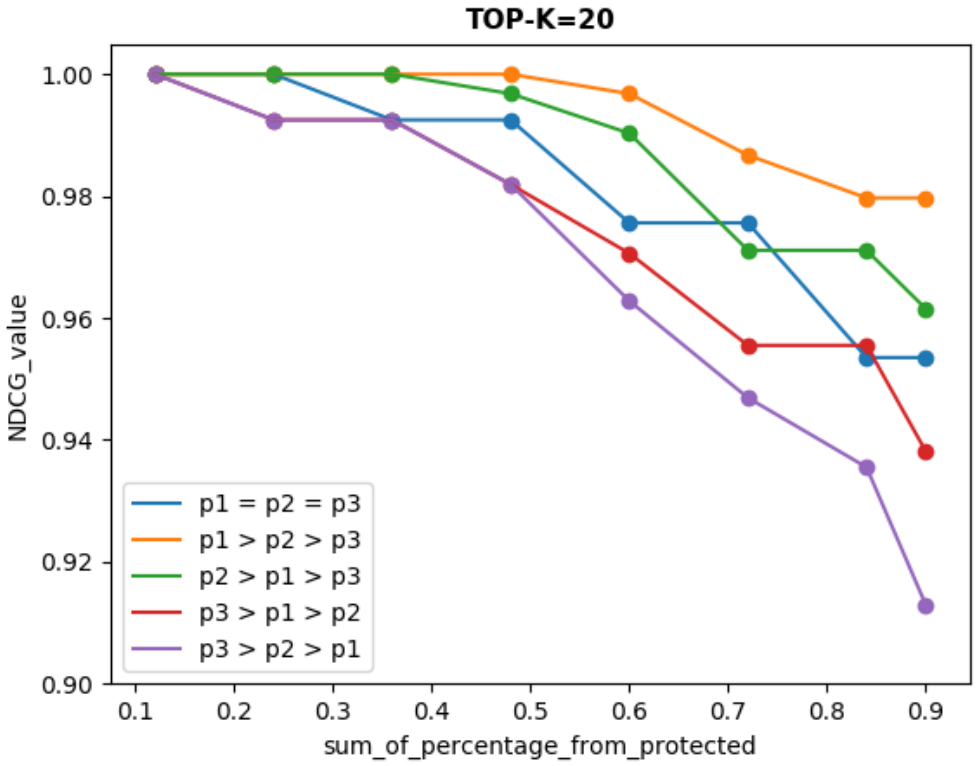
4 groups database (variation II): Test for our database (Predict whether income exceeds $50K/year based on census data) with NDCG and UTILITY metrics. Below we quote the list of tests we ran in order to test how the extension of FA*IR behaves. We have one nonprotected category (race = White) and three protected categories (race = Asian-Pac-Islander, race = Black, race = American-Indian-Eskimo).

Whites = 75 (the nonprotected category) ||

Black = 200 (are illustrated by the symbol p1 in the graph below) ||

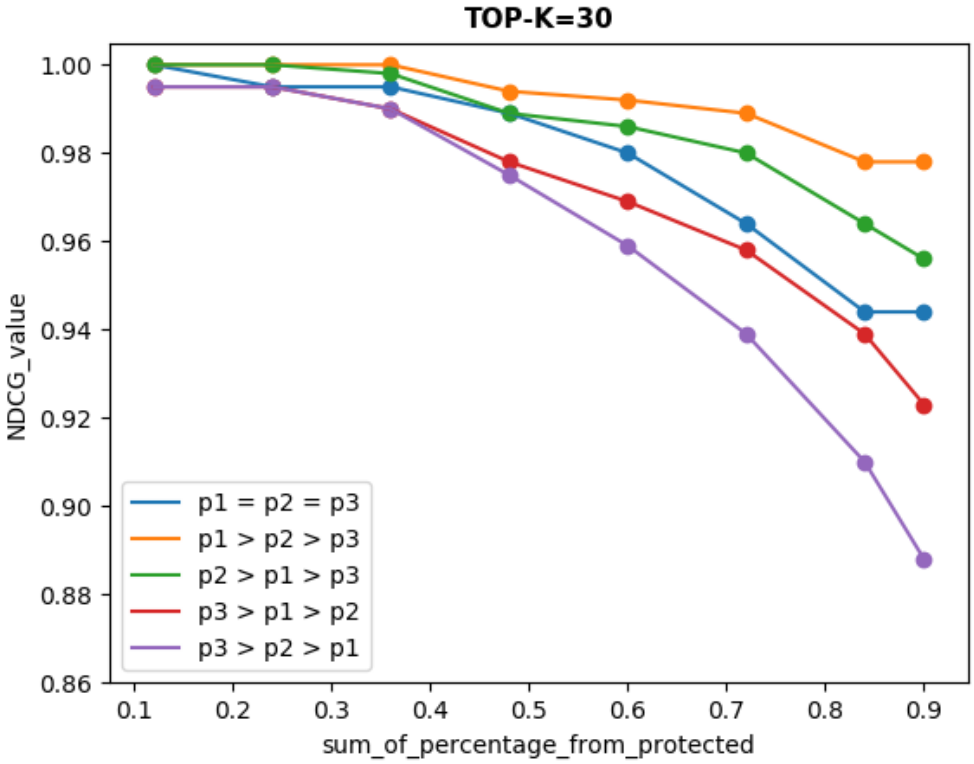
Asian-Pac-Islander = 100 (are illustrated by the symbol p2 in the graph below) || American-Indian-Eskimo= 25 (are illustrated by the symbol p3 in the graph below)

