Ca' Foscari University of Venice

Department of Environmental Sciences, Informatics and Statistics

Computer Science

New fairness measure applied to a learning to rank method

Supervisor:
Chiar.mo Prof. Orlando

Co-Supervisor:
Dr. Federico Marcuzzi

Graduand:
Tintari Nicanor
866510

Academic Year 2023/2024

Contents

List of Figures

Abstract

Nowadays, the need to organize data and enhance its accessibility in a systematic way is ubiquitous. However, data is not free from bias and preconceptions, and the inherent data bias may negatively influence data-driven algorithms. Consequently, the information represented may tend to favour certain groups over others, thereby perpetuating discrimination against so-called protected groups. This is also particularly evident in the field of learning to rank (LtR), in which LtR algorithms are trained to rank a set of items represented as feature vectors.
Numerous studies have been conducted in recent years on fairness management for machine learning algorithms, with the objective of reducing the effects of data biases on the trained models. In this thesis, we focus on the relationship between LtR and fairness. We modify an LtR framework, LambdaMART, whose original goal is to optimize NDCG@k, by adding a group-based fairness measure to optimize, called Normalised Discounted Difference (rND). Following an initial study focusing on LtR and fairness, various methodologies for combining the two metrics and their application on two real datasets will be proposed and evaluated.

Chapter 1

Introduction

Nowadays, machine leaning, particularly its application in the **Learning to Rank (LtR)** field, is more critical in everyone's daily use. This extensive usage is significant because LtR algorithms are used in search engines and recommended system used by the platforms that are used daily. Since they are so used, it is important to deal with a particular problem that may be present in these algorithms, i.e., equity and impartiality, which can be defined as **Fairness**. In particular, in LtR, resolving the problem of Fairness means resolving all the problems that can be created by ranking making discriminant choices. This discrimination can be created because the data or the algorithm used is not fair, and this creates a situation where a specific group, described by a specific attribute like 'Sex' or 'Age', is considered not protected and then discriminated against. The scope of this work is to create an LtR algorithm that deals with the problems created by unfairness, particularly a model that finds a trade-off to optimise both Effectiveness and Fairness. Regarding Fairness the objective is to achieves statistical parity between the different groups and this means that the groups are treated in the same way regardless how they are defined.
The thesis is divided in different chapters. In the chapter 2, we are going to explain all the basic concepts of learning to rank and Fairness necessary to better understand our problem. Then, in chapter 3, we will explain the approach we adopted and the motivations for the algorithm's and we will show some experiments done to demonstrate its validity.

Chapter 2

State of the art

In this chapter, we will explain the thesis's fundamental theory. It is divided into four sections. The first section, 2.1, explains machine learning, its importance, and information retrieval with the importance of the search engine. The second section, 2.2, explains a specific combination of machine learning and information retrieval, and that's learning to rank, with its different implementations. In the third section, 2.3, a specific problem is addressed and explained, i.e. the problem of Fairness, which has become an important aspect to take care of when using machine learning, and the different methods that can be used to resolve it. Finally the last section, 2.4, explains the two dataset that we will use to test our method.

2.1 Machine Learning and Information Retrieval

2.1.1 Machine Learning

Machine learning is a group of techniques and methods that uses data to make decisions or predictions. The decisions that can be made are different and can be applied into different domains with different applications such as recommendation systems, prediction of disease based on patient analysis, fraud detection, image analysis and object detection, and many other. To do all these things, it is necessary to have data to use as a basis for the algorithms.
The data can be of different types based on the algorithm's scope, this means that the input can be an image, some text, numbers representing some kind of information and other types, the important is that is transformed in number format so the algorithm can "understand" it. Usually the data is processed; first cleaned [20] (so the duplicates, the missing values, and the errors are removed), and then normalised (this means that it is scaled to a standard format, which makes it easier to work on it). Obviously, another important part is the algorithm. Some examples are regression or classification tree, clustering methods or neural networks, and it is essential to choose the right one because each has its advantages and disadvantages for each kind of problem that it treats.
The data is usually given as three sets: the *training set*, which contains the data that the algorithm will use to "learn" the patterns and then generate the output; the *validation set*, which is used as a first testing platform where the model is applied to see if the results are good, and if not, then it is adjusted, and last one is the *testing set* which

contains the data that the algorithm will use to test if it learned correctly and then with different measures it is verified its accuracy.
As just said, there are different algorithms but they can by grouped in four big categories [31]:

- **Supervised learning**: This kind of algorithms takes labelled data, given in the format of a pair (x, y) where x is the actual data, and y is the expected output. The output is generated by an unknown function $y = f(x)$. There are different algorithms, such as decision trees, support vectors, or linear regression models. The problems are usually divided into classification, the scope of which predicts belonging to a class, or regression, which predicts a score of how much an object belongs to a particular class

- **Unsupervised learning**: In this case, the data do not have a class, so the input is received only from the description of the data, and then different algorithms can be used to cluster the data, i.e., to group the items that are similar based on functions that describe the distance between the items

- **Semi-supervised learning**: This is a combination of supervised and unsupervised algorithms and uses the advantages of both methods to classify items

- **Reinforcement learning** This is more particular, as in the unsupervised, the data are not labelled, and it is based on an agent, an environment, some policy and the rewards. Here, the idea is that the model, through a process of try and error, it starts from a specific condition and iteration after iteration through this process it has the scope to achieve optimal conditions. The basic idea is that the agent, which is the model, makes some choices; based on these choices, there is a specific output. If the output is optimal, there is a reward; otherwise, another choice is made, this process is repeated until the optimal state is achieved. These methods are principally used in financial predictions, games, robotics and other similar fields

2.1.2 Information Retrieval

Information retrieval can be defined as the task of finding documents that are relevant to a user's need [37]. One well known example of an information retrieval system is the search engine, which is also necessary because the large amount of accessible data makes it difficult to access it efficiently [28].
In Figure 2.1, it is showed how a search engine is composed and in the following parts all the components are explained:

- **Crawler** which uses some particular strategies to gather documents from the web

- **Parser** that analyses the documents and creates an index of terms and a hyperlink graph according to the analysis

- **Indexer** that transforms the parser's results into indexes, and this allows to have a quick document search

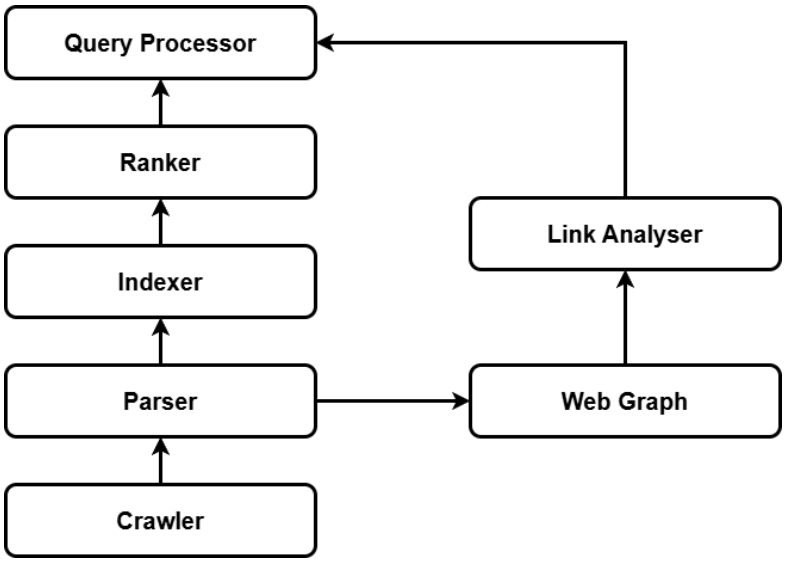

Figure 2.1: Parts of a search engine (Insp. by [28])

- **The link analyzer** then evaluates the graph the parser generates to assess each page's significance. It also acts as a ranking feature to help the page to be recrawled
- **The query processor** creates the connection between the user and the search engine. Then, the queries are processed using text-cleaning techniques, such as stemming, lemmatisation, removing stop words, and so forth. This helps index terms that search engines can then interpret
- **Ranker** compares the queries with the indexed documents by taking them as input and can then, thanks to some heuristic formulas, calculate a matching score which it can also be determined by extracting different features for every query-document pair, and then combining them

Considering the importance of the ranker, much attention was given to the methods that do this part of the process.

As just said the part of the ranker is very important and there are different algorithms, that do this task, that were created in the years. The first, and most simple, algorithm that can be used is a **Boolean Retrieval Model** [3]. In this case, each query is expressed as a boolean expression combined by the operators *AND*, *OR* and *NOT*. This model works in the following way: given N documents, each document is treated as a boolean feature, which will answer *True* if a specific word is present and *False* otherwise. The query, as said, is expressed as a boolean expression, and then the model will give all the relevant documents that the expression evaluates as *True*. Even if the model is simple to implement, it has different disadvantages, such as the fact that it is not possible to order the documents, and it is not easy to express the queries as a boolean expression. As can be deduced, the method could be more usable, and new algorithms, such as the **Vector Space Model** [3], were created. This kind of model takes as input, or calculates it in some way, a document's relevance to the given query, so thanks to that it is possible to know the relevance of each document and then rank them. In order

to do so each document is represented by a vector of words describing it and then with different methods is possible to calculate how important each document is to the relative given query. This association between the query and the document is done thanks to techniques such as the cosine distance, which allows to calculate how much the query vector and the document vector are similar. Then, thanks to this distance each document can be ranked. One of this new algorithms is the one that uses the **BM25** [36] function defined as

$$BM25(d_j, q_{1:N}) = \sum_{i=1}^{N} IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k+1)}{TF(q_i, d_j) + k \cdot \left(1 - b + b \cdot \frac{|d_j|}{L}\right)} \tag{2.1}$$

- N is the number of documents
- $TF(q_i, d_j)$ is the count of the number of times the word q_i appears in the document d_j
- $|d_j|$ is the length of document d_j in words
- L is the average document length defined by $L = \sum_i \frac{|d_i|}{N}$
- k and b are tuned by cross-validation
- $IDF(q_i)$ is the inverse frequency of the word q_i given by

$$IDF(q_i) = log \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5}$$

- $DF(q_i)$ is the number of documents that contain the word q_i

BM25 is influenced by how many times a word appears in a document, but it only has to appear in a few documents; otherwise, it is not considered relevant since it is present in almost all of them and this implies that even if the word is very frequent its value to the query it is not so high.

2.2 Learning to rank

In Section 2.1 is explained the importance of the ranker in the search engines and what are the machine learning algorithms, **Learning to Rank (LtR)** is a combination of both. LtR is a model based on scores, so the scope is to output a relevance score for each pair composed by a query and an item and then sort them to give a rank.
In Figure 2.2, is seen the fundamental process of the LtR method which is composed of two phases: the learning phase which takes the training data, formed by n training queries q_i $(i = 1, ..., n)$ and their corresponding items represented by the feature vector $x^{(i)} = \left\{x_j^{(i)}\right\}_{j=1}^{m^{(i)}}$ (where $m^{(i)}$ is the number of documents linked to query q_i), along the score that indicates how relevant the document is for the query [28], and this is defined as the relevance judgement and can be given by:

- *Relevance degree*: in this way, the relevance of a document to a given query is given by some human operators, and this is one of the most used methods given its simplicity due to the fact that is only necessary to watch and confront the given values
- *Pairwise preference*: also here, the preferences are given by humans, but it is between two documents, so it is just which one is more relevant to the given query
- *Total order*: the preference is given as an order of documents based on the query denoted as a particular permutation, and this is the most costly method of the three

One problem that can be present in this kind of algorithm is the absence of judgment in some cases and those then are considered by default as non relevant.
Once that all these data are prepared then a *loss function* h uses them in the training phase to learn to rank the items. After this is done new data is passed to the model, this is the testing phase, and is produced a ranking. Therefore is calculated a score with a specific measure that confronts the relevance judgement given by the algorithm and the true ones. This score is calculated for each query and then is averaged to get a total score.
Different implementations of learning to rank have been made and can be classified into three types: **pointwise**, **pairwise**, and **listwise**. These techniques utilise distinct strategies based on various input, output, hypotheses spaces, and loss functions.

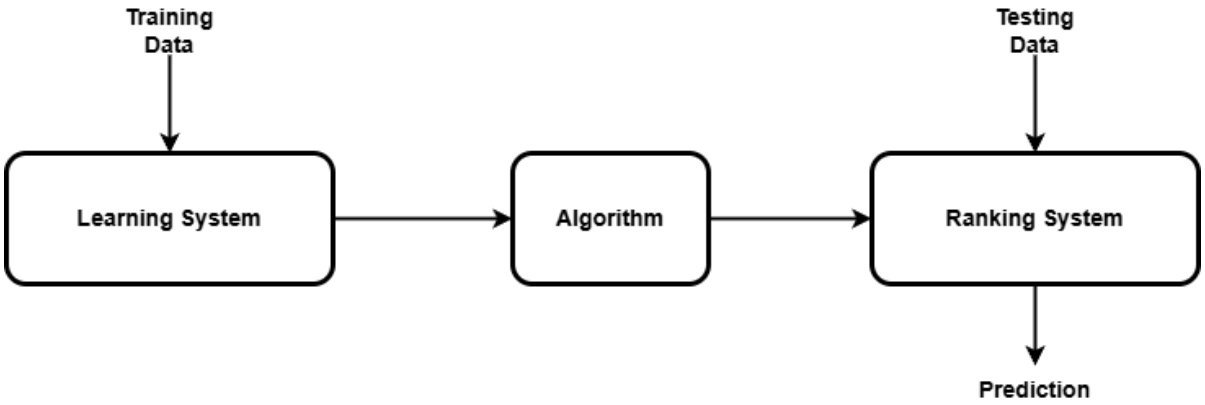

Figure 2.2: Learning to rank frame (Insp. by [28])

2.2.1 Measures Used

As said the ranking produced is measured and there are different types of measures that can be used, in the following section some of them are described:

1. **Mean Average Precision(MAP)**: Before defining MAP, it is necessary to define the *Precision at Position* k, $P@k$, which calculates the fraction of documents that are relevant in top k positions. In case that the judgment is binary, this can be defined as

$$P@k(\pi, v) = \frac{\sum_{t \leq k} I_{\{v_{\pi^{-1}(t)}=1\}}}{k} \tag{2.2}$$

$I_{\{\cdot\}}$ is the indicator function which is 1 if the condition inside the parenthesis is fulfilled. $\pi^{-1}(j)$ is the document ranked at position j of the list π. v is a list of relevance of items and is 1 in position i if the i-th item is relevant otherwise is 0. But this is the average precision only at some position k, then the *Average Precision (AP)* defined for all the items is

$$AP(\pi, v) = \frac{\sum_{k=1}^{m} P@k \cdot I_{\{v_{\pi^{-1}(k)}=1\}}}{m_1} \tag{2.3}$$

m is the total number of documents related to query q, particularly those with label 1, are m_1. Now that AP is defined then MAP [3] is defined as the mean value of AP calculated on all the queries

2. **Discounted Cumulative Gain (DCG)**: DCG [24, 23] has the advantage of handling binary and multi level relevance judgments. It calculates the scores giving more importance to the items placed in the top positions. For a specific query, it can be defined as

$$DCG@T \equiv \sum_{i=1}^{T} \frac{2^{l_i} - 1}{log(1+i)} \tag{2.4}$$

l_i is the relevance label of i-th document; T indicates the truncation level, so in case that the algorithm does not need all the ranking it can stop at a specific level

3. **Normalized Discounted Cumulative Gain (NDCG)**: Another measure is the NDCG [24], which is defined as

$$NDCG@T \equiv \frac{DCG@T}{maxNDCG@T} \tag{2.5}$$

With the denominator that is the maximum $DCG@T$ for the query taken into consideration

The previous measures are all defined in $[0, 1]$ and are position based, so they are discontinuous and non-differentiable. Their particularity is that their values change only when one item is ranked higher than the other.

2.2.2 Pointwise

This method is the most used, that's because each document is scored independently with the ground truth target values. The input space is formed by documents described by a feature vector. The output space is given by the relevance degree of each single document. The true label can be converted from different types of judgment and can be described in terms of relevance degree; some examples are:

- If it is given the relevance degree l_j, the ground truth label for document x_j is defined as $y_j = l_j$
- By determining the pairwise preference $l_{u,v}$, it is possible to obtain the ground truth label by counting the number of times a document outperforms another document

The hypothesis space, which is the set of all hypotheses the algorithm returns, includes functions that use the document's feature vector to predict its relevance degree. The function f, called the *scoring function*, takes this data and creates the results that allow the documents to be ranked.

There are several problems with this method, some of them are the following:

- The input object is just one document, so the interconnections between the document and the order of documents are ignored, and the loss function focuses only on the prediction of the ground truth label
- Multiple documents can be linked to a single query, but this is not considered. If various documents are associated with different queries, the precision of the loss function will diminish
- The loss function may give excessive importance to documents that are unimportant to the user, as it needs to consider their position in the ranking and the function does not do it
- The training set requires the true label

2.2.2.1 Example of algorithms

As said previously, the goal is to create a ranked list of the documents which can be viewed as a regression [19], classification, or ordinal problem.

2.2.2.1.1 Regression

In this scenario, solving the ranking becomes a regression problem. This means that the model has to predict a continuous target variable, and the degree passed is also a continuous variable. An instance of this is the algorithm described in "Subset Ranking Using Regression" [14], which requires a group $x = \{x_j\}_{j=1}^{m}$ related to a training query q and the true labels of the documents given by $Y = \{y_j\}_{j=1}^{m}$. The documents are then ranked by a scoring function f, and the loss function is defined as:

$$L(f; x_j, y_j) = (y_j - f(x_j))^2 \tag{2.6}$$

In Figure 2.3, is possible to observe that zero loss is present only if the scoring function $f(x_j)$ is exactly equal to the label y_j.

2.2.2.1.2 Classification

When viewing the ranking as a classification, the relevance degree is given as a category label. Here, the classification can be divided in binary and multi-class:

- **Binary**: An example is the **SVM-Based Method** [32, 44] that uses documents, $x = \{x_j\}_{j=1}^{m}$, and their binary relevance judgements, $y = \{y_j\}_{j=1}^{m}$, with respect to a query q. If the documents are relevant $y_j = +1$; otherwise, $y_j = -1$. SVM is then

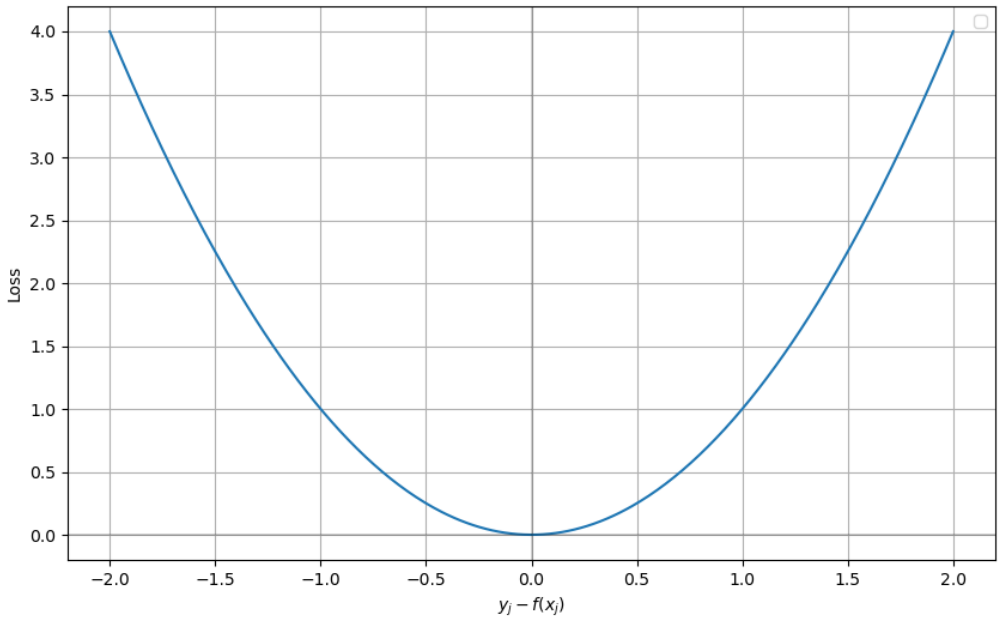

Figure 2.3: Square Loss Function (Insp. by [28])

defined as

$$\begin{aligned} &\min \frac{1}{2}||w||^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{m^{(i)}} \xi_j^{(i)} \\ &s.t.\ w^T x_j^{(i)} \leq -1 + \xi_j^{(i)},\ \text{if } y_j^{(i)} = 0 \\ &w^T x_j^{(i)} \geq 1 - \xi_j^{(i)},\ \text{if } y_j^{(i)} = 1 \\ &\xi_j^{(i)} \geq 0,\ j = 1, ..., m^{(i)}, i = 1, ..., n \end{aligned} \tag{2.7}$$

Where $x_j^{(i)}, y_j^{(i)}$ are the j-th document and its label with respect to a query q_i. Moreover $m^{(i)}$ is the total number of documents associated with query q_i. The correct classification of $x_j^{(i)}$ needs to satisfy the defined constraints, and the loss function is a hinge loss determined for each document. This means that if the label $y_j^{(i)}$ is $+1$ and the model predicts a value equal to or greater than $+1$, the document will have zero loss. Otherwise, the loss is $\xi_j^{(i)}$.
The graphical representation is shown in Figure 2.4.

- **Multi-class**: A multi-class classification technique is proposed in *"McRank: Learning to Rank Using Multiple Classification and Gradient Boosting"* [27]. It takes the documents $x = \{x_j\}_{j=1}^m$ related to the query q and their relevance judgment $y = \{y_j\}_{j=1}^m$ and predicts $\widehat{y_j}$ on x_j. The loss function replaces the $0-1$ classification

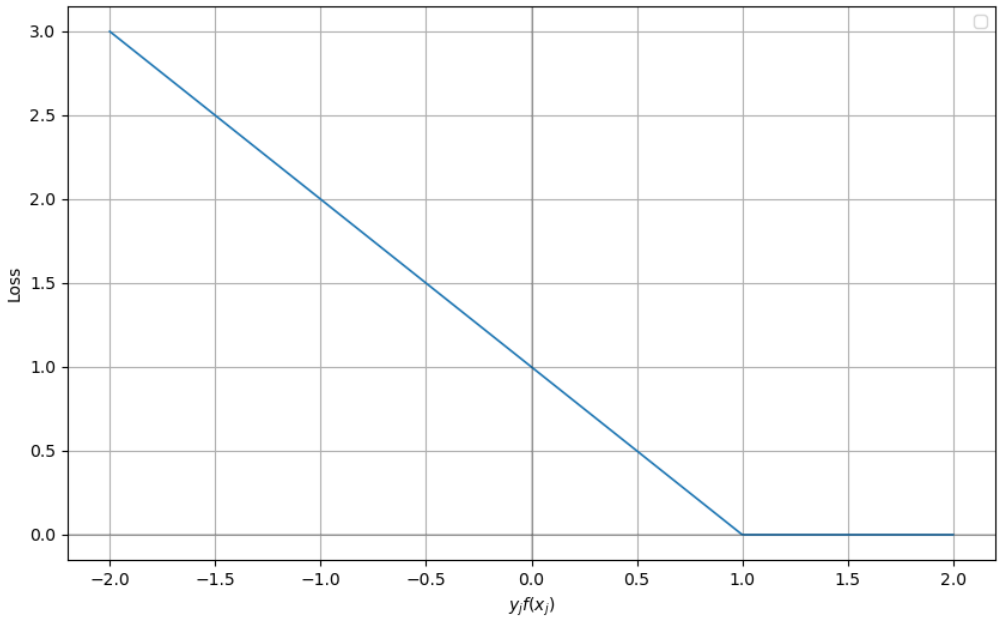

Figure 2.4: Hinge loss of the function $y_j f(x_j)$ (Insp. by [28])

error, $L(\widehat{y_j}, y_j) = I_{y_j \neq \widehat{y_j}}$, and is defined as

$$L_\Phi(\widehat{y_j}, y_j) = \sum_{j=1}^{m} \sum_{k=1}^{K} -log\, P(\widehat{y_j} = k)\; I_{y_j=k} \tag{2.8}$$

Belonging to a category is determined by a particular function, and the classifier's output is transformed into a probability, with a weighted combination being utilised to calculate a document's final ranking score.

2.2.2.1.3 Ordinal

This method considers the order of the true labels during the training phase.
The condition that is necessary is to have the categories that are K-ordered. The objective is then to find a scoring function f and consequently utilise the thresholds $b_1 \leq b_2 \leq ... \leq b_{K-1}$ to differentiate the outputs of the scoring function into distinct ordered categories.
PRanking [15] is such an algorithm. The algorithm projects into a vector w the documents and then the scope is to find the direction of this vector. To get the result it is necessary to do an iterative learning process, this means that the instance x_j, is defined at an iteration t associated with query q. After having x_j, the algorithm estimates $\hat{y}_j = \text{arg}min_k \left\{w^T x_j - b_k < 0\right\}$. Then compares the outputted value with y_j, and if the answer is incorrect then it indicates that a threshold k exists where the value $w^T x_j$ is not aligned with b_k. To fix this situation, the values of $w^T x_j$ and b_k are adjusted towards each other, and this implies that w is transformed in $w = w + x_j$; this is repeated

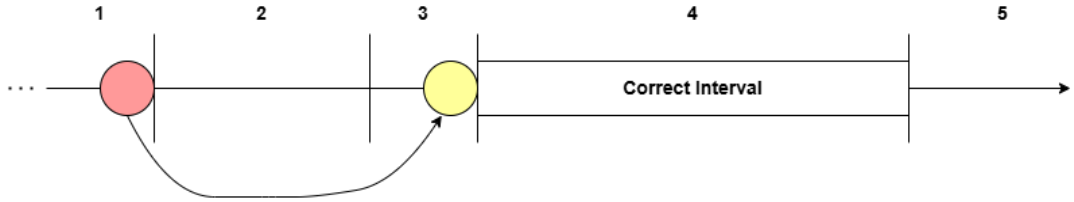

Figure 2.5: Learning process of PRanking. The red dot is the output created by the algorithm, the yellow one is how it is placed after the update (Ins. by [15])

until convergence. In Figure 2.5, an example of the learning process of this algorithm is presented.

2.2.3 Listwise

Here, the set of documents related to query q are the input space, this is denoted as $x = \{x_j\}_{j=1}^{m}$, and in the output space, there is the document's ranked list.
From the judgments, it is possible to get the true labels in the following way:

- If the judgment is of a relevant degree l_j, then all the permutations consistent with the judgment are ground truth permutations.
- In case of a pairwise preferences, all permutations consistent with the pairwise preferences are considered valid.

The hypothesis space uses a function h that predicts the order of the documents by using a scoring function f, this last function gives the score to each document and arranges them in such a manner that at the end is present a specific permutation.

The listwise methods have two types of loss functions: one is directly correlated to the evaluation measures, and the other is not. Consequently, the algorithms can be divided into those that use the first type and those that use the second.

2.2.3.1 Example of algorithms

2.2.3.1.1 Measure-Specific Loss

The objective here is to improve the metrics used to evaluate ranking performance. The issue is that the metrics usually used, like NDCG and MAP, are based on position, making them discontinuous and non-differentiable [35]. This challenges optimisation because most functions are created to deal with continuous and differentiable scenarios. One algorithm for this type of loss is *SoftRank* [43], which incorporates randomness into the ranking process by treating a document's score as a random variable whose mean is equal to the score derived from the scoring function. This execution allows a document to be placed in different positions with different probabilities. Then, for each ranking is calculated the NDCG and is made an average of all of them and is used as a smooth approximation to calculate the original NDCG.

2.2.3.1.2 Non measure-Specific

In this model the evaluation measures are not optimised directly way and the discrepancy between the model results and the ground truth permutation π_y is evident.
ListNet [10] is an algorithm that defines the loss function using the probability distribution. That is possible because the ranked list and the permutations are directly connected. ListNet then uses the Plackett–Luce model [30, 33] to apply this concept in LtR, and it defines the probability for each permutation π according to this chain rule:

$$P(\pi|s) = \prod_{j=1}^{m} \frac{\varphi(s_{\pi^{-1}(j)})}{\sum_{u=1}^{m} \varphi(s_{\pi^{-1}(u)})} \tag{2.9}$$

$\pi^{-1}(j)$ is the document ranked at the j-th position of the permutation π, and φ is a function that can be linear, exponential or sigmoid. ListNet defines two permutation probability distributions for a given query q: one is based on the scores from the scoring function f; the other, $P_y(\pi)$, is based on the ground true label. To create the loss function ListNet uses the *K-L divergence* between the probability distribution of the ranking model and the ground truth and combines it with the previous definition. This is defined by:

$$L(f; x, \Omega_y) = D\,(\,P_y(\pi)\;||\;P(\pi|(f(w,x))\,) \tag{2.10}$$

A gradient descent is sufficient for the global minimum because K-L is a convex loss function [47].
One big problem of this approach is its high complexity, which grows exponentially with m. The reason is that for each query q, when K-L is calculated, adding m-factorial is required and this makes the algorithm more complex.

2.2.4 Pairwise

In this method, the focus is on comparing pairs of items, which means that the ranking have to choose which item of the pair is the best one for the query.
The input here is formed by pairs of objects represented by feature vectors. The pairwise preference (which takes values from $\{+1, -1\}$) for every pair of items is the output of the method. The true labels are defined as:

- Relevance degree l_j and the pairwise preference for (x_u, x_v) are defined as $y_{u,v} = 2 \cdot I_{\{l_u \succ l_v\}} - 1$
- If the pairwise preference is defined then $y_{u,v} = l_{u,v}$

The hypothesis space defines a function h that receives a pair of documents and determines their relative order as output. In some algorithms [13] the scoring function f can be defined as $h(x_u, x_v) = 2 \cdot I_{\{f(x_u) > f(x_v)\}} - 1$. The objective is to minimise the number of misclassifications in the document pairs. It is crucial to note that the document pairs should not be independent since this would violate the basics of classification.
The loss function considers only a pair of items at a time because it considers only the relative order between them but this does not provide enough information to place the items in the final ranking. Additionally, the method overlooks that specific pairs are

generated from the documents associated queries.

One main algorithm in this kind of methods is *LambdaMART* [9], in the following sections are explained LambdaMART and the algorithms that are used to create it.

2.2.4.1 RankNet

RankNet [7] is a Pairwise algorithm so it has the hypothesis space constructed with a scoring function f, and the loss function is based on pairs of items. This model, as base that learns to do the ranks, can use boosted trees or neural networks. In the case that it uses a neural network and in particular if it is a two-layer neural network, such as the one in Figure 2.6, the features are taken as input, and the bottom layer takes care of them. Instead, the second layer is formed by different hidden nodes, each with a sigmoid transformation that gives the items a ranking.
The training phase is defined as follows:

- Training data is divided based on queries
- The input feature vector $x \in R^n$ is transformed into a value $f(x)$
- The model then takes for each query a pair of items, I_i and I_j, that have different labels, along with their corresponding feature vectors x_i and x_j
- The model calculates the scores $s_i = f(x_i)$ and $s_j = f(x_j)$
- Since the labels are different, it is evident that one of the two items should be ranked higher, and this information is defined by $I_i \rhd I_j$; in this case, I_i is required to be ranked above I_j
- The likelihood that I_i should be ranked higher than I_j is given by the sigmoid function, which leads to good probability estimates [4], such as:

$$P_{ij} \equiv P(I_i \rhd I_j) \equiv \frac{1}{1 + e^{-\sigma(s_i - s_j)}} \tag{2.11}$$

  Where σ determines the shape of the sigmoid

- The cost is then defined as

$$C = -\overline{P}_{ij}\, log P_{ij} - (1 - \overline{P}_{ij})\, log(1 - p_{ij}) \tag{2.12}$$

  It is given by a cross entropy function, which penalises any deviation of the model's predicted probabilities from the target ones.