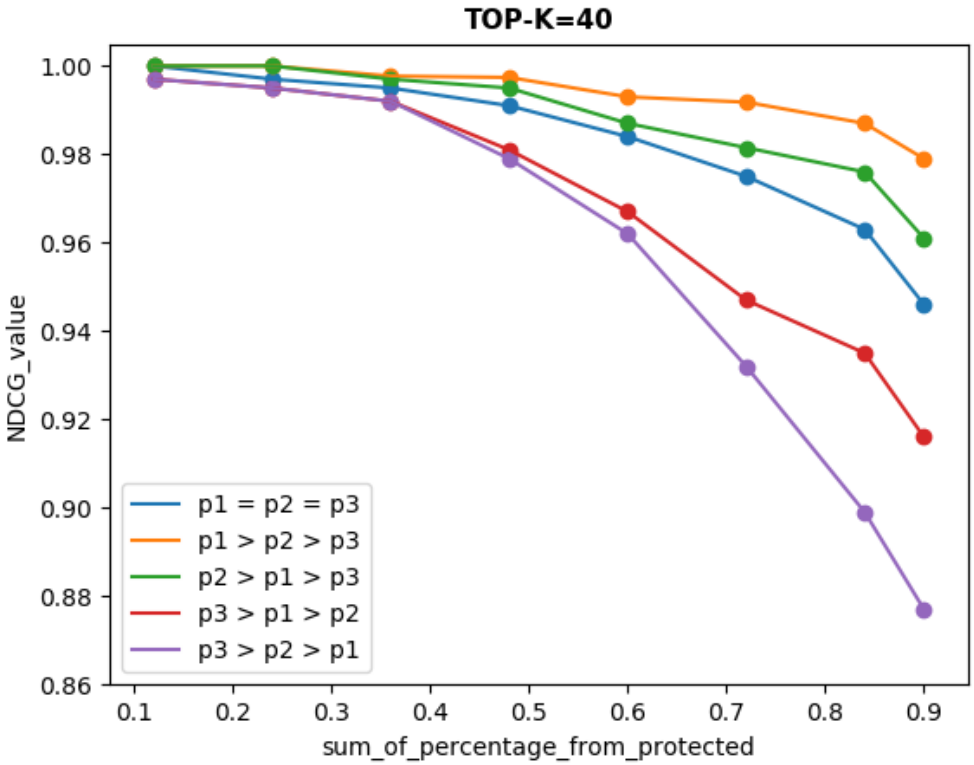
NDCG metric:

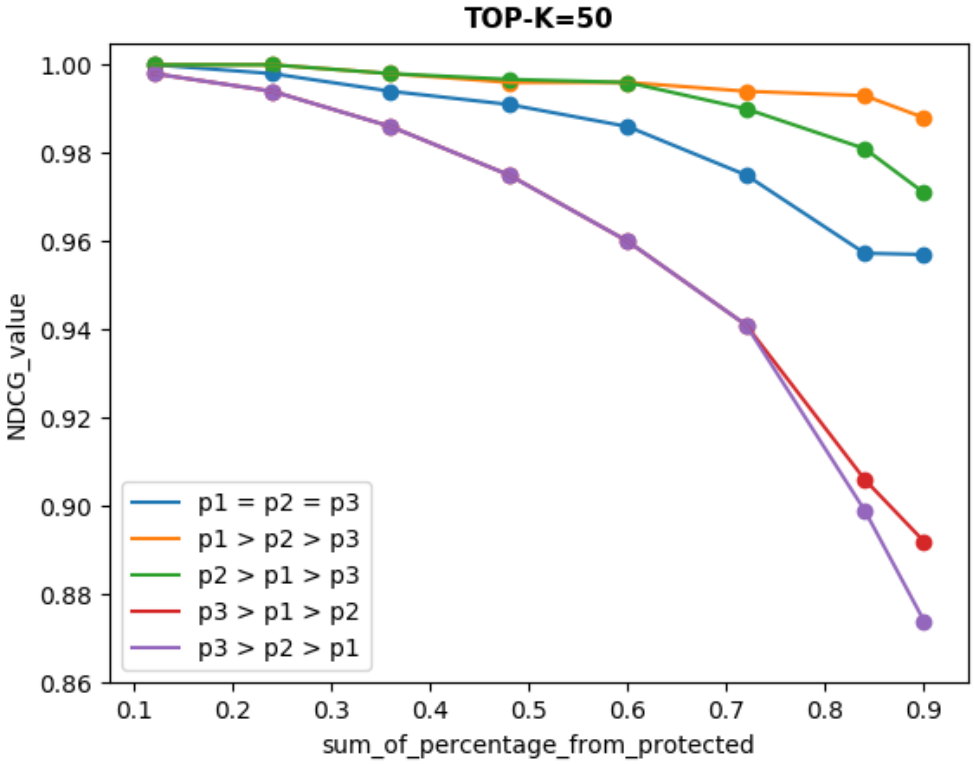


Based on this metric we can arise the following conclusions:

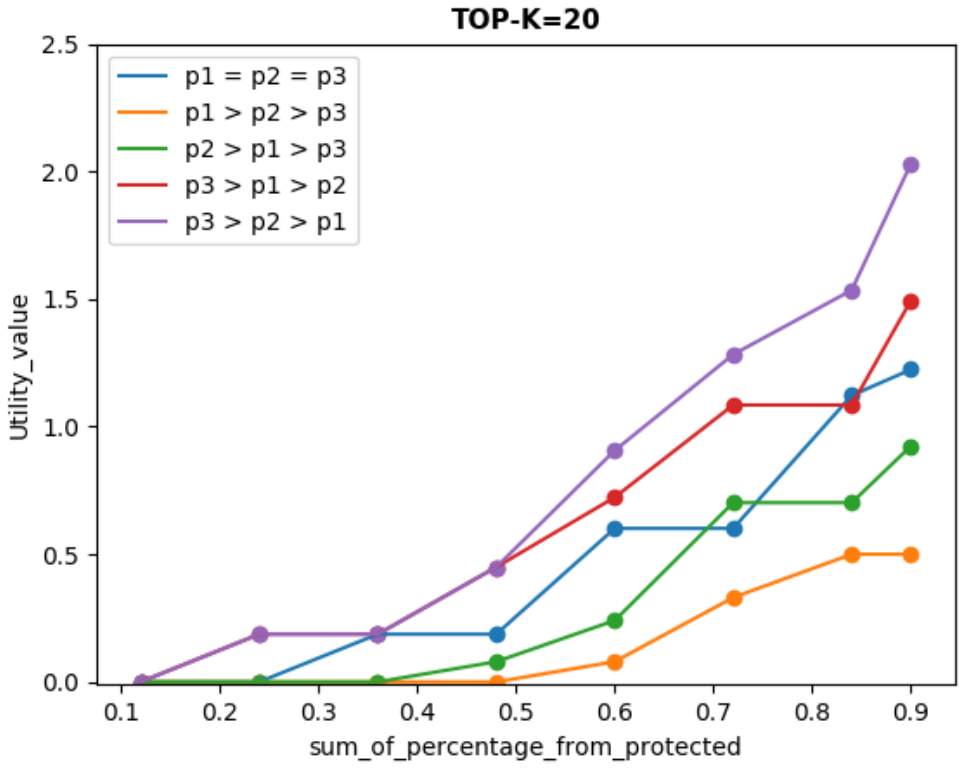
- When p1 has the highest value and p2 the second-highest then the resulting ranking is similar to the original one.
- When p3 has the highest value and p2 the second-highest then the resulting ranking is quite different from the original one.
- Finally, we should mention that red line (p3>p1>p2) and purple line (p3>p2>p1) have exactly the same results, so the red line is not appearing in the graph.