As said pairs of items are taken into consideration, so if the i-th document is deemed more relevant than the j-th document, the variable defined as $S_{ij} \in \{0, \pm 1\}$ is 1, -1 otherwise; if not, then it is 0. It is possible to assume $\overline{P}_{ij} = \frac{1}{2}(1 + S_{ij})$, and this gives the following equation:

$$C = \frac{1}{2}(1 - S_{ij})\, \sigma(s_i - s_j) + log\left(1 + e^{-\sigma(s_i - s_j)}\right) \tag{2.13}$$

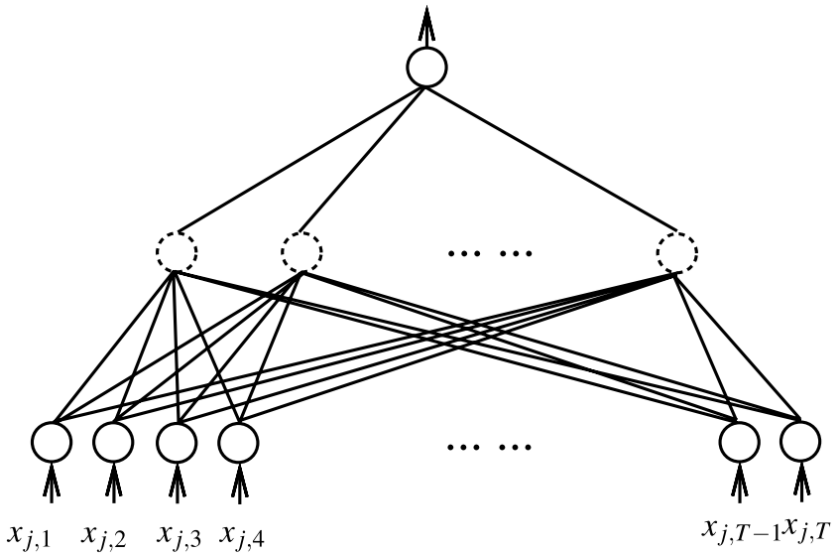

Figure 2.6: Two layer neural network [28]

The cost is symmetric, this means that it is not affected by the sign of S_{ij}, the position of i, and j. So, it is defined as

$$C = \begin{cases} \log\left(1 + e^{-\sigma(s_i - s_j)}\right) & \text{if } S_{ij} = \pm 1 \\ \log 2 & \text{if } s_i = s_j \end{cases} \tag{2.14}$$

When scores are assigned to documents with varying labels, they are moved further apart in the ranking. In the long run, if the scores result in an inaccurate ranking, the cost is linear; otherwise, it is zero. Thus is possible to have

$$\frac{\partial C}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 - e^{\sigma(s_i - s_j)}}\right) = -\frac{\partial C}{\partial s_j} \tag{2.15}$$

Then, the weights $w_k \in R$ are updated by the gradient as

$$\begin{aligned} w_k &\to w_k - \eta \frac{\partial C}{\partial w_k} \\ &= w_k - \eta \left(\frac{\partial C}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \frac{\partial C}{\partial s_j}\frac{\partial s_j}{\partial w_k}\right) \end{aligned} \tag{2.16}$$

η represents the positive rate of learning, then

$$\begin{aligned} \partial C &= \sum_k \frac{\partial C}{\partial w_k} \\ &= \sum_k \frac{\partial C}{\partial w_k}\left(-\eta \frac{\partial C}{\partial w_k}\right) \\ &= -\eta \sum_k \left(\frac{\partial C}{\eta w_k}\right)^2 \leq 0 \end{aligned} \tag{2.17}$$

In order to update the model, it is needed the gradient of the cost concerning the model parameters w_k, which necessitates them concerning the model scores s_i.

2.2.4.2 LambdaRank

LambdaRank is an evolution of RankNet that uses the gradients directly instead of deriving them from the cost function, but this does not mean that they are not cost gradients. The gradients are considered as forces, and is possible to think of them as arrows of λ that represent the magnitude of the contributions of the item. To apply this concept to a pair of items firstly the first item, I_1, can be assumed to be more relevant than the second, I_2. Then the second item is pushed down by a force equal to $|\lambda|$, and I_1 is displaced upward by the same quantity, obviously, if the values of the two items are the opposite, then the items will be moved oppositely.
The transformation of RankNet [8] into LambdaRank can be done in the following way, i.e., when considering a pair of items I_i, I_j, by combining the definitions given before, the gradient can be expressed as:

$$\begin{aligned}\frac{\partial C}{\partial w_k} &= \frac{\partial C}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \frac{\partial C}{\partial s_j}\frac{\partial s_j}{\partial w_k} \\ &= \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right)\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right) \\ &= \lambda_{ij}\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right)\end{aligned} \tag{2.18}$$

From this, is possible to derive

$$\lambda_{ij} \equiv \frac{\partial C(s_i - s_j)}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) \tag{2.19}$$

If it is increased by the magnitude of the change in NDCG, known as $|\Delta_{NDCG}|$, which it is obtained by swapping the ranking of only I_1 and I_2, then

$$\lambda_{ij} = \frac{\partial C(s_i - s_j)}{\partial s_i} = \frac{-\sigma}{1 - e^{\sigma(s_i - s_j)}}|\Delta_{NDCG}| \tag{2.20}$$

Since the priority is to maximise C, then the weights w_k are defined as

$$w_k \to w_k + \eta\frac{\partial C}{\partial w_k} \tag{2.21}$$

and

$$\partial C = \frac{\partial C}{\partial w_k}\delta w_k = \eta\left(\frac{\partial C}{\partial w_k}\right)^2 > 0 \tag{2.22}$$

Usually, there is the problem of a flat or discontinuous score functions, given that the gradients are calculated after the items are sorted based on their score. However this method overcomes this problem.

2.2.4.3 MART

MART [18] is a boosted tree model that produces its output by combining the outputs of multiple regression trees through a linear combination. As in the usual tree algorithms, the regression trees [29] use the data and corresponding labels to split into right and left nodes.
The output $F(x)$ can be written as

$$F_N(x) = \sum_{i=1}^{N} \alpha_i f_i(x) \tag{2.23}$$

Each $f_i(x) \in R$ is a function defined by a single regression tree, and the weight $\alpha_i \in R$ is associated with the i-th regression tree. f_i, α_i and γ_{kn} (the value associated with each leaf) are associated with the tree's leaf node and are determined through training.
Since the model is a boosted tree, the basic idea is that the next tree created should improve on the part where the previous one made mistakes, and in order to do so, MART uses the gradient descent. This means that the following tree is created using as a base the m derivatives of the cost evaluated in the previous points, so

$$\delta C \approx \frac{\partial C(F_n)}{\partial F_n} \delta F \tag{2.24}$$

This implies that if $\delta F = -\eta \frac{\partial C}{\partial F_n}$ then $\delta C < 0$. The new tree is added with a form of regularisation, so it is possible to adjust the relevance and importance of each tree, and this weight is given by $\eta\gamma_{kn}$, where η is the actual part that gives the regularisation.

2.2.4.4 LambdaMART

The combination of LambdaRank and MART creates *LambdaMART* [46]. In this case, every tree represents the λ_i values for the complete dataset, not just individual queries. Additionally, this algorithm requires the Newton step when the exact step size cannot be calculated. To achieve this, some specific preliminary steps are required and they are explained in the following points:

- After sorting the items based on the score, the pair I_i and I_j is taken, with the condition that $I_i \rhd I_j$, then it is possible to obtain λ_{ij} defined as

  $$\lambda_{ij} = \frac{-\sigma|\Delta Z_{ij}|}{1 + e^{\sigma(s_i - s_j)}} \tag{2.25}$$

  Z_{ij} represents the utility difference from swapping the two items rank positions

- Then

  $$\lambda_i = \sum_{j:\{i,j\}\in I} \lambda_{ij} - \sum_{j:\{j,i\}\in I} \lambda_{ij} \tag{2.26}$$

  Which simplified, becomes

  $$\sum_{\{i,j\}\rightleftharpoons I} \lambda_{ij} \equiv \sum_{j:\{i,j\}\in I} \lambda_{ij} - \sum_{j:\{j,i\}\in I} \lambda_{ij} \tag{2.27}$$

- With the previous definition, it is now possible to write the utility function where λ_i represents the utility's derivative for a particular item I_i

$$C = \sum_{\{i,j\} \rightleftharpoons I} |\Delta Z_{ij}| \, log\left(1 + e^{-\sigma(s_i - s_j)}\right) \tag{2.28}$$

and can be rewritten as

$$\frac{\partial C}{\partial s_i} = \sum_{\{i,j\} \rightleftharpoons I} \frac{-\sigma |\Delta Z_{ij}|}{1 + e^{\sigma(s_i - s_j)}} \equiv \sum_{\{i,j\} \rightleftharpoons I} -\sigma |\Delta Z_{ij}| \, \rho_{ij} \tag{2.29}$$

From which it is obtainable ρ_{ij} as

$$\rho_{ij} \equiv \frac{1}{1 + e^{\sigma(s_i - s_j)}} = \frac{\lambda_{ij}}{\sigma |Z_{ij}|} \tag{2.30}$$

- Then the second derivative is defined as

$$\frac{\partial^2 C}{\partial s_i^2} = \sum_{\{i,j\} \rightleftharpoons I} \sigma^2 |\Delta Z_{ij}| \, \rho_{ij}(1 - \rho_{ij}) \tag{2.31}$$

Following these steps, is possible to establish the Newton step size for the k-th leaf of the m-th tree as

$$\begin{aligned} \gamma_{km} &= \frac{\sum_{x_i \in R_{km}} \frac{\partial C}{\partial s_i}}{\sum_{x_i \in R_{km}} \frac{\partial^2 C}{\partial s_i^2}} \\ &= \frac{-\sum_{x_i \in R_{km}} \sum_{\{i,j\} \rightleftharpoons I} |\Delta Z_{ij}| \rho_{ij}}{\sum_{x_i \in R_{km}} \sum_{\{i,j\} \rightleftharpoons I} |\Delta Z_{ij}| \, \sigma \rho_{ij}(1 - \rho_{ij})} \end{aligned} \tag{2.32}$$

The LambdaMART algorithm has the possibility to customise the model to use scores from a starting base model.

As said, LambdaMART is a combination of MART and LambdaRank, but one difference is in how LambdaRank and LambdaMART manage the update of their parameters. The first adjusts the weights after each query, while the second makes the decisions using all the data in a node at a particular time. This means it updates only some parameters at time and not all of them. In particular, it chooses to split based on the leaf values that decrease the utility for specific queries as long as the overall utility improves.

2.3 Fairness

Lately, machine learning has been widely used in diverse social settings but when algorithms are written, are considered factors like performance, accuracy, and some aspects such as privacy and transparency. One characteristic that is not so much considered is fairness, but this is an important aspect to take care of considering that it is crucial to avoid discrimination and bias in information since they may have important social implications such as mistreat to some groups based on specific sensitive attributes like sex,

Algorithm 1: LambdaMART

Data: The number of trees N is needed, number of training samples m, number of leaves per tree L and learning rate η

Result: Ranking model represented as a sum of N trees

for $i = 0$ *to* m **do**
  $F_0(x_i) = \text{BaseModel}(x_i)$
end
for $k = 1$ *to* N **do**
  for $i = 0$ *to* m **do**
    $y_i = \lambda_i$
    $w_i = \frac{\partial y_i}{\partial F_{k-1}(x_i)}$
  end
  $\{R_{lk}\}_{l=1}^{L}$
  $\gamma_{lk} = \frac{\sum_{x_i \in R_{lk}} y_i}{\sum_{x_i \in R_{lk}} w_i}$
  $F_k(x_i) = F_{k-1}(x_i) + \eta \sum_l \gamma_{lk} I(x_i \in R_{lk})$
end

gender, age and many others. Given that this behaviour it is unethical the community has begun to acknowledge more the issue in recent years.

It is not easy to define the concept of *Fairness* because there are more than twenty definitions, and they all depend on the specific case and data to which it is applied. Certain characteristics include whether fairness is based on individuals, which means that similar individuals are treated similarly, or on groups, which ensures equity among different groups. The models that focus on group fairness can be divided into two categories: those that are for *We're All Equal (WAE)*, where all the groups have the same abilities, and those in which is valid *What You See is What You Get (WYSIWYG)*, where observations reflect group characteristics.

2.3.1 Attributes

Attributes help determine whether an object qualifies as belonging to a protected category. For this reason, individuals with the same value of a particular sensitive attribute, such as gender or race, are referred to as a group. Generally, the methods that resolve fairness problems can be categorised based on how they treat these variables.

Cardinality Algorithms may be categorised according to the number of sensitive attributes considered. Specific algorithms focus on *binary* sensitive attributes while others handle higher cardinality, and these are called *multinary*. In the first case, the data is divided based on belonging to a specific attribute like gender or ethnic group. In the second one, multiple sensitive are represented simultaneously; for example "women" and "Asians". There is further categorisation when considering the multinary variables

because some methods can consider one of the values protected, while others can treat all values of the sensitive attributes as potentially subject to discrimination.

Number of sensitive attributes A different categorisation is determined by the number of attributes an algorithm can process. Algorithms can be divided into those capable of processing only one attribute, like gender *or* race, and those capable of processing multiple attributes simultaneously, such as race *and* gender, together. The second group is further divided based on their approach to attributes, either separately, which means that, for example, it can handle both categories "women" and "black people", or together, then for example it can handle the category of "black women".

2.3.1.1 Bias

As mentioned, data bias is the primary reason behind unfair algorithms, and it is possible to categorise different biases [17]:

- **Pre-existing bias**: Includes all biases present without being influenced by an algorithm and originates from social factors. In some instances, racial and class disparities result in unequal opportunities for individuals
- **Technical bias**: Refers to how elements are constructed, such as the size of a screen or how they are positioned in a list; higher items have more visibility than lower ones or how they are positioned on a page. In the Western world, we read from top to bottom and from left to right, and so items in the top-left corners have more visibility [2]
- **Emergent bias**: Can arise when a system was initially intended for a specific scenario but evolves with societal norms, creating a scenario in which the "winner takes it all" attitude prevails

2.3.2 Mitigation methods

Depending on the type of problems solved, whether score-based or learning to rank, there are different methods to mitigate the bias. The concept remains the same in both methods; the principal difference is when the intervene in the process is done. In fact the possible work can be done before, during, or after the execution.

2.3.3 Pre-Processing

This method aims to learn a fair representation and eliminate bias, that can be technical, emergent or pre-existing, while preserving other important information. The resulting algorithm then uses the updated data with reduced bias.
The advantages of this mitigation technique are:

- In the pipeline, the top priority is ensuring fairness

- Unlike other methods, which depend on the labels of the data, here the protected group is chosen considering the proximity between objects
- It provides increased management of data even when it is only partially comprehensive

A disadvantages is that it can handle only individual fairness and considers group fairness as a specific instance of individual fairness.

2.3.3.1 Example of algorithms

2.3.3.1.1 iFair

iFair [26] is based on the principle of fairness given by "Fairness Through Awareness" [16], defined by individual fairness, where similar items are treated alike. The algorithm changes input record X_a into a more equitable representation $\widetilde{X}_a$ using a mapping function ϕ. So, for example, when taking two individuals, a and b, that are indistinguishable when considering only the non-sensitive attributes after the transformation, they should also be indistinguishable in the fairer representation. As said, here are not taking in consideration the labels but only the distances, so it is defined the similarity measure, d, free of bias, capable of capturing the diversity between the two items

$$\left|d(\phi(x_i), \phi(x_j)) - d(x_i^*, x_j^*)\right| \leq \epsilon \tag{2.33}$$

$\phi(x_i)$ is the fair representation, x_i^* is the feature vector containing only non-sensitive features, and ϵ is the threshold defining when the functions are distinguishable.
The algorithm process is defined as follows: it searches for similar items based on the non-protected attributes, then creates a new feature set, $\phi(X)$, that maintains distances while incorporating sensitive attributes to disrupt the correlation of non-sensitive attributes defined before.
The algorithm formalises the problem as a clustering problem, so there are K clusters composed of similar individuals and a prototype vector v_k representing each cluster. Next, a candidate record, X_a, is allocated to one of the vectors using a probability distribution u_i and is calculated the distance between all the clusters to find the most similar. The fair representation is then

$$\phi(x_i) = \widetilde{x}_i = \sum_{k=1\ldots K} u_{ik} \cdot v_k \tag{2.34}$$

The loss fair, which maintains distances between data records on non-protected attributes, is defined as

$$L_{\text{fair}}(X, \widetilde{X}) = \sum_{i,j=1,\ldots M} \left(d(\widetilde{x}_i, \widetilde{x}_j) - d(x_i^*, x_j^*)\right)^2 \tag{2.35}$$

The loss utility is

$$L_{\text{util}}(X, \widetilde{X}) = \sum_{i=1}^{M} ||x_i - \widetilde{x}_i||_2 = \sum_{i=1}^{M} \sum_{j=1}^{N} (x_{ij} - \widetilde{x}_{ij})^2 \tag{2.36}$$

The objective function, which has to be minimised, is a combination of the data loss and the fairness loss and is defined as

$$L = \lambda \cdot L_{\text{util}}\left(X, \widetilde{X}\right) + \mu \cdot L_{\text{fair}}\left(X, \widetilde{X}\right) \tag{2.37}$$

λ and μ are hyper-parameters, so to have better performances it is necessary to try different values for them.

2.3.4 In-Processing

In this case, the algorithms expand the objective function with a fairness component. Therefore, the algorithm's optimisation problem involves both accuracy objective and fairness objective; the technique then works on finding a balance between the two.
Some advantages are:

- It has better trad-offs between fairness and accuracy since it is the main objective
- Can manage various biases without identifying a specific one

Some disadvantages are:

- The impact that it gives is not as evident as other methods
- It is not easy to understand what the method prioritises due to the need to balance accuracy and fairness

2.3.4.1 Example of algorithms

Some algorithms that uses this kind of method are: *DELTR* [54]; *Fair-PG-Rank* [40]; *Pairwise Fairness for Rankings* [5].

2.3.4.1.1 DELTR

DELTR is a ListNet extension and is based on "Fairness of Exposure in Rankings" [39]. The fairness definition relies on the differences in average visibility that determine disparities in exposure and this affects the final ranking.
The exposure of document d, in a ranked list generated by a probabilistic ranking P and adapted to match ListNet's accuracy metric, is given by the probability of appearing in the top position of a ranking for a specific query and is

$$\text{Exposure}\left(x_i^q | P_{\widehat{y}^q}\right) = P_{\widehat{y}^q}(X_i^q) \cdot v_1 \tag{2.38}$$

v_1 is the *position bias* of position 1, which signifies how essential an item is in the ranking [23]. The average exposure of documents in group G_p with $p \in \{0, 1\}$ is

$$\text{Exposure}(G_p | P_{\widehat{y}^q}) = \frac{1}{|G_p|} \sum_{x_i^q \in G_p} \text{Exposure}(x_i^q | P_{\widehat{y}^q}) \tag{2.39}$$

An unfairness criterion measured in terms of disparate exposure is

$$U(\widehat{y}^q) = \max\left(0, \text{Exposure}(G_0)|P_{\widehat{y}^q}) - \text{Exposure}(G_1|P_{\widehat{y}^q})\right)^2 \tag{2.40}$$

G_0 represents the non-protected group, and G_1 represents the protected group. Only if the protected group receives less exposure can the asymmetric hinge loss be identified as unfair. The accuracy metric L, which uses the function defined by ListNet, is defined as

$$L_{\text{DELTR}}(y^q, \widehat{y}^q) = L(y^q, \widehat{y}^q) + \gamma\, U(\widehat{y}^q) \tag{2.41}$$

$\gamma \in R_0^+$ is a parameter that creates a trade-off between ranking utility and disparate exposure. By increasing γ, the focus is more on minimising the differences in exposure. Otherwise, it is more on discrepancies between the training data and the ranking algorithm's output.

2.3.5 Post-Processing

The assumption in this case is that the model has been trained; thus, a predicted ranking is provided for the method to re-order items to improve fairness. Usually, the methods are group based, where certain groups are protected, and one is non-protected. Some benefits of this technique are:

- The guarantee that the protected group has excellent visibility
- The re-ordering and utility measures used and how they affect the ranking are apparent and comprehensible

Some disadvantages are:

- Increasing fairness can reduce the accuracy
- A small amount of bias can lead to a notable decrease in performance

2.3.5.1 Example of algorithms

There are different algorithms such as: *FA*IR binary* [52] and *multinary* [53]; *Fairness-Aware Ranking at Linkedln* [21];*Continuous Fairness with Optimal Transport* [51], *Fairness of Exposure* [39] and *Equity of Attention* [6].

2.3.5.1.1 FA*IR

FA*IR, in the binary case, is based on the concept that it is important to achieve statistical parity between the protected and not protected groups. It defines fairness by assuming that a ranking is fair if items positions are decided thanks to a Bernoulli distribution unaffected by the candidate's sensitive attributes. Thanks to a statistical evaluation that determines if a Bernoulli process could have generated the ranking, it is guaranteed that the proportion of protected candidates does not exceed a specific minimum percentage p. Given a minimum proportion p of elements at each position in the ranking, is possible to pre-compute a table that contains this values. This is done

by calculating $F^{-1}(\alpha; k; p)$, i.e., the inverse of the binomial CDF. Once these tables are created, sorting them individually in descending order of scores and combining them into one ranking is possible. If the ranking does not meet the minimum quantity defined before, a protected item is placed so the proportion is preserved.

2.3.6 Intervening on the Ranked Outcome

This technique is like the post-processing, so the idea is to take action on the ordered result and set limitations to ensure a certain amount of variety or representation within the top-k.

2.3.6.1 Example of algorithms

Some works are: *Ranking with Fairness Constraints* [12]; *Rank-aware Proportional Representation* [50]; *Balanced Ranking with Diversity Constraints* [48]; *Online Set Selection with Fairness and Diversity Constraints* [42].

2.3.6.1.1 Rank-aware Proportional Representation

Considering two groups: G_1, the protected group, and G_2, the non-protected. The items are assigned to them based on a single binary sensitive attribute. The central concept of fairness emphasises the importance of an item being ranked higher; this implies that it is more critical to achieve proportional representation in higher positions. A ranking shows statistical parity when being part of a protected group does not impact where an item is placed in the final rank. The fairness measures defined are based on sets, meaning that the values for *top-b*, *top-2b*, *top-3b*, etc..., where b is a parameter that indicates how many elements are considered in each set, are combined using a logarithmic function. This discount prioritise higher positions over lower ones, emphasising the principal concept of the method. All the measures described here have values in $[0, 1]$, where closer to 0 means that the rank it is more fairer.
There are three measures that express this concepts in different ways:

- **Normalized discounted difference (rND)** is defined as

$$rND(\tau) = \frac{1}{Z} \sum_{k=b,2b,..}^{N} \frac{1}{log_2\, k} \left| \frac{|S^+_{1...i}|}{k} - \frac{|S^+|}{N} \right| \tag{2.42}$$

  It computes the difference in the proportion of members from the protected group at the top-k positions and the overall population. As seen in the formula, the values are accumulated at discrete points in the ranking with a logarithmic discount and finally normalised. The normaliser Z is computed as the highest possible value of *rND*.
  It is possible to observe that this formula is convex, continuous, and not differentiable at 0. If one of the groups, S^+ or S^-, has a more significant proportion, then the rank is unfair.

  Figure 2.7 shows how rND behaves on synthetic datasets of 1000 items with 300,

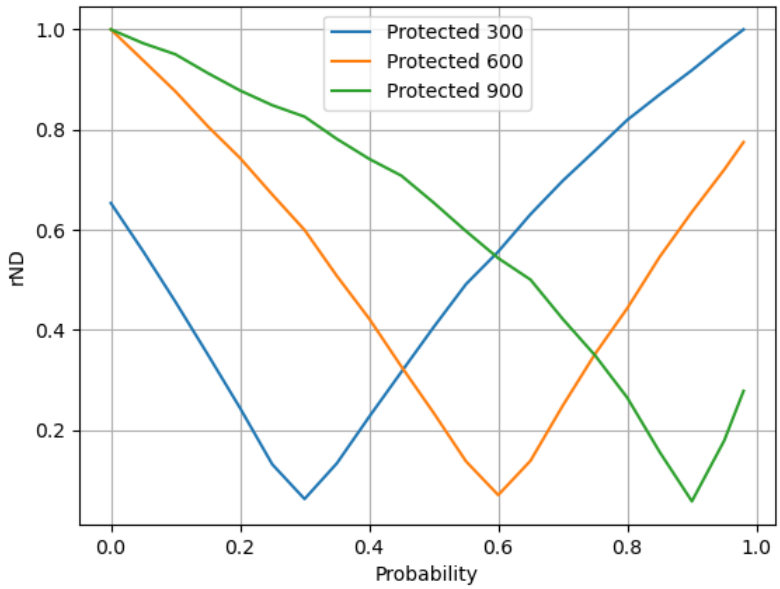

Figure 2.7: Example of rND execution

600 and 900 items in S^+. In particular it creates a fake ranking based on the probability given, which is between 0 and 0.98, and it creates a random value $u \in [0, 1]$, and if it is smaller than the probability taken in consideration at that moment, then is putted a protected item firstly, otherwise it is putted in the non protected one. This way it is possible to choose how many items of each group will be at top and can be created different possible ranks. After this ranks are created then the measure *rND* is applied and given the relative score. This method is also applied to the other measures and examples.

- **Normalized discounted KL-divergence (rKL)**: Kullback-Leibler (KL) calculates the average of the logarithmic discrepancy between two discrete probability distributions P and Q:

$$D_{KL}(P||Q) = \sum_k P(k)\ log\frac{P(k)}{Q(k)} \tag{2.43}$$

With this, it is possible to calculate the difference between how the protected group is treated in the top k and how it is treated in the rest of the population

$$\begin{aligned} P &= \left(\frac{|S^+_{1...i}|}{k}, \frac{|S^-_{1...k}|}{k}\right), \\ Q &= \left(\frac{|S^+|}{N}, \frac{|S^-|}{N}\right) \end{aligned} \tag{2.44}$$

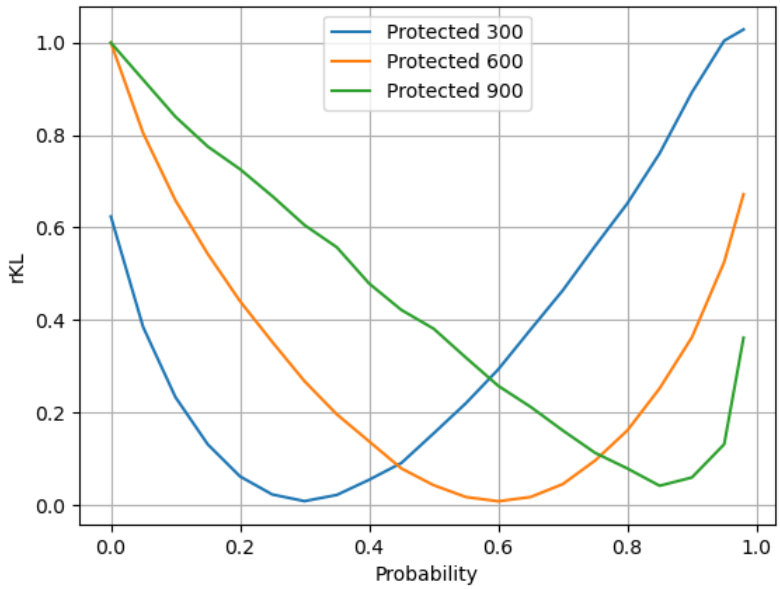

Figure 2.8: Example of rKL execution

Normalised discounted KL-divergence (rKL) is then defined as

$$rKL(\tau) = \frac{1}{Z} \sum_{k=b,2b,\dots}^{N} \frac{D_{KL}(P||Q)}{log_2 k} \tag{2.45}$$

Figure 2.8 shows how rKL behaves on synthetic datasets of 1000 items with 300, 600, and 900 items in S^+.

- **Normalized discounted ratio (rRD)**: It is defined similarly to the rND, but instead of using the number of elements for the denominator is used the number of non protected elements till position k. rRD is then defined as

$$rRD(\tau) = \frac{1}{Z} \sum_{k=b,2b,\dots}^{N} \frac{1}{log_2 k} \left| \frac{|S^+_{1\dots k}|}{|S^-_{1\dots k}|} - \frac{|S^+|}{|S^-|} \right| \tag{2.46}$$

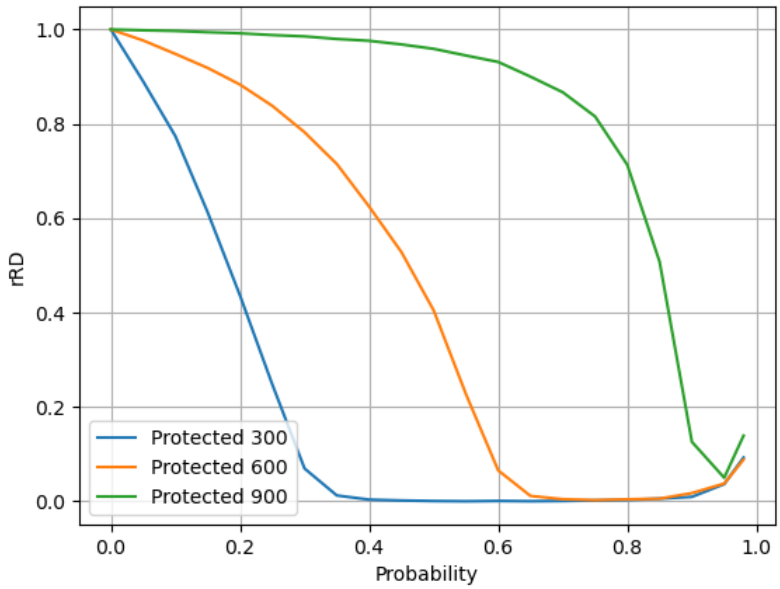

Figure 2.9: Example of rRD execution

Figure 2.9 shows how rRD behaves on synthetic datasets of 1000 items with 300, 600 and 900 items. By looking at the example of execution, and the fact that the two groups are not treat equally, is possible to observe that *rRD* has good results when S^+ corresponds to at most 50% of the underlying population, and the fairness probability is less than 0.5.

2.3.6.1.2 Balanced Ranking with Diversity Constraints

Here, in-group fairness is addressed, which requires that the best items for each category be placed in the top k positions in the final ranking. In order to do so, two measures are presented:

- **IGF-Ratio**: Its value is in $(0, 1]$, with higher values meaning that fairness is more respected. The idea is to quantify the utility of the group by calculating the ratio between the most valuable item skipped and the lowest one put in the rank, and this is given by

$$\text{IGF-Ratio(v)} = \frac{a_v}{b_v} \tag{2.47}$$

  a_v is the score of the lowest value item accepted, and b_v is the highest valued item rejected

- **IGF-Aggregated**: This is similar to *IGF-Ratio*. The difference is that there are not only pairs of items but all the items of a specific group are taken in consideration. In particular, it takes all the items of the group and all the ranked items of the same group. This is represented by

$$\text{IGF-Aggregated}(v) = min_{i \in A_v} \left\{ \frac{\sum_{h \in A_i} s_h}{\sum_{h \in I_{i,v}} s_h} \right\} \tag{2.48}$$

The numerator is the sum of all ranked items, and the denominator is the sum of all the group items; in both cases, it takes only the items up to position k. This gives a fair result when all the qualified items are taken; otherwise, the gap between the two will be significant, meaning there is an unfair rank.

2.3.7 Intervening on the Score Distribution

All the algorithms here assume that there is a pre-existing bias such that the ranker disadvantages the non-protected groups. So, the objective is to adjust the score before it is given to the ranker.

2.3.7.1 Example of algorithms

Some algorithms with this technique are: *Selection Problems in the Presence of Implicit Bias* [25]; *Interventions for Ranking in the Presence of Implicit Bias* [11]; *Causal intersectionality and fair ranking* [49].

2.3.7.1.1 Causal intersectionality and fair ranking

Here, fairness is defined by how the rank will change if an item belongs to a different group than the original one; by this concept, the new rank is generated by some fabricated scores. The model uses a *structural causal model (SCM)* that consists of a directed acyclic graph $G = (V, E)$ where V represents the variables, and E depicts the casual relationship from source to target vertices, as illustrated in Figure 2.10. By looking at the arrows of the graph, it is possible to understand how direct and indirect discrimination works. To be direct, it is just necessary that the arrows point from a protected variable to the output node; it is considered indirect if it passes through a mediating variable, which can be resolving or not. If it is resolving, then it is just necessary to resolve discrimination problems generated by the protected variables; if it is non resolving, it is also necessary to resolve the discrimination created by the mediator.
Having calculated the vector of sensitive attributes A and new possible values a', the goal is to calculate the counterfactual $Y_{A \leftarrow a'}$ by replacing the original values with the new ones. The changes are then propagated in the graph, creating the new ranking. The changes are applied to the variables defined by A and the relative non-resolving attributes, not to the resolving ones.
A ranking $\widehat{\tau}$ can be considered fair if for all possible values x for all the attributes $a \neq a'$ then

$$\begin{aligned} &P\left(\widehat{\tau}(Y_{A \leftarrow a}(U)) = k | X = x, A = a\right) \\ =&P\left(\widehat{\tau}(Y_{A \leftarrow a'}(U)) = k | X = x, A = a\right) \end{aligned} \tag{2.49}$$

This is valid for any rank k.

2.3.8 Intervening on the Ranking Function

The motivation to use this technique is that sometimes the initial formula results in one group being present in a lower percentage, so adjustments are required to ensure the population is in equal proportion.

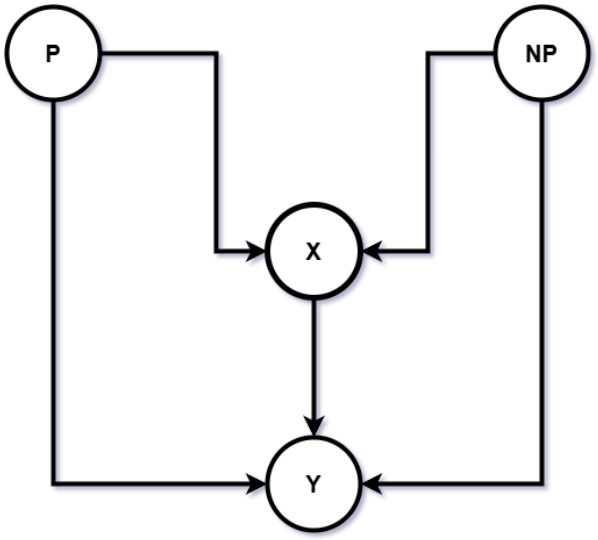

Figure 2.10: SCM Example (Insp. by [49])

2.3.8.1 Example of algorithms

One algorithm that does this kind of work is *Designing Fair Ranking Schemes* [1].