TOP-K=30
NDCG_value
sum_of_percentage_from_protected
p1 = p2 = p3
p1 > p2 > p3
p2 > p1 > p3
p3 > p1 > p2
p3 > p2 > p1
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


TOP-K=40
NDCG_value
sum_of_percentage_from_protected
p1 = p2 = p3
p1 > p2 > p3
p2 > p1 > p3
p3 > p1 > p2
p3 > p2 > p1
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

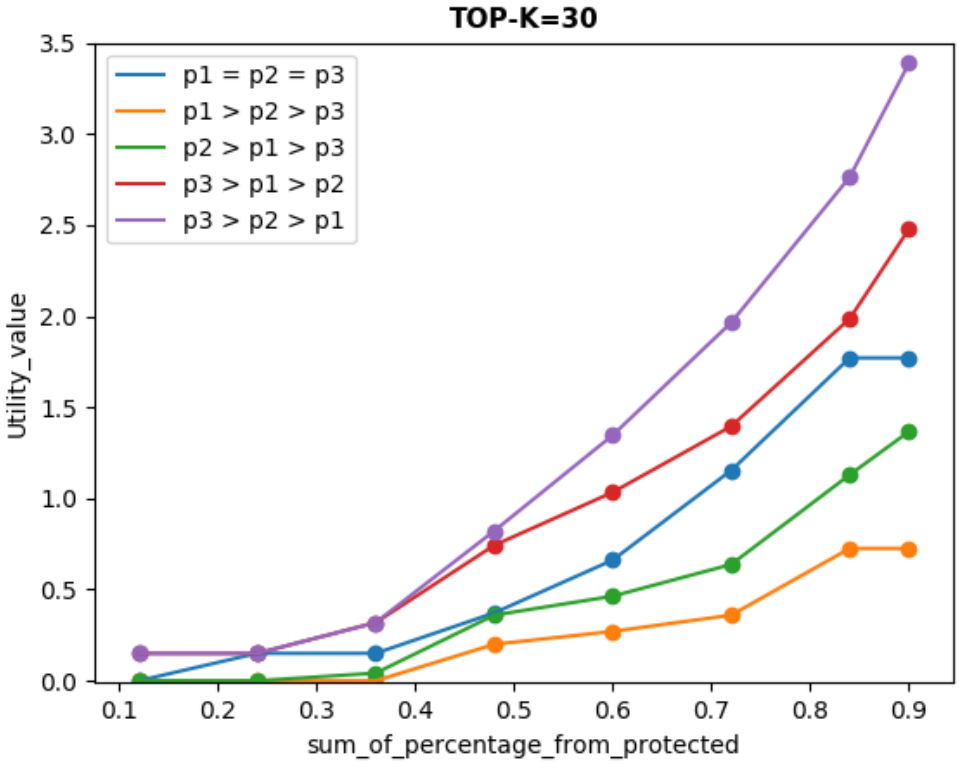

TOP-K=50
NDCG_value
sum_of_percentage_from_protected
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
p1 = p2 = p3
p1 > p2 > p3
p2 > p1 > p3
p3 > p1 > p2
p3 > p2 > p1

Utility metric:

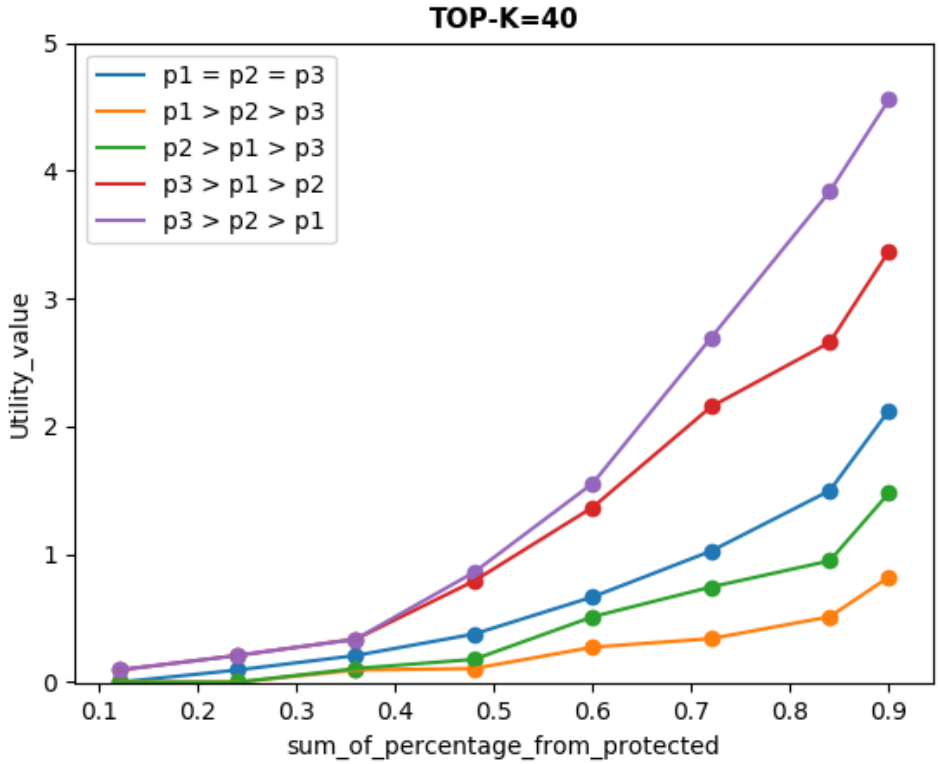


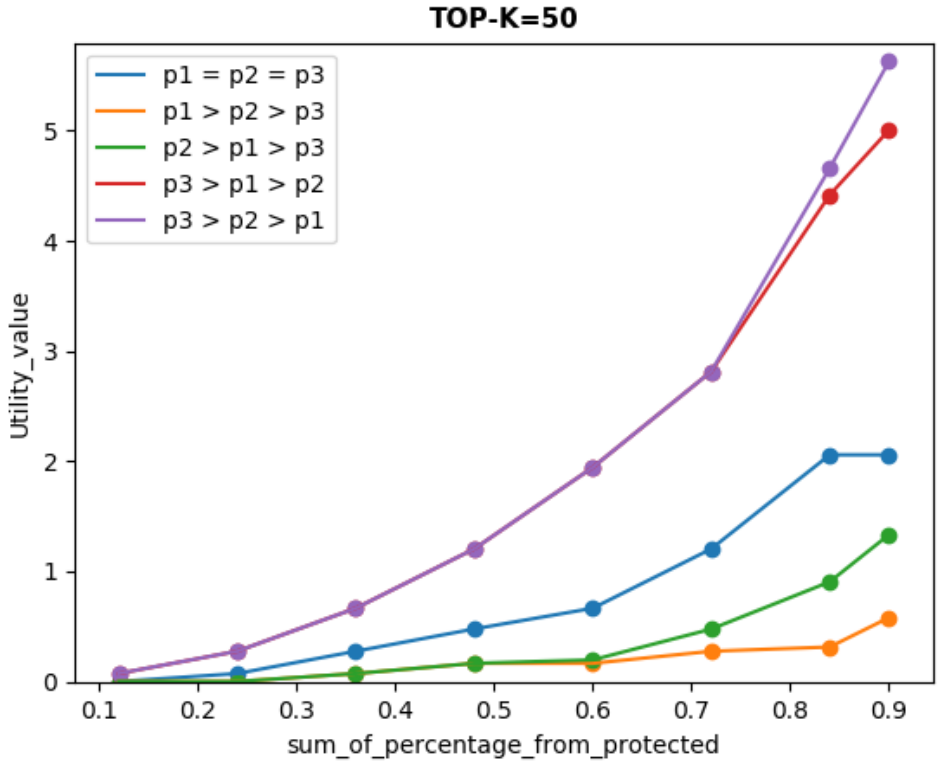
Based on this metric we can arise the following conclusions:

- When p1 has the highest value and p2 the second-highest then the resulting ranking is similar to the original one. As we can see the orange line takes small utility values.
- When p3 has the highest value and p2 the second-highest then the resulting ranking is quite different from the original one. As we can see the purple line takes the highest utility values.
- When the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph.



In this metric, by observing the blue line, we can see that when the sum of percentages P is 0.8 and 0.9, they get the same utility value. This is happening because in top 30 positions there are more data of the protected groups than desired. That's why no change is introduced in the utility value.





In this metric, purple line accepts almost same values with red line. As a result, the red line appears slightly in the upper right.

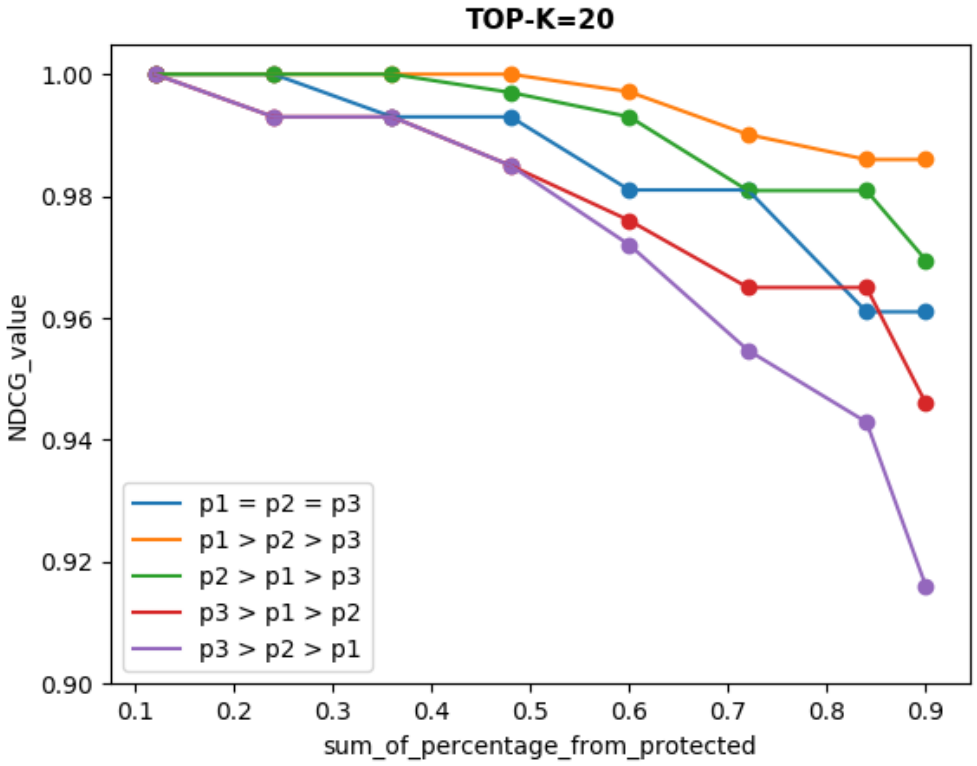
4 groups database (variation III): Test for our new database (Predict whether income exceeds $50K/year based on census data) with NDCG and UTILITY metrics. Below we quote the list of tests we ran in order to test how the extension of FA*IR behaves. We have one nonprotected category (race = White) and three protected categories (race = Asian-Pac-Islander, race = Black, race = American-Indian-Eskimo).

Whites = 150 (the nonprotected category) ||

Black = 90 (are illustrated by the symbol p1 in the graph below) ||

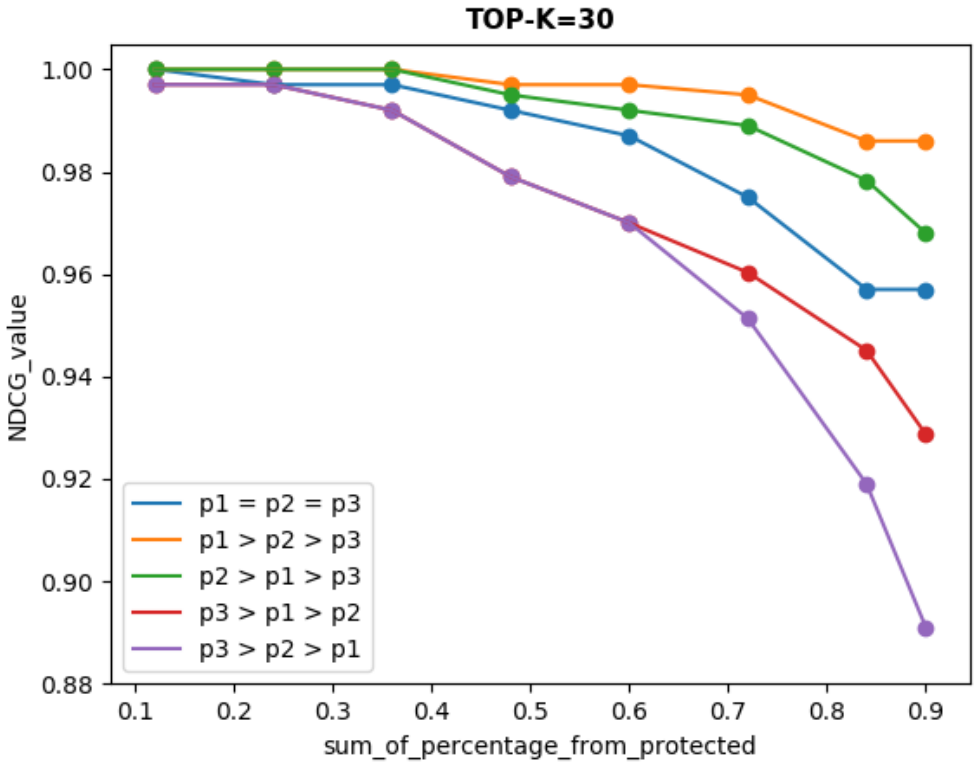
Asian-Pac-Islander = 130 (are illustrated by the symbol p2 in the graph below) || American-Indian-Eskimo= 30 (are illustrated by the symbol p3 in the graph below)