2.3.8.1.1 Designing Fair Ranking Schemes

Here is used a combinatorial geometry approach to navigate the search space and identify a fair scoring function $\widetilde{f}$ close to f regarding the angular distance between their weight vectors, if such a function exists.
This algorithm uses a fairness oracle O that accepts a dataset C and a linear ranking function f, with a weight vector $\vec{w}$. The oracle returns *True* if the ranking satisfies the fairness criteria and *False* if it does not.

2.4 Datasets

Different datasets are used in the field of learning to rank and fairness, but for this thesis, we decided to use the following two: *Microsoft Learning to Rank* [34] and *German Credit* [22].

2.4.1 Microsoft Learning to Rank Dataset

Using the queries and logs provided by the search engine Bing, Microsoft created two datasets with different queries written by users, one with 10.000 queries, *MSLR10K* and one with 30.000 queries, *MSLR30K*. The dataset is divided into five folders because a five-fold cross-validation is applied to them. Each dataset represents the data as a pair composed of the query, and a vector of variables represents the associated document. The dataset, MSLR30K, contains an average of 120.19 documents for each query and the relevance associated with each document is represented by a value $v \in \{0, 1, 2, 3, 4\}$, with higher values meaning that the document is more relevant to the query. The vector representing the document associated with the query contains 136 variables. The first

100 contain information such as the value of TF*IDF, variance of term frequency, stream length, etc. The remaining variables give information about the results from a language model, the page rank, url click count or quality of the page and others.

2.4.2 German Credit

This dataset contains the financial information of 1.000 individuals and is usually used by a binary classification task that predicts whether an individual's credit is good or bad. The information found here is about each individual such as the credit taken; the scope and quantity of credit; the age; the sex; the type of work, and other information.

Chapter 3

Experimental Analysis

In this chapter, we will explain the process that we applied to solve the Fairness problem explained in section 2.3. Firstly, we are going to explain the motivation behind our solutions, Section 3.1, then how we implemented it, Section 3.2, and finally, the results of some experiments done are shown in Section 3.4.

3.1 Motivations

The objective of the thesis was to implement a learning to rank algorithm that its optimisation based on efficiency and fairness. To do so, we used the LightGBM library [38], which uses LambdaMART, Section 2.2.4.4, as its basis. The type of bias we treat is the pre-existing one, and to solve it, we used the *rND metric*, Section 2.42, which has the scope to maximise the statistical parity between different groups, allowing us an exemplary implementation with LambdaMART.

LightGBM is one of the best-performing learning models for ranking. It is based on the LambdaMART method, and thus on Trees and Gradient Boosting, i.e., it creates each time trees that correct the errors of the previously created one. Trees are created using the Leaf-Wise technique, i.e., they are created layer by layer. One of the reasons for its speed is the fact that when it has to choose how to divide the data in the feature tree, it chooses which to divide using a particular histogram, sum of gradients and sum of squared gradients. Another thing done for optimisation is not to consider all the data but only those with more significant gradients. One more advantage is that it allows parallel processes to be used and the GPU to be used for the calculations, thus further optimising the timing.

3.2 Implementation

Given that the objective is to maximise both performance and Fairness, in order to do so, we used the formula of the gradient λ_i of LambdaMART, equation 2.26, and integrated in it the part of the Fairness, then now it becomes

$$\lambda_i = \alpha\lambda P_i^{NDCG} + (1-\alpha)\lambda F_i^{rND} \tag{3.1}$$

P is the pairwise preference related to *NDCG*, which influences the *performance*, while F is the one for *rND*, which is related to *Fairness*. The hyperparameter $\alpha \in [0, 1]$ allows us to choose which aspect is more relevant; for simplicity, we will call $(1 - \alpha)$ as β.

The idea behind the model is quite simple, but the difficulty lies in the fact that is difficult to order the items to have a fair rank because there is not a defined order that assures the fairness. When we calculate the order of the items based on the performance, it is relatively simple because we have *NDCG*, which is calculated from the ordering done on the relevance labels of the items, and then the rank is simple to do. However, when calculating the order for the fairness, we can not do the same because *rND* does not specify an order of the items but only the proportion of protected and non protected. Nevertheless, we can take advantage of this definition of *rND* because since it only cares about proportion, we can swap position of items belonging to the same group and that are inside the same set. This way, we can maintain the same proportion but improve the final rank to optimise performance and fairness. From this method, we can conclude that the two measures are strictly related, allowing us to reach the algorithm's scope.

As said, we used LightGBM as an algorithm and then modified it to adapt to the integration of *rND*. The code is based on the one provided by the paper [41], but there are some differences. One difference is in how the normaliser Z is calculated; the original paper uses two constant NORM_ITERATION and NORM_CUTPOINT to specify the max iteration and batch size used in the calculation. First, the maximum value of the input group fairness measure is obtained at different fairness probabilities. Run the above calculation NORM_ITERATION times. Then, compute the average value of the above results as the maximum value of each fairness probability. Finally, the maximum value is chosen as the normaliser of this group's fairness measure. Since we considered this solution not optimal, then we implemented it another way. Our idea is based on the fact that since Z represents the highest possible value of rND, it represents the worst ranking for fairness. This happens when all the predictions are grouped together based on the category, so all the protected items are on one side and the non protected on the other. Based on this idea the calculation of Z is done in the following way: the items are ordered such that all the protected are at the beginning and the relative rND is calculated. After this the same procedure is repeated but with the protected items that are all putted at the end. Then to get the value of Z we take the maximum between the two *rNDs* calculated. Another thing added is stopping at $@T$ elements; that is because, as said, we are only interested in the top elements and not all of them. One more difference is that LightGBM skips the pairs with the same relevance and we added the case that the elements that are both protected or belong to the same set are skipped, allowing a faster algorithm.

Since the algorithm of rND is considered at sets of K elements and considering that the different calculations should be made for each query, that is because this is how the algorithm of LightGBM is done, to improve the performance, we decided to calculate for each query, the number of protected, non protected, the length of the query, the max rND that it is possible to have and save them in arrays, so when needed in the process, it is just necessary to access the data and not recalculate them each time.

As said, the difficulty is in ordering the elements for the fairness measure. To achieve this we created three different strategies.

3.2.1 First Strategy

The first and most straightforward strategy is to take the rank produced by the algorithm at iteration l and take two items, i_i and i_j such that $i_i \rhd i_j$. Then ΔrND_{ij} is calculated, representing the change in rND after swapping the rank positions of items i_i and i_j. Specifically, it is interpreted in the following way: in case i_i is ranked higher than i_j and is correct, more fair, then after the swap, the value of the fairness parameter increases and $\Delta rND_{ij} > 0$; otherwise, if i_i should be below i_j then $\Delta rND_{ij} < 0$. From this, we can define

$$F = \{(i,j) \,|\, \Delta rND_{ij} > 0\} \cup \{(j,i) | \Delta rND_{ij} < 0\} \tag{3.2}$$

In this case, the parameter F is independent of P, so the condition of adjustment between the two is not guaranteed.

3.2.2 Second Strategy

This second strategy overcomes the problem of the first one by combining the two metrics. In particular, it creates a ranking, defined as π_2, based on the concept that the NDCG must be maximised and rND is at its minimum, meaning that fairness is prioritised over performance. F is then defined as

$$F = \{(i,j) \,|\, ind_b(i_i, \pi_2) < ind_b(i_j | \pi_2)\} \tag{3.3}$$

ind_b is a function that gives the index of the set that contains a specific item.
The ranking is created in the following way:

1. Items are sorted based on relevance to the query q, at this step, $NDCG$ is maximised, and the problem of more items with the same relevance label is resolved since the scores given by the algorithm give the order

2. Some changes are made to respect the proportion of several protected items over the total number of items. The changes made move some elements from one set to another, trying to keep the previous order the same. This way, rND is minimised without changing too much the value of $NDCG$

3.2.3 Third Strategy

In this case, the idea is similar to the second strategy, but here, we give more importance to $NDCG$, so the scope is to minimise rND under the constraint that $NDCG$ is maximised. Then, F is defined as

$$F = \{(i,j) \,|\, ind_b(i_i | \pi_3) < ind_b(i_j | \pi_3)\} \tag{3.4}$$

where ind_b is defined as in the second strategy and π_3 is the ranking that prioritises performance over fairness and is created in the following way:

1. Items are sorted based on relevance to the query q, at this step, *NDCG* is maximised, and the problem of more items with the same relevance label is resolved since the scores given by the algorithm give the order

2. We consider only the items that have the same relevance and change only them. This way, *rND* can be minimised while *NDCG* can be at its maximum

3.3 Dataset modification

As said in Section 2.4, we used the MSLR and German Credit datasets, but they are not suitable as they are and need to be modified.
MSLR is modified following the idea of the paper *"Probabilistic Permutation Graph Search: Black-Box Optimization for Fairness in Ranking"* [45] which takes the feature **QualityScore2**, attribute id 133, as the discriminatory feature, and uses the value 10 as threshold to divide the documents into two groups with a ratio of $3 : 2$ between them.

For the German Credit, we created new data because the data in the original dataset was not enough to do some relevant tests. For each query, we sampled 50 individuals with a ratio of $4 : 1$ for non-creditworthy to creditworthy individuals, and so in total now, we have 100.000 queries. Then, as protected variables, we consider **gender**, whether *Male* or *Female*, and **age**, if they are under 35 years or over.

3.4 Experiments

Now, we are going to show some of our experiments that are all made with $b = 5$ and the cutoff at 15 and 50. Unless otherwise stated, all the following results are based on the validation set.

3.4.1 LambdaMART on the datasets

Firstly, let us see how the basic LambdaMART model performs on the three datasets created: MSLR, German Credit with Age protected, German Credit with Sex as protected variable.

In Figures 3.1, 3.1, 3.1 3.2, we can see that the model has a Fair score not so bad, since it is under 0.5, remembering that nearer it is to 0 fairer it is, and it is not its primary scope. At a first observation there is not a correlation between the cutoff and the quality, we can observe this by looking at the dataset German Sex, where with @15, has better fairness. But if we look at the other datasets the case @50 have better fairness. In Figure 3.3 and 3.4, we can see the accuracy based on the NDCG measure; in the case of MSLR, the performance is not bad since it is over 0.5 and in the other datasets, it reaches 1.0, or almost 1.0. Because, as said the dataset is created by repeating data and at some point the algorithm learns the different patterns.

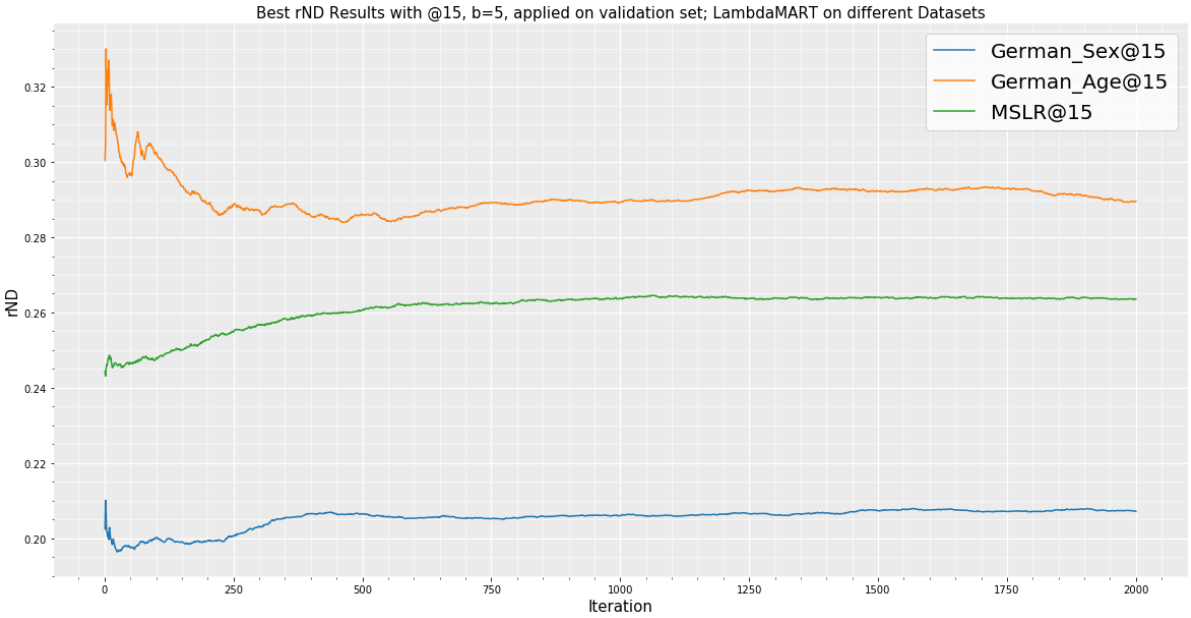

Figure 3.1: Results of LambdaMART rND@15 on the validation set of the three datasets

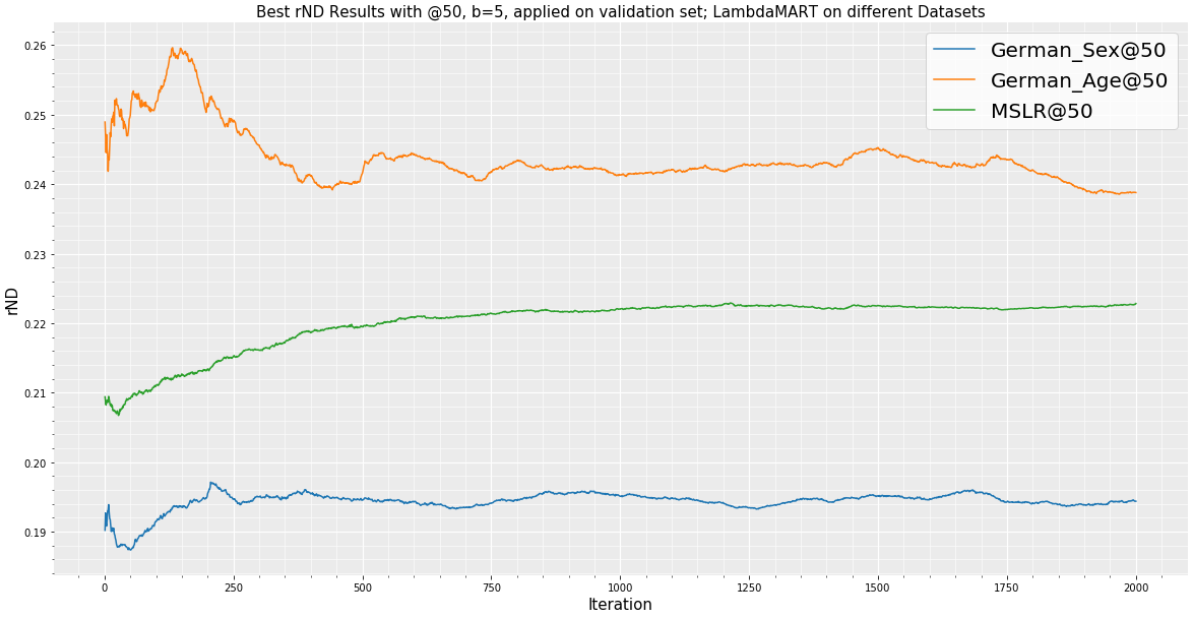

Figure 3.2: Results of LambdaMART rND@50 on the validation set of the three datasets

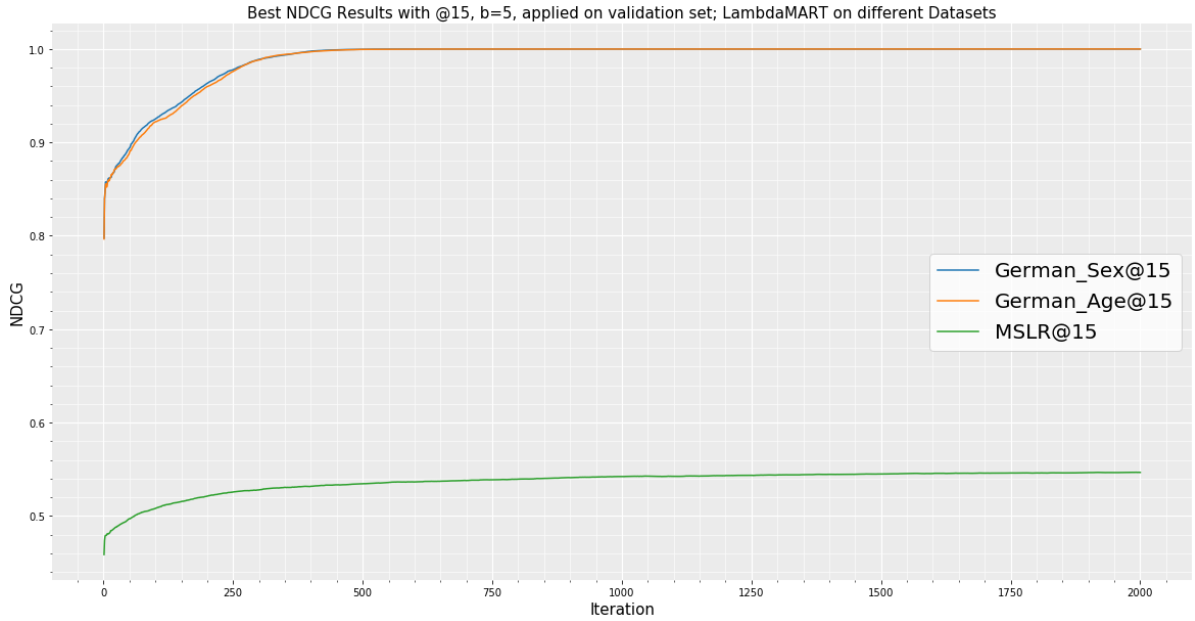

Figure 3.3: Results of LambdaMART NDCG@15 on the validation set of the three datasets

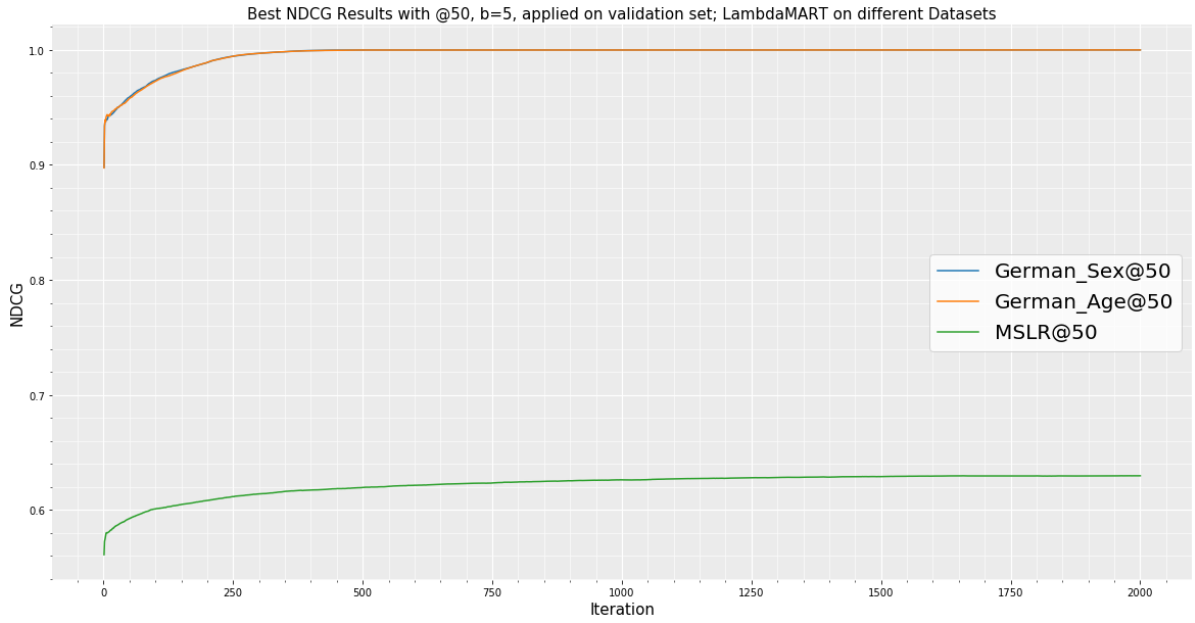

Figure 3.4: Results of LambdaMART NDCG@50 on the validation set of the three datasets

3.4.2 Hyperparameters Tuning

Now that we have seen how the LambdaMART basic performs on the datasets, we can apply our strategy to the different datasets and by changing the value of α, we can see how it performs in terms of performance and fairness.

3.4.2.1 MSLR

In Table 3.1 we can see how the algorithm performed on the dataset created from MSLR. In particular it shows how the three strategies acted with the different α. As the basic LambdaMART in case of @50 the results are slightly better than @15. While in Table 3.2 are represented the best results of the different tries and the relative plots are visible in Figures 3.5, 3.6, for *NDCG*, and Figures 3.7, 3.8, for *rND*.

Models	**alpha**	**beta**	**NDCG@15**	**rND@15**	**NDCG@50**	**rND@50**
LambdaMART	1.00	0.00	55.02	26.48	63.19	22.52
Strategy 1	0.90	0.10	54.41	23.27	62.63	19.33
Strategy 2	0.90	0.10	54.72	24.98	62.93	20.53
Strategy 3	0.90	0.10	54.86	25.27	62.95	20.97
Strategy 1	0.80	0.20	53.91	23.00	62.40	19.04
Strategy 2	0.80	0.20	54.63	24.51	62.78	20.34
Strategy 3	0.80	0.20	54.72	25.04	62.88	20.68
Strategy 1	0.70	0.30	53.59	22.76	62.20	19.00
Strategy 2	0.70	0.30	54.39	24.55	62.71	20.35
Strategy 3	0.70	0.30	54.71	24.84	62.79	20.65
Strategy 1	0.60	0.40	53.25	22.65	62.06	18.89
Strategy 2	0.60	0.40	54.44	24.37	62.65	20.22
Strategy 3	0.60	0.40	54.40	24.71	62.73	20.63
Strategy 1	0.50	0.50	52.95	22.63	61.84	18.85
Strategy 2	0.50	0.50	54.23	24.27	62.61	20.18
Strategy 3	0.50	0.50	54.48	24.78	62.73	20.55
Strategy 1	0.40	0.60	52.57	22.62	61.57	18.70
Strategy 2	0.40	0.60	54.22	24.26	62.52	20.04
Strategy 3	0.40	0.60	54.41	24.60	62.63	20.42
Strategy 1	0.30	0.70	52.13	22.50	61.35	18.60
Strategy 2	0.30	0.70	54.00	24.12	62.46	19.97
Strategy 3	0.30	0.70	54.25	24.51	62.62	20.37
Strategy 2	0.20	0.80	53.93	23.98	62.26	19.93
Strategy 3	0.20	0.80	54.06	24.47	62.43	20.23
Strategy 1	0.20	0.80	51.49	22.29	60.98	18.56
Strategy 1	0.10	0.90	50.52	22.30	60.36	18.56
Strategy 2	0.10	0.90	53.39	23.69	61.95	19.84
Strategy 3	0.10	0.90	53.63	24.24	62.27	20.32

Table 3.1: Results of our algorithm with different α and β on MSLR dataset

Models	cutoff	alpha	beta	NDCG	rND
LambdaMART	15	1.00	0.00	55.02	26.48
Strategy 1	15	0.10	0.90	50.52	22.30
Strategy 2	15	0.10	0.90	53.39	23.69
Strategy 3	15	0.10	0.90	53.63	24.24
LambdaMART	50	1.00	0.00	63.19	22.52
Strategy 1	50	0.10	0.90	60.36	18.56
Strategy 2	50	0.10	0.90	61.95	19.84
Strategy 3	50	0.20	0.80	62.43	20.23

Table 3.2: Best results of our algorithm on MSLR

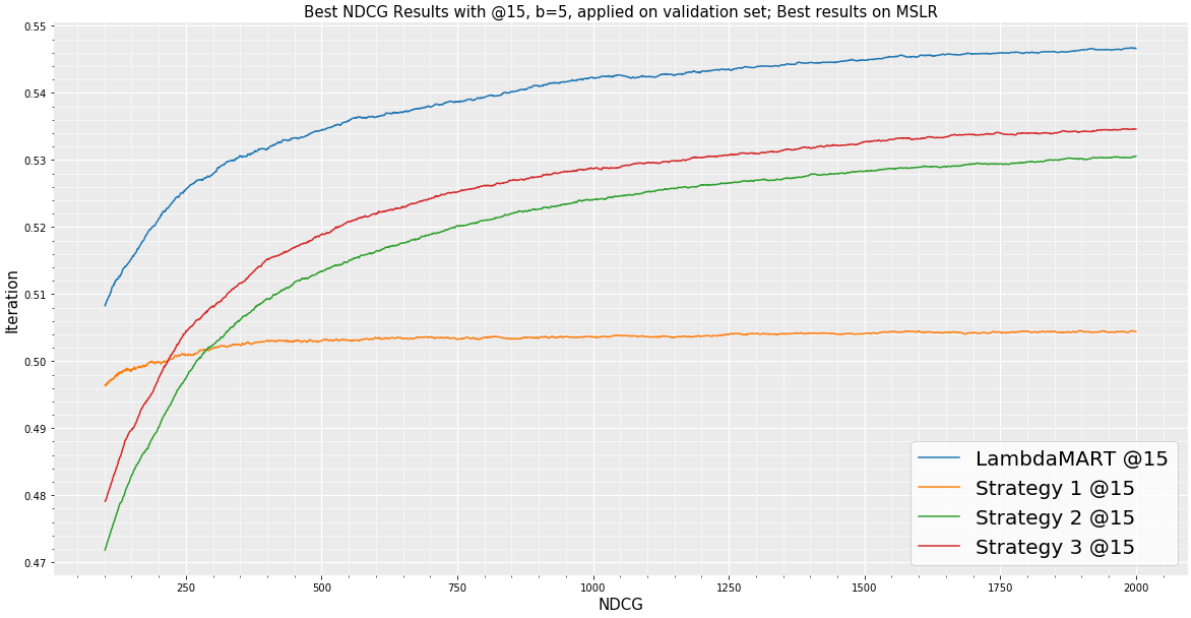

Figure 3.5: Best Results in terms of NDCG@15 on validation set of MSLR

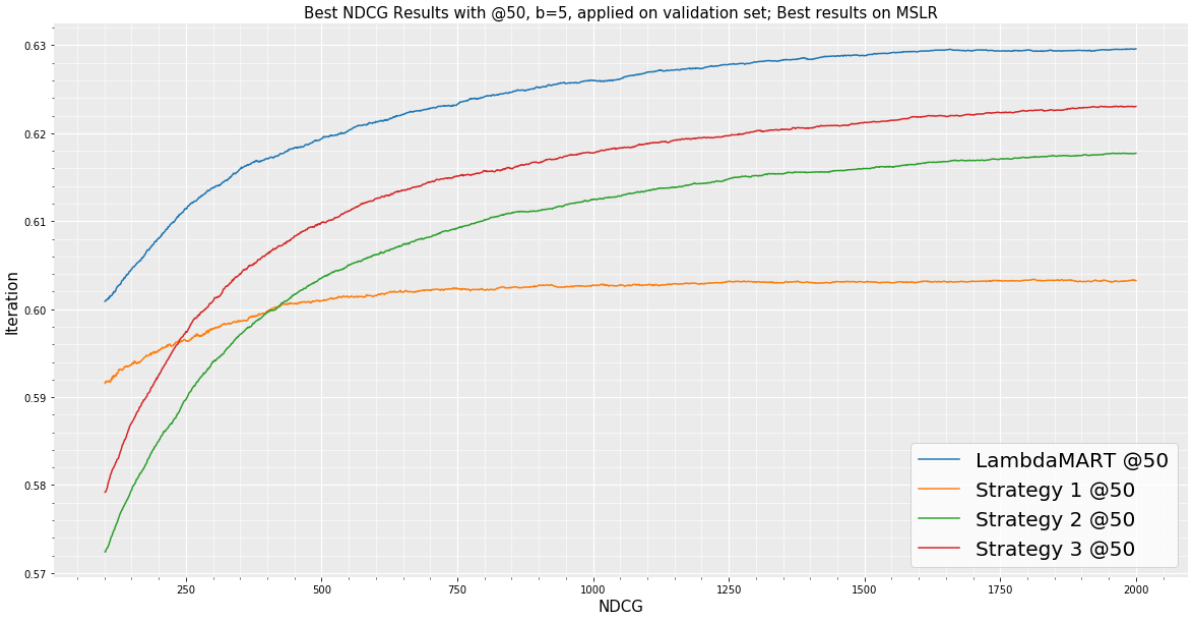

Figure 3.6: Best Results in terms of NDCG@50 on validation set of MSLR

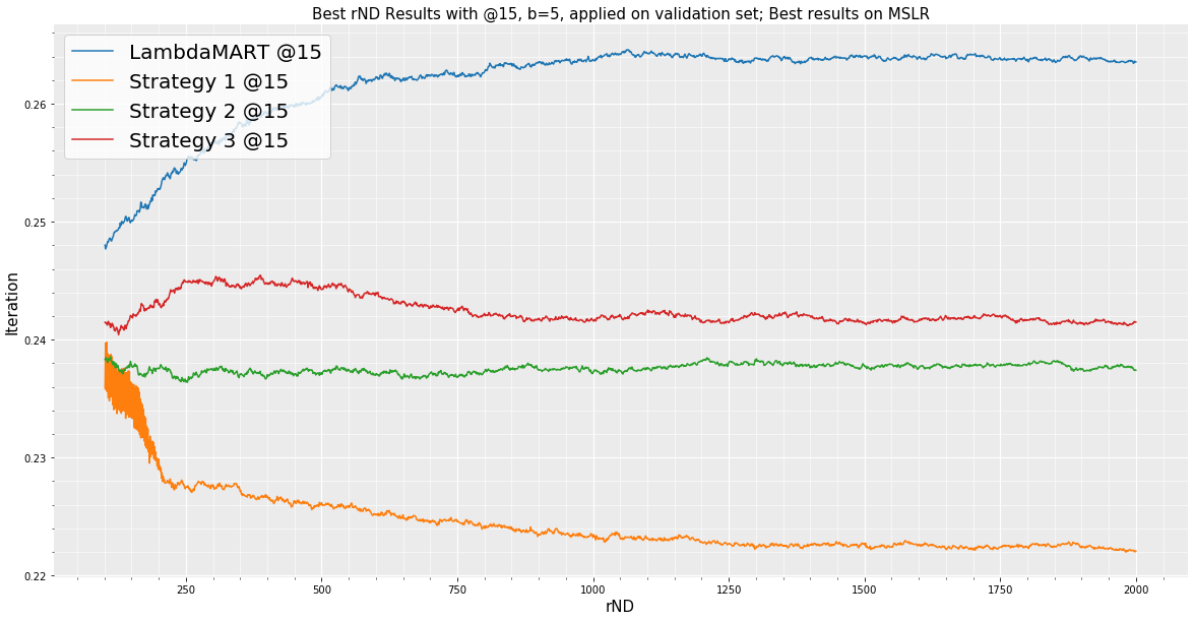

Figure 3.7: Best Results in terms of rND@15 on validation set of MSLR

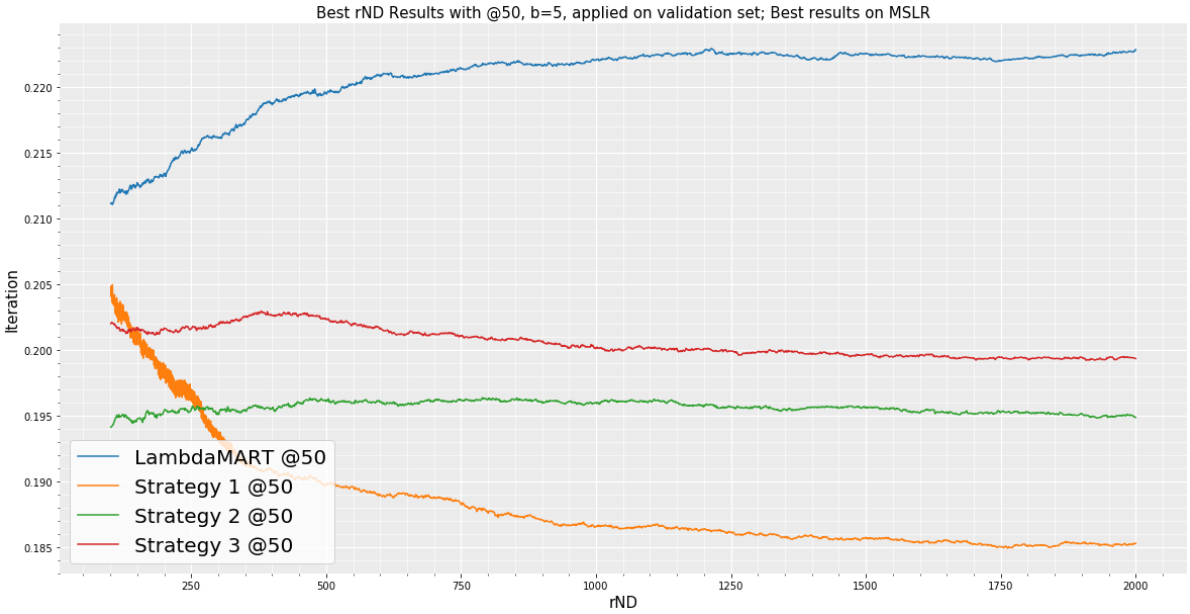

Figure 3.8: Best Results in terms of rND@50 on validation set of MSLR

3.4.2.2 German_Sex

In Table 3.3 we can see the performance of the model applied to the dataset German Credit when using the Sex, so *Female* or *Male*, as the protected variable. Here, as before, the case @50 is slightly better but the difference between the two cases is not so big. The Table 3.4 resumes the best results of the different combinations and the relative plots are in Figures 3.9, 3.10, for *NDCG* and Figures 3.11, 3.12, for *rND*.