NDCG metric:



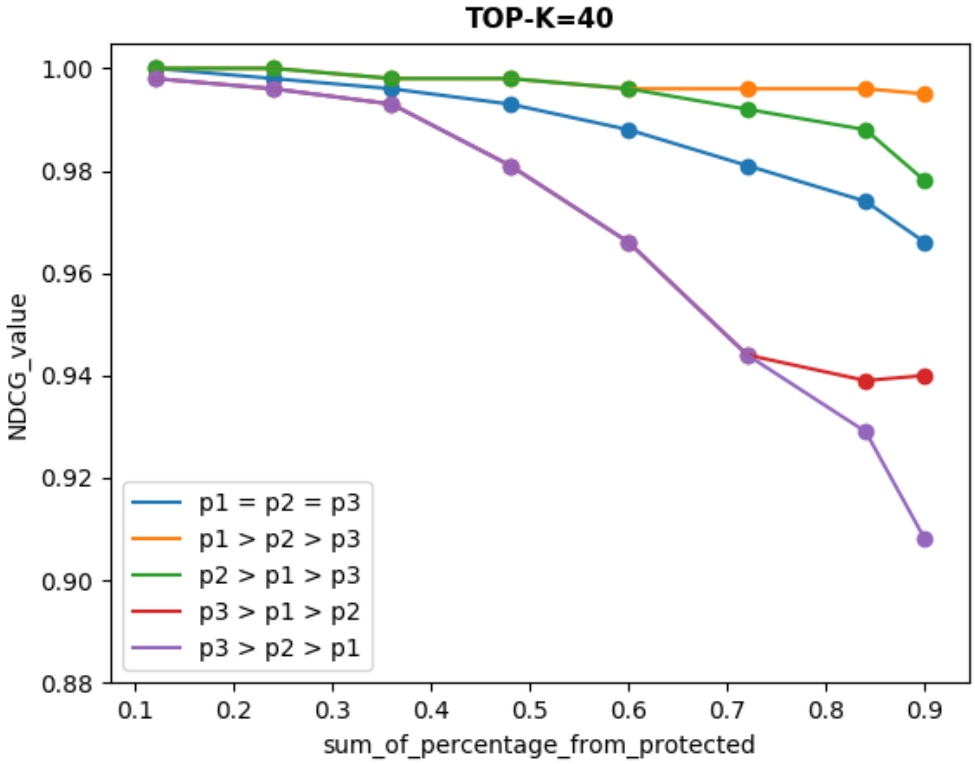
Based on this metric we can arise the following conclusions:

- When p1 has the highest value and p2 the second-highest then the resulting ranking is similar to the original one.
- When p3 has the highest value and p2 the second-highest then the resulting ranking is quite different from the original one.

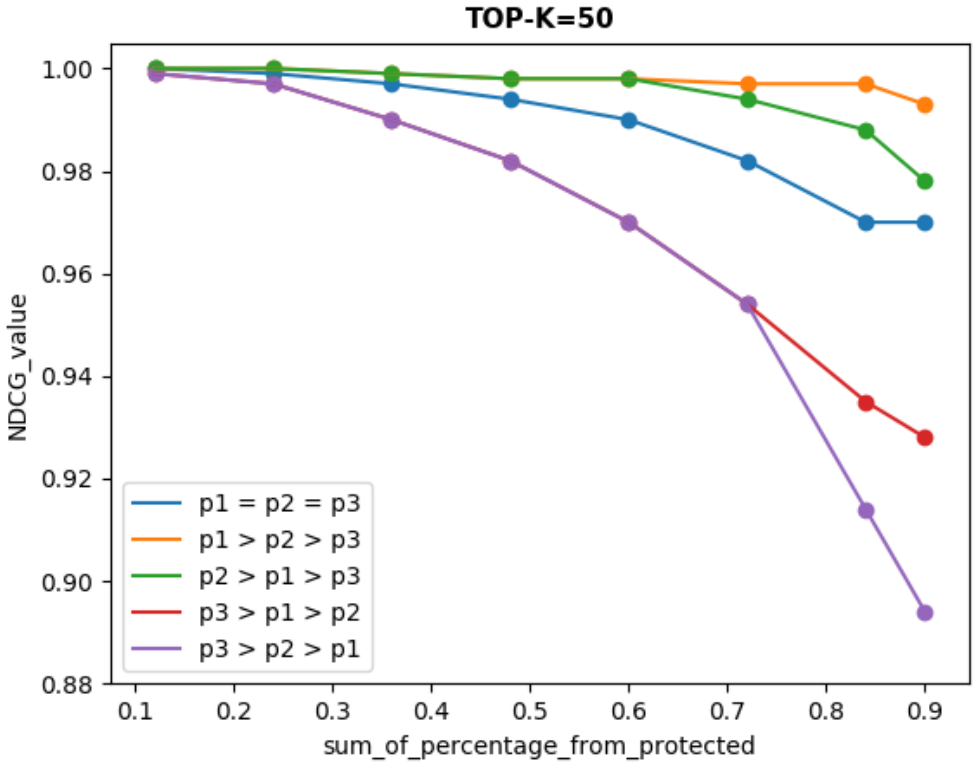


Based on this metric we can arise the following conclusions:

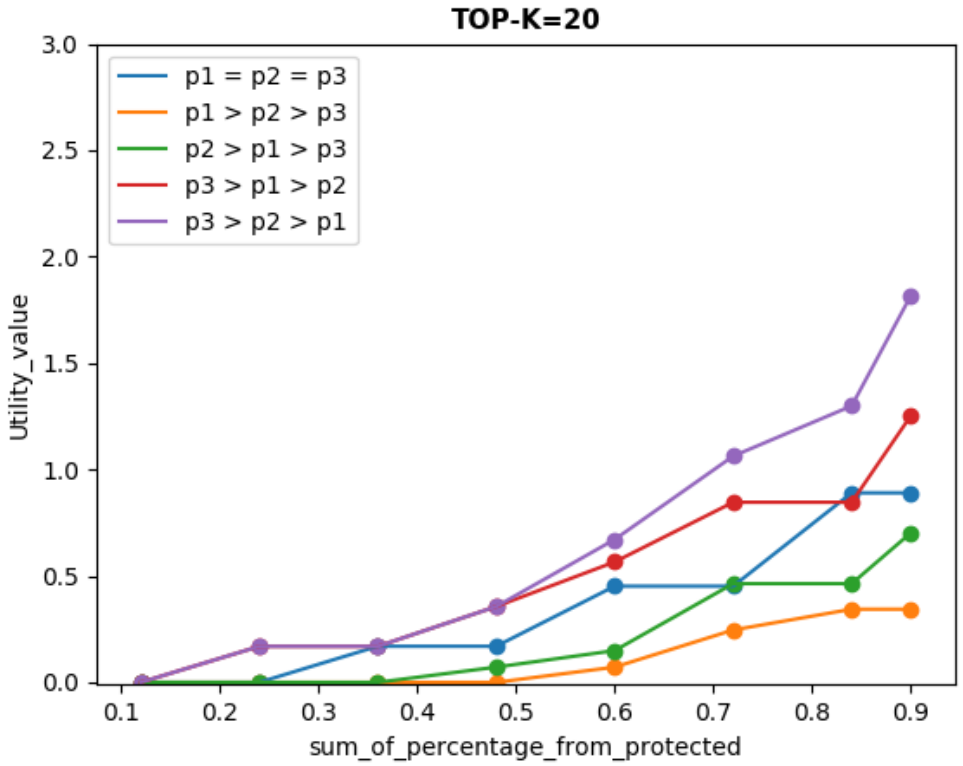
- When p1 has the highest value and p2 the second-highest then the resulting ranking is similar to the original one.
- When p3 has the highest value and p2 the second-highest then the resulting ranking is quite different from the original one.
- When all proportions have the same value (p1 = p2 = p3) then we notice that the final ranking is in the middle. In other words, as much as it differs from the initial classification, it also differs from the classification with values of p3 > p2 > p1.



In this metric, purple line accepts almost same values with red line. As a result, the red line appears slightly in the lower right.

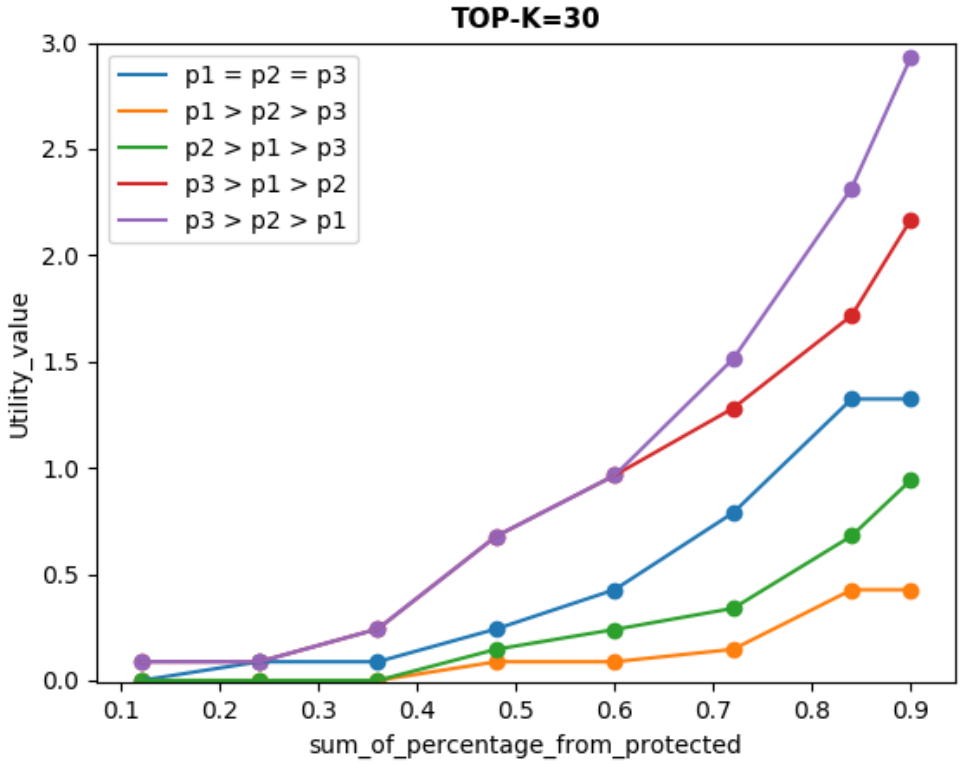


Utility metric:

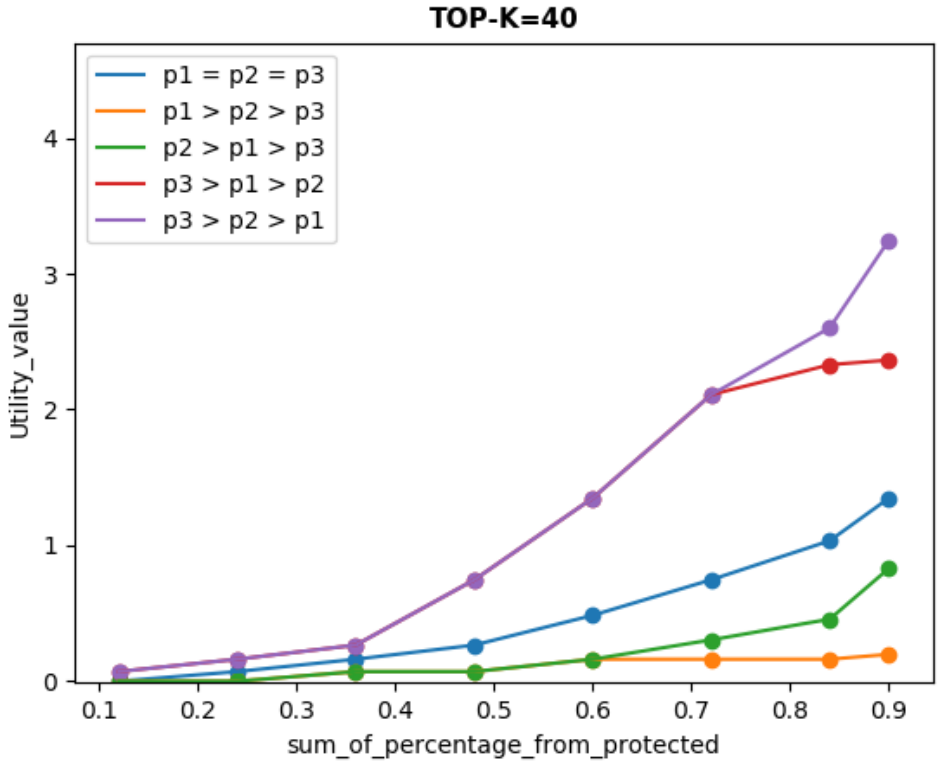


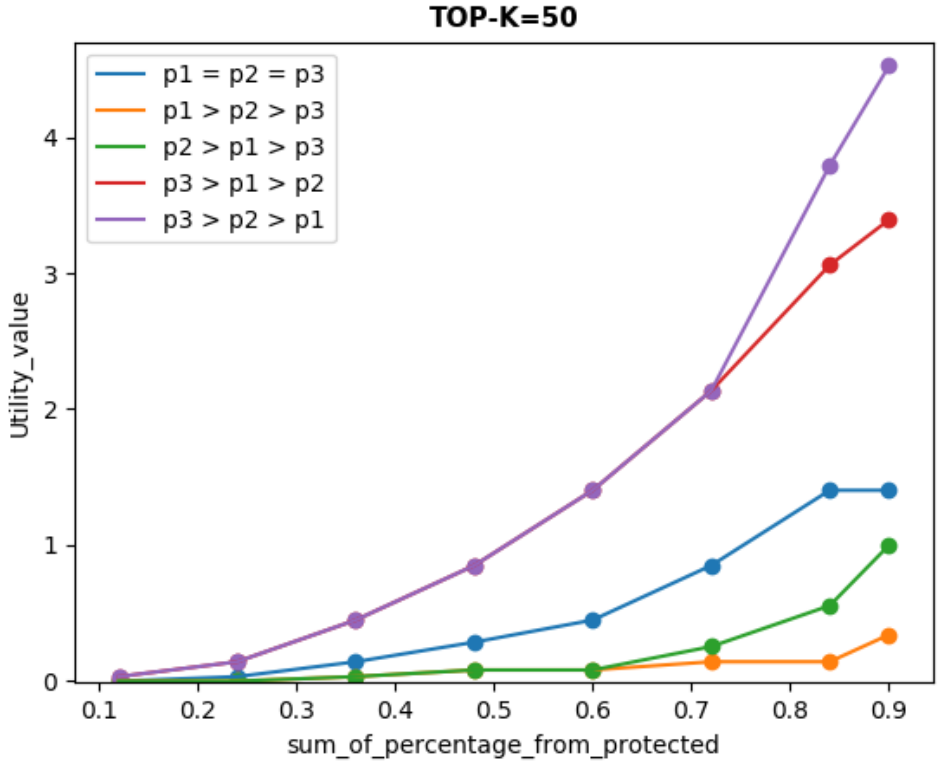
Based on this metric we can arise the following conclusions:

- When p1 has the highest value and p2 the second-highest then the resulting ranking is similar to the original one. As we can see the orange line takes small utility values.
- When p3 has the highest value and p2 the second-highest then the resulting ranking is quite different from the original one. As we can see the purple line takes the highest utility values.
- When the three protected categories have the same value, we notice a medium change of the initial ranking. The blue line reflects this observation in the graph.
- Moreover, the green line is quite similar with the orange line.



As we can observe the results are similar to the top-20 graph with the only difference being that the final rankings are starting to differ more and more. This can be easily understood by looking at the gaps created between the line segments, as now the differences between the final rankings are more obvious.


TOP-K=40
p1 = p2 = p3
p1 > p2 > p3
p2 > p1 > p3
p3 > p1 > p2
p3 > p2 > p1
Utility_value
sum_of_percentage_from_protected
0
1
2
3
4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


TOP-K=50
p1 = p2 = p3
p1 > p2 > p3
p2 > p1 > p3
p3 > p1 > p2
p3 > p2 > p1
Utility_value
sum_of_percentage_from_protected
0
1
2
3
4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Chapter 5. Epilogue

5.1 Summary and conclusions

5.1.1 Useful information that came up while testing DELTR

Researchers design DELTR as a list-wise learning-to-rank approach that provides an in-processing approach to fairness-aware rankings. They further show that if we simply ignore protected attributes, which is a naïve attempt to overcome discriminatory models, it can yield the best results in some cases, while in others it is among the worst results both in terms of performance and fairness. This makes it difficult to identify when or when not to include the protected attribute in the training process.

We have approved that DELTR performs well in terms of fairness and relevance in all tested scenarios. These experiments guide us some interesting insights:

- it is difficult to produce fair training data because discrimination is baked into all attributes. Based on researches, when we focus only on one feature it is quite possible to produce an unfair ranking. To limit discrimination, we should take into account more features in the training session, helping with this way the algorithm to be fairer.
- re-ordering items in a “fair” way can lead to significant performance decline and even reverse discrimination. This conclusion emerged from the observation of the final ranking when the parameter gamma was given high values. Particularly, the protected group possesses the highest positions in the final ranking, which maybe leads us to the conclusion that this ranking is unfair.
- Determining the appropriate value for the variable gamma is a fairly complicated process as it may take several hours for the appropriate value to be found. This happens because there is no methodology for generating the variable gamma, since the variable is assigned by the user.

- The only information we have about the variable gamma is that if we prefer larger γ value we express a preference for solutions that reduce disparate exposure for the protected group and if we prefer smaller γ value we express a preference for solutions that reduce the differences between the training data and the output of the ranking algorithm. As we can observe from our own basis, and from the results of the authors we can easily understand that gamma can obtain sufficiently high values, which means that this variable has a huge range of values.
- In addition, another observation that came up from our measurements showed that the variable gamma may behave differently, namely perhaps giving a different and unexpected classification even with slight differences in its values.
- It is quite difficult to understand when we should include the protected attributes in training process.

In conclusion, DELTR makes no assumptions on whether unfairness against a protected group is present in the training data. Additionally, we show that the current understanding of a necessary trade-off between relevance and fairness can be sometimes misleading. Only if we seek to enhance equality of opportunity, we have to trade performance against fairness. In the case of non-discrimination, optimizing for fairness as equal exposure will increase relevance.