Models	alpha	beta	NDCG@15	rND@15	NDCG@50	rND@50
LambdaMART	1.00	0.00	100.00	20.51	100.00	19.34
Strategy 1	0.90	0.10	100.00	19.56	100.00	18.52
Strategy 2	0.90	0.10	100.00	19.45	100.00	18.39
Strategy 3	0.90	0.10	100.00	19.51	100.00	18.38
Strategy 1	0.80	0.20	100.00	19.41	100.00	18.46
Strategy 2	0.80	0.20	100.00	19.43	100.00	18.29
Strategy 3	0.80	0.20	100.00	19.48	100.00	18.36
Strategy 1	0.70	0.30	100.00	19.47	100.00	18.58
Strategy 2	0.70	0.30	99.99	19.31	99.99	18.28
Strategy 3	0.70	0.30	100.00	19.40	100.00	18.36
Strategy 1	0.60	0.40	100.00	19.42	100.00	18.39
Strategy 2	0.60	0.40	99.97	19.28	99.97	18.20
Strategy 3	0.60	0.40	100.00	19.43	100.00	18.33
Strategy 1	0.50	0.50	100.00	19.43	100.00	18.40
Strategy 2	0.50	0.50	99.94	19.24	99.95	18.19
Strategy 3	0.50	0.50	100.00	19.40	100.00	18.34
Strategy 1	0.40	0.60	100.00	19.39	100.00	18.42
Strategy 2	0.40	0.60	99.91	19.10	99.92	18.15
Strategy 3	0.40	0.60	100.00	19.43	100.00	18.31
Strategy 1	0.30	0.70	100.00	19.39	100.00	18.46
Strategy 2	0.30	0.70	99.88	19.13	99.89	18.15
Strategy 3	0.30	0.70	100.00	19.40	100.00	18.29
Strategy 1	0.20	0.80	100.00	19.41	100.00	18.36
Strategy 2	0.20	0.80	99.85	19.01	99.83	18.07
Strategy 3	0.20	0.80	100.00	19.44	100.00	18.35
Strategy 1	0.10	0.90	99.99	19.35	100.00	18.38
Strategy 2	0.10	0.90	99.79	18.99	99.76	18.08
Strategy 3	0.10	0.90	100.00	19.44	100.00	18.36

Table 3.3: Results of our algorithm with different α and β on German Credit dataset that uses 'sex' as protected variable

Models	cutoff	alpha	beta	NDCG	rND
LambdaMART	15	1.00	0.00	100.00	20.51
Strategy 1	15	0.10	0.90	99.99	19.35
Strategy 2	15	0.10	0.90	99.79	18.99
Strategy 3	15	0.30	0.70	100.00	19.40
LambdaMART	50	1.00	0.00	100.00	19.34
Strategy 1	50	0.10	0.90	100.00	18.38
Strategy 2	50	0.20	0.80	99.83	18.07
Strategy 3	50	0.80	0.20	100.00	18.36

Table 3.4: Best results on German Credit dataset that uses 'Sex' as protected variable

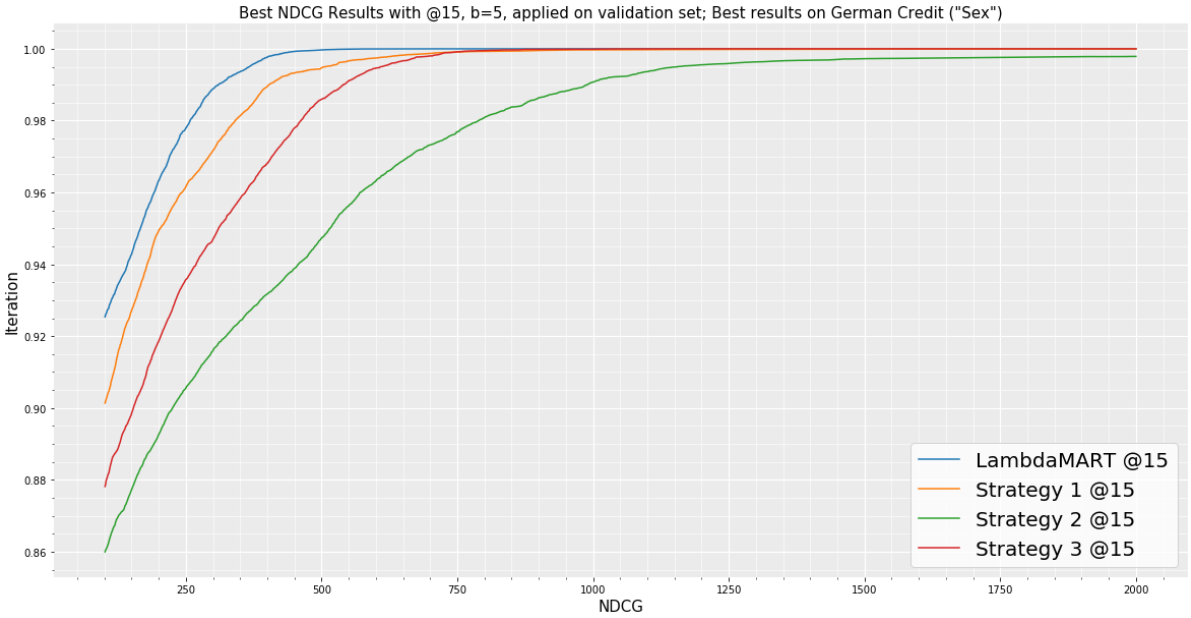

Figure 3.9: Best Results in terms of NDCG@15 on validation set of German Credit with 'Sex' as protected attribute

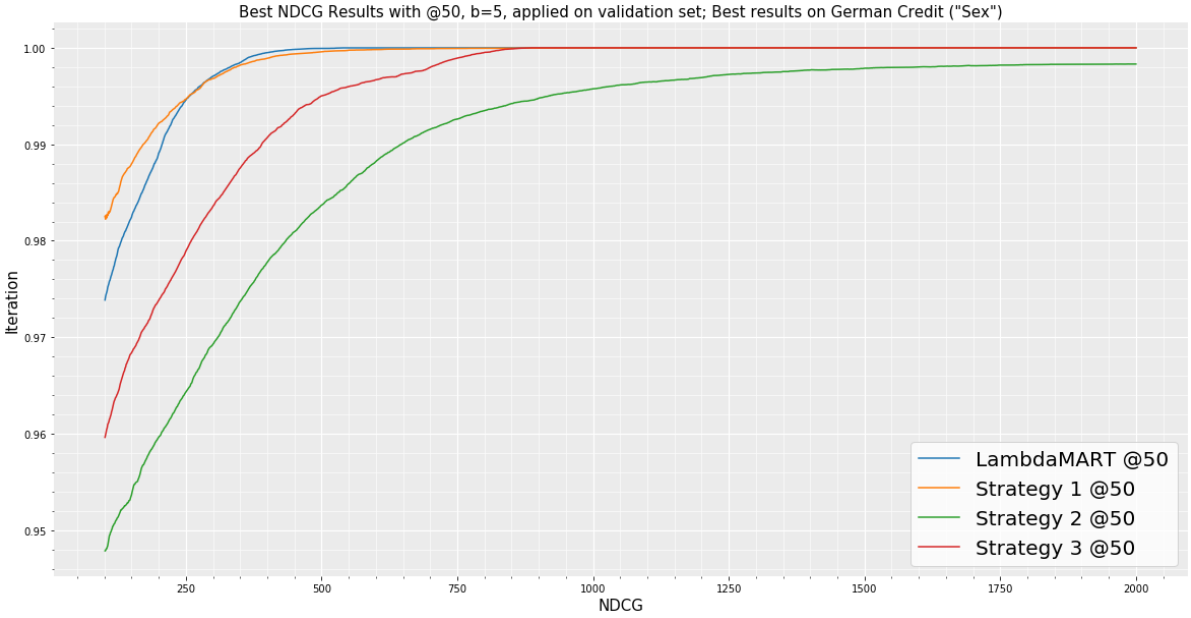

Figure 3.10: Best Results in terms of NDCG@50 on validation set of German Credit with 'Sex' as protected attribute

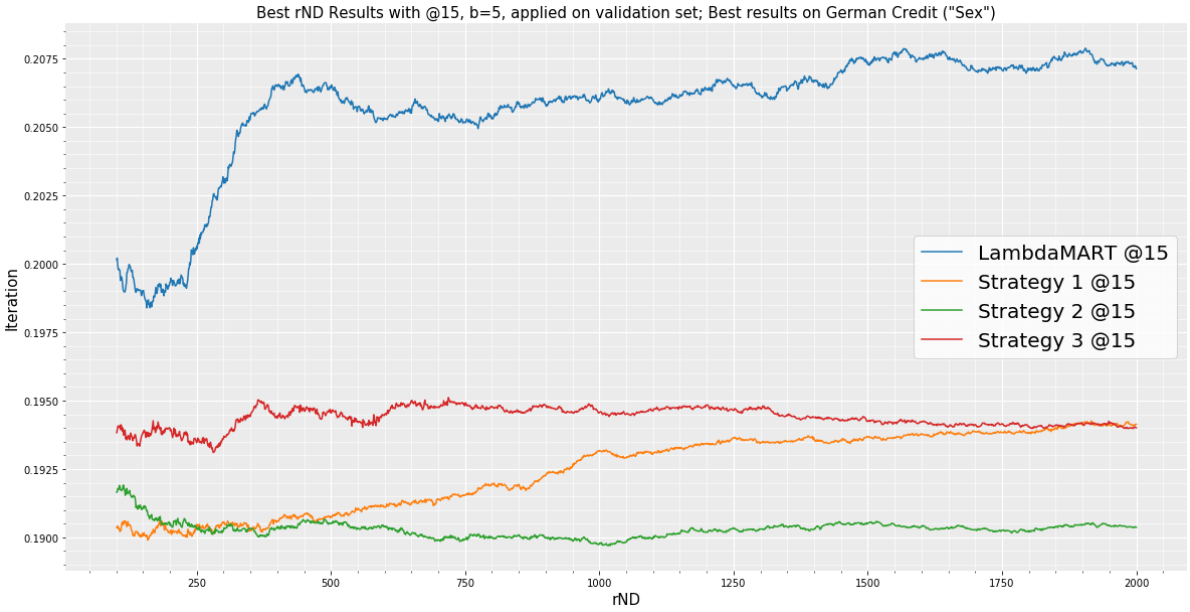

Figure 3.11: Best Results in terms of rND@15 on validation set German Credit with 'Sex' as protected attribute

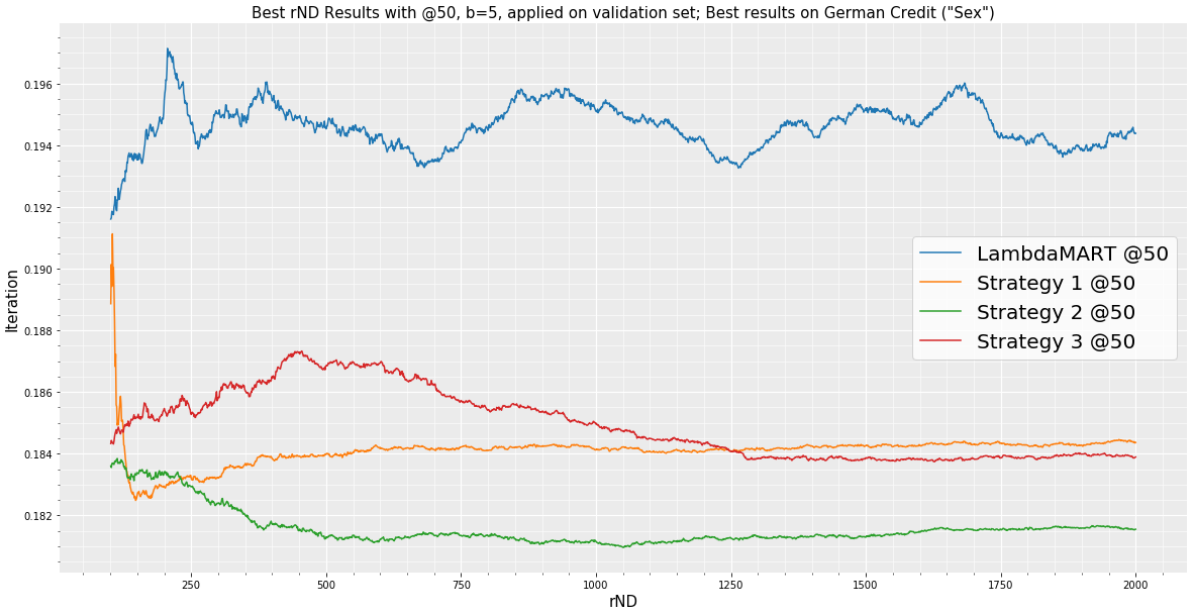

Figure 3.12: Best Results in terms of rND@50 on validation set German Credit with 'Sex' as protected attribute

3.4.2.3 German_Age

In Table 3.5 we can see how the algorithm performed when the dataset is the German Credit with the Age, less or more than 35 years, as protected variable. The results follow the same pattern seen before, so with @50 we have better results for performance and fairness, but here as in the case of *MSLR* the advantage of using @50 is more visible.

If we look at Table 3.4 we can see the best results and the relative plots are in Figures 3.13, 3.14, for *NDCG* and Figures 3.15, 3.16, for *rND*.

Models	**alpha**	**beta**	**NDCG@15**	**rND@15**	**NDCG@50**	**rND@50**
LambdaMART	1.00	0.00	100.00	29.21	100.00	24.26
Strategy 1	0.90	0.10	100.00	22.00	100.00	19.65
Strategy 2	0.90	0.10	100.00	22.00	100.00	19.57
Strategy 3	0.90	0.10	100.00	22.09	100.00	19.62
Strategy 1	0.80	0.20	100.00	22.10	100.00	19.69
Strategy 2	0.80	0.20	99.97	21.77	99.98	19.40
Strategy 3	0.80	0.20	100.00	21.96	100.00	19.58
Strategy 1	0.70	0.30	100.00	22.09	100.00	19.71
Strategy 2	0.70	0.30	99.91	21.28	99.93	19.22
Strategy 3	0.70	0.30	100.00	22.12	100.00	19.68
Strategy 1	0.60	0.40	100.00	22.04	100.00	19.74
Strategy 2	0.60	0.40	99.85	21.12	99.88	19.03
Strategy 3	0.60	0.40	100.00	22.08	100.00	19.64
Strategy 1	0.50	0.50	100.00	22.13	100.00	19.80
Strategy 2	0.50	0.50	99.79	20.92	99.83	18.91
Strategy 3	0.50	0.50	100.00	22.06	100.00	19.65
Strategy 1	0.40	0.60	100.00	22.05	100.00	19.77
Strategy 2	0.40	0.60	99.73	20.75	99.75	18.81
Strategy 3	0.40	0.60	100.00	22.14	100.00	19.62
Strategy 1	0.30	0.70	99.96	21.79	100.00	19.80
Strategy 2	0.30	0.70	99.64	20.60	99.67	18.68
Strategy 3	0.30	0.70	100.00	22.00	100.00	19.64
Strategy 1	0.20	0.80	99.89	21.44	99.98	19.72
Strategy 2	0.20	0.80	99.54	20.53	99.58	18.61
Strategy 3	0.20	0.80	100.00	22.04	100.00	19.58
Strategy 1	0.10	0.90	99.79	20.96	99.88	19.23
Strategy 2	0.10	0.90	99.42	20.36	99.44	18.59
Strategy 3	0.10	0.90	100.00	22.01	100.00	19.64

Table 3.5: Results of our algorithm with different α and β on German Credit dataset that uses 'Age' as protected variable