5.1.2 Useful information that came up while testing FA*IR

Conclusions that came up will testing FA*IR:

- First, we observe that in general changes in utility with respect to the color-blind (a ranking which ignores group and individual fairness) ranking are minor, as the utility is dominated by the top positions, which do not change dramatically.
- Second, FA*IR achieves higher or equal ordering utility in all conditions. This is also reflected in the rank loss of the most unfairly treated candidate included in the ranking (i.e., the candidate that achieves the maximum ordering utility loss).
- Third, FA*IR is way faster than the DELTR algorithm.
- Fourth, FA*IR, on the contrary with DELTR can work properly in any given database with two restrictions. First the restriction of being only two individuals groups and second the requirement that the variable p ranges from 0.02 to 0.98.

The method which has been presented can generate a ranking with guaranteed ranked group fairness, and as we have observed, it does not introduce a large utility loss. We also do not assume that the distributions of qualifications in the protected and non-

protected groups have a similar shape. More importantly, we can directly control through a parameter p the trade-off between fairness and utility.

5.2 Future extensions

There are many ideas for future expansion of the FA*IR algorithm and it would be very interesting to deal with their implementation. After demonstrating the FA*IR function with four groups, we propose to convert the code into a dynamic one for future work so that there are no restrictions on the groups that the database should contain. Through it we will break down any restrictions that existed so far regarding the databases that FA*IR could work on.

As for the DELTR algorithm, we would also recommend applying additional metrics to better understand how the algorithm works and to be able to better determine the values that are most appropriate for each variable gamma.

Chapter 6. References

[AGHL09] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, Samuel Leong. Diversifying Search Results. WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining. pp 5–14, February 2009.

[ALMK16] Julia Angwin, Jeff Larson, Surya Mattu, Lauren Kirchner. Machine Bias. ProPublica, May 2016. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[Binn18] Reuben Binns. Fairness in Machine Learning: Lessons from Political Philosophy. Proceedings of the First Conference on Fairness, Accountability and Transparency, vol 81, pp.149-159, Feb 2018.

[BSR+05] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, Greg Hullender. Learning to Rank using Gradient Descent. ICML '05: Proceedings of the Twenty-Second international conference on Machine learning. Pp 89-96, August 2005.

[CDKV16] L. Elisa Celis, Amit Deshpande, Tarun Kathuria, Nisheeth K. Vishnoi. How to be Fair and Diverse?. Pp 5, October 2016.

[CXL+06] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, Hsiaowuen Hon. Adapting ranking SVM to document retrieval. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Pp 186 – 193, August 2006.

[GHBO99] Thore Graepel , Ralf Herbrich , Peter Bollmann-sdorra , Klaus Obermayer. Classification on Pairwise Proximity Data. Proceedings of the 1998 conference on Advances in neural information processing systems II, Vol 11, Pp 438–444, July 1999.

[Joac02] Thorsten Joachims. Optimizing search engines using clickthrough data. Proceedings of the Eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 133–142, July 2002.

[ZBC+17] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. FA*IR: A fair top-k ranking algorithm. In Proc. of the 2017 ACM on Conference on Information and Knowledge Management. ACM, pp 1569-1578, January 2017.

[ZeDC19] Meike Zehlike, Gina-Theresa Diehn, Carlos Castillo. Reducing Disparate Exposure in Ranking: A Learning To Rank Approach. pp 11, May 2019.

[ZQL+07] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, Hang Li. Learning to Rank: From Pairwise Approach to Listwise Approach. Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, pp 129-136, January 2007.

[ZSCK19] Meike Zehlike, Tom Sühr, Carlos Castillo, Ivan Kitanovski. FairSearch: A Tool For Fairness in Ranked Search Results. Pp 4, May 2019.

Chapter 7.

7.1 Annex

Below we describe the most import methods that we insert to extend FA*IR functionality.

Function name: **splitCategories**

Input: All the documents from our database.

Output: An array list in which will be placed the protected documents, an array list in which will be placed the nonprotected documents.

Description: Splits the documents into protected and nonprotected.

Function name: **setProtected**

Input: All the documents from our database and the category of each document.

Output: the category of each document with Boolean values.

Description: Shows me which documents belong to the protected group and sets them as true. Also, sets the rest documents (the other two protected categories and the nonprotected) as false.

Function name: **fairTopK**

Input: All the documents from our database, the category which every document belongs to, the percentage of protected documents, the number of top elements in the ranking, a value alpha which shows as if the ranking is fairly representing the protected group.

Output: returns top-k results in ranking concerning to variable k.

Description: Calculated Fair top-k results - documents.

function name: **resetScore**

Input: an array list with the initial scores of the ranking and an array list with the scores of the ranking after fair algorithm run.

Output: an array list that combines the initial scores (before FA*IR run) and the final positions (after FA*IR run) of documents.

Description: After the run of the FA*IR algorithm we have a new rank and new scores of documents. Specifically, documents have new scores that FA*IR gave to them. With this method we match documents with their original scores (the score they have in the original ranking) and their final position in the ranking.

function name: **mergeTopK**

Input: an array list which contains the top k results of "black" as protected group and the other three categories as nonprotected, an array list which contains the top k results of "Asian-Pac-Islander" as protected group and the other three categories as nonprotected, an array list which contains the top k results of "Amer-Indian-Eskimo" as protected group and the other three categories as nonprotected, three variables called max elements that contains the minimum percentage of protected documents for each protected category, the number of top elements in the ranking.

Output: the final ranking.

Description: This method joins the results of the three rankings that have been created before and create the final fair ranking.

function name: **insertNotProtected**

Input: the final ranking.

Output: a nonprotected document.

Description: This method inserts a nonprotected document in the ranking if it does not already exist and puts it in the appropriate position based on its score.

function name: **isInList**

Input: the final ranking.

Output: returns true if the document already exists in the ranking or false if the document does not exist in the ranking.

Description: This method checks if a document is already on the list.

Function name: **insertProtected**

Input: the final ranking, an array list which contains the top k results of one protected group and the other three categories as nonprotected, a variable called max elements that contains the minimum percentage of protected documents for the protected category.

Output: a protected document.

Description: This method inserts the protected documents of one list in the final ranking list. Here we should mention that we take only the minimum number of p1, p2, p3 to make sure that the sum of the protected categories will not exceed k value.

function name: **clearList**

Input: an array list which contains the top k results of the protected group and the other three categories as nonprotected.

Output: an array list which does not contains any protected documents.

Description: This method just deletes the protected documents of a list.

function name: **nextNotProtected**

Input: an array list which contains the top k results of the protected group and the other three categories as nonprotected.

Output: next nonprotected document.

Description: This method returns the next nonprotected document and simultaneously remove it from the list.