Models	cutoff	alpha	beta	NDCG	rND
LambdaMART	15	1.00	0.00	100.00	29.21
Strategy 1	15	0.10	0.90	99.79	20.96
Strategy 2	15	0.10	0.90	99.42	20.36
Strategy 3	15	0.80	0.20	100.00	21.96
LambdaMART	50	1.00	0.00	100.00	24.26
Strategy 1	50	0.10	0.90	99.88	19.23
Strategy 2	50	0.10	0.90	99.44	18.59
Strategy 3	50	0.90	0.10	100.00	19.62

Table 3.6: Results of our algorithm with different α and β on German Credit dataset that uses 'Age' as protected variable

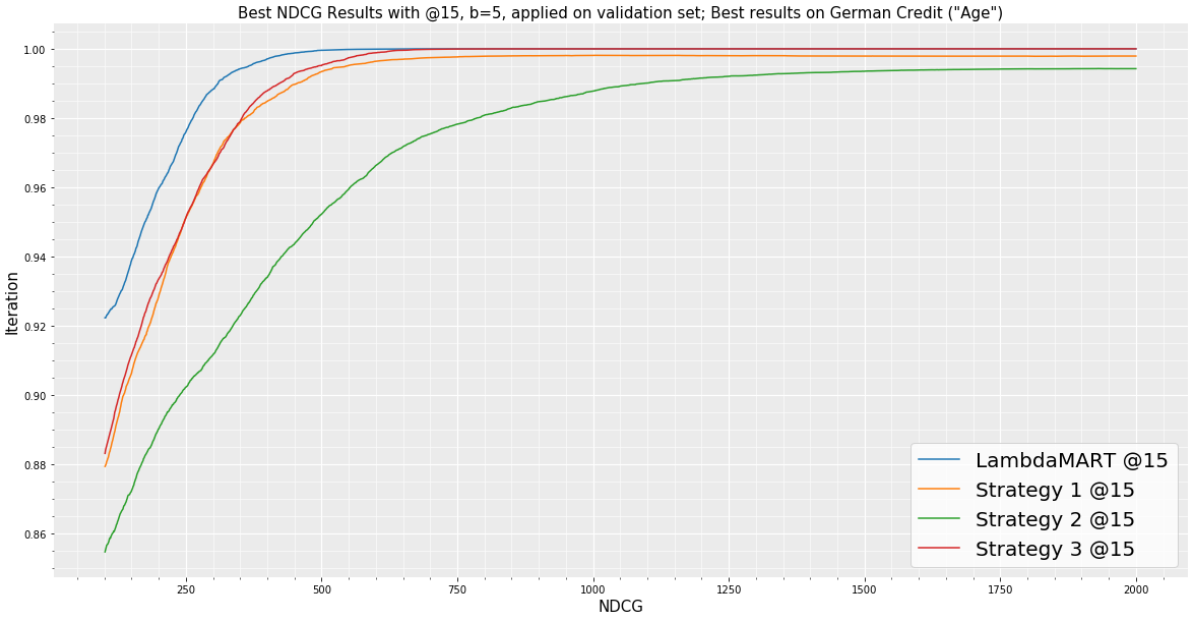

Figure 3.13: Best Results in terms of NDCG@15 on validation set of German Credit with 'Age' as protected attribute

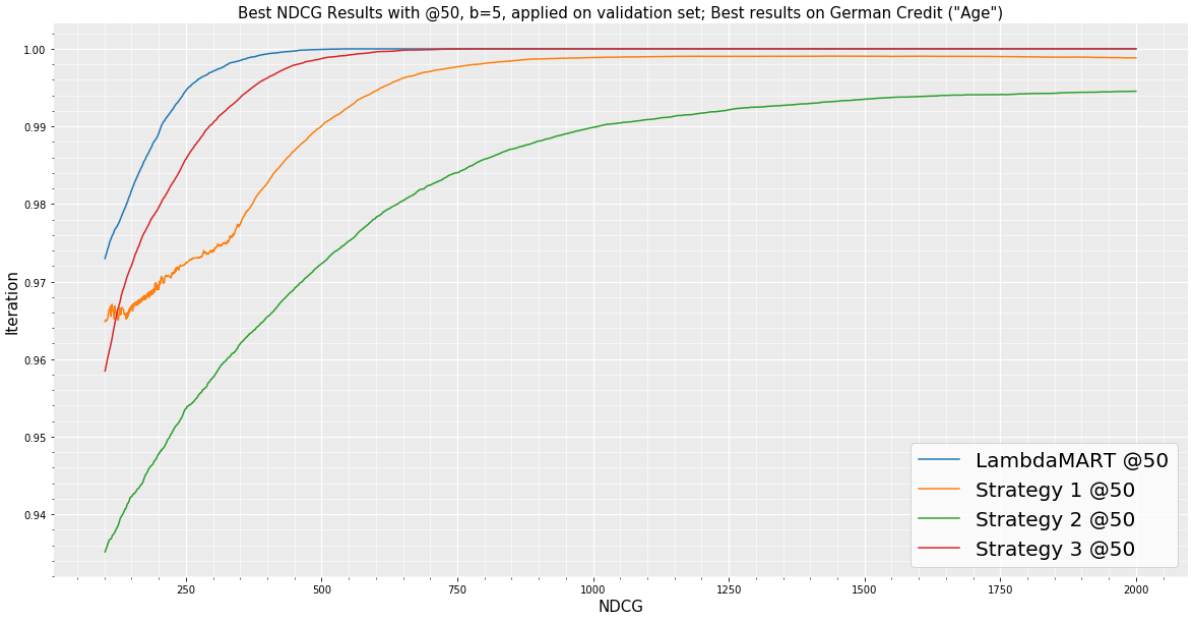

Figure 3.14: Best Results in terms of NDCG@50 on validation set of German Credit with 'Age' as protected attribute

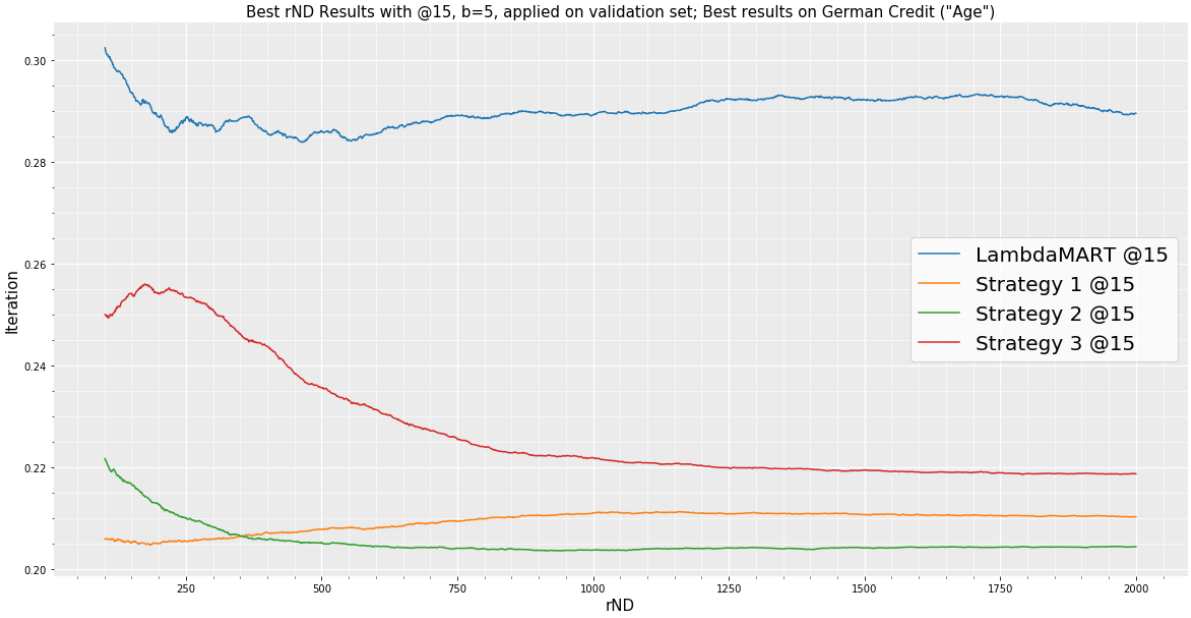

Figure 3.15: Best Results in terms of rND@15 on validation set German Credit with 'Age' as protected attribute

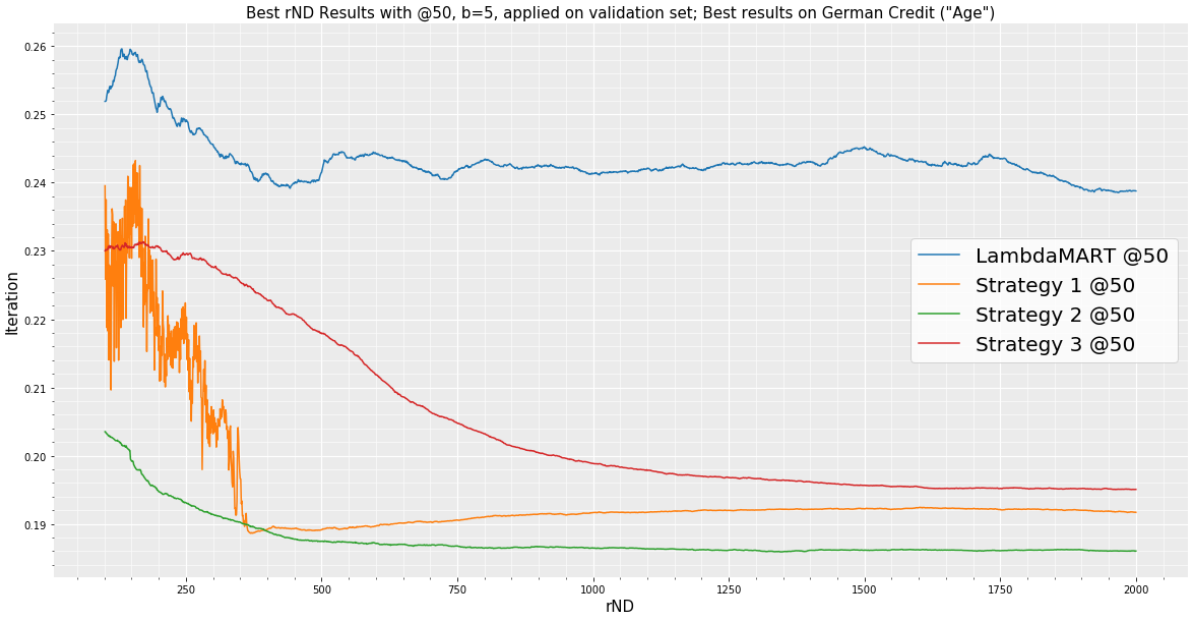

Figure 3.16: Best Results in terms of rND@50 on validation set German Credit with 'Age' as protected attribute

3.5 Evaluations

By examining all the experiments conducted we can draw some conclusions. According to Tables 3.2, 3.4 and 3.6, all the three strategies produce fairer results compared to LambdaMART; although the performance is not as strong, it is still acceptable.

When focusing on the MSLR dataset, the First Strategy provides the most favorable outcomes regarding fairness. Despite expectations, the Second Strategy, designed specifically for fairness, could perform better. This discrepancy may be due to the labels of the items being limited to a range of 0 to 4, making it challenging to achieve a satisfactory ranking that meets the criteria of the Second Strategy approach.
When examining the algorithm's performance with various strategies on the German Credit dataset, we find that the Second Strategy excels in fairness, the Third in performance, and the First strikes a balance between both aspects. The results are conceptually identical for both datasets of German Credit, that are based on 'Sex' and 'Age'. Here, the results are expected because they respect the idea behind the strategies, and this is also thanks to the fact that the relevance labels are 0 or 1 and then this allows to have good results and execution of the formulas.

Chapter 4

Conclusion

In this work, we addressed the problem of Fairness described in Section 2.3 and, in particularly, its application in learning to rank. To solve this problem, we adopted a fairness measure based on the concept of statistical parity, i.e., the scope is to have the two groups (protected and non protected) in similar percentages in the final ranking. This measure, *rND*, was combined with the *NDCG* given by the algorithm LambdaMART.
The objective is then to create an algorithm based on LambdaMART that optimises both performance and fairness. In order to do so, we developed three different strategies, each focused on a specific part of the problem, described in Section 3.2. Then, these strategies are tested on two different real datasets that are used in learning to rank, described in Section 2.4, i.e., MSLR, which is a dataset created by Microsoft that contains different queries with the relative documents that are related to them and the German Credit, which contains individuals that asked a Credit. Here, it is possible to find the personal information, such as age, sex, scope of the credit, about the people, and since it is a smaller dataset than MSLR we enlarged it by creating synthetic data by using the one already present in it. The three strategies differ in how we arrange the items in the produced rank.
The first one choose if two elements must be swapped, following the idea of the LambdaMART, Section 2.2.4.4, if the difference in terms of *rND* given by the swap is positive or not, so if we do the swap, it has a better fairness. This strategy applied to the datasets gave good results, and it respected the fact that it has the scope to optimise fairness and performance.
The second strategy is more complicated because it has the objective to optimise the combination of *NDCG* and *rND* by maximising *NDCG* while *rND* is at its minimum. That is because, with lower values, the algorithm is more fair, so as evident the main scope here is to give more importance to fairness than performance. By applying the method to the datasets, we evaluated that this goal was respected and achieved.
The third strategy is similar to the second, but here, the scope is to minimise *rND* while the *NDCG* is at its maximum. That is because with more significant values of NDCG, the performance is better, and the *rND* should be minimised. Here, instead, we give more importance to the performance of the model than the fairness and by looking at the results in Section 3.4, we can see that this condition is respected.

In conclusion our research has successfully achieved its main objective. As the positive results demonstrate, we have developed an algorithm that optimises performance and Fairness. This algorithm has the potential to significantly impact the field of machine learning and Fairness in algorithms.

Bibliography

[1] Abolfazl Asudeh et al. "Designing Fair Ranking Schemes". In: *Proceedings of the 2019 International Conference on Management of Data*. SIGMOD/PODS '19. ACM, June 2019. URL: http://dx.doi.org/10.1145/3299869.3300079.

[2] Ricardo Baeza-Yates. "Bias on the web". In: *Commun. ACM* 61.6 (2018), 54–61. ISSN: 0001-0782. URL: https://doi.org/10.1145/3209581.

[3] Ricardo Baeza-Yates et al. "Modern Information Retrieval". In: (July 1999).

[4] Eric B. Baum and Frank Wilczek. "Supervised Learning of Probability Distributions by Neural Networks". In: *Neural Information Processing Systems*. Ed. by Dana Z. Anderson. New York: American Institute of Physics, 1987.

[5] Alex Beutel et al. *Fairness in Recommendation Ranking through Pairwise Comparisons*. 2019. arXiv: 1903.00780 [cs.CY].

[6] Asia Biega, Krishna Gummadi, and Gerhard Weikum. "Equity of Attention: Amortizing Individual Fairness in Rankings". In: (May 2018). DOI: 10.1145/3209978.3210063.

[7] Chris Burges et al. "Learning to rank using gradient descent". In: *Proceedings of the 22nd international conference on Machine learning*. ICML '05. New York, NY, USA: ACM, 2005, pp. 89–96. ISBN: 1-59593-180-5. URL: http://doi.acm.org/10.1145/1102351.1102363.

[8] Christopher J. C. Burges, Robert J. Ragno, and Quoc V. Le. "Learning to Rank with Nonsmooth Cost Functions". In: *Neural Information Processing Systems*. 2006. URL: https://api.semanticscholar.org/CorpusID:8604596.

[9] Christopher JC Burges. "From ranknet to lambdarank to lambdamart: An overview". In: *Learning* 11.23-581 (2010), p. 81.

[10] Zhe Cao et al. "Learning to rank: from pairwise approach to listwise approach". In: *Proceedings of the 24th International Conference on Machine Learning*. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, 129–136. ISBN: 9781595937933. URL: https://doi.org/10.1145/1273496.1273513.

[11] L. Elisa Celis, Anay Mehrotra, and Nisheeth K. Vishnoi. *Interventions for Ranking in the Presence of Implicit Bias*. 2020. arXiv: 2001.08767 [cs.CY].

[12] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. *Ranking with Fairness Constraints*. 2018. arXiv: 1704.06840 [cs.DS].

[13] W. W. Cohen, R. E. Schapire, and Y. Singer. "Learning to Order Things". In: *Journal of Artificial Intelligence Research* 10 (May 1999), 243–270. ISSN: 1076-9757. DOI: 10.1613/jair.587. URL: http://dx.doi.org/10.1613/jair.587.

[14] David Cossock and Tong Zhang. *Subset ranking using regression*. Jan. 2006, pp. 605–619. URL: https://doi.org/10.1007/11776420_44.

[15] Koby Crammer and Yoram Singer. "Pranking with Ranking". In: *Advances in Neural Information Processing Systems*. Ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. MIT Press, 2001. URL: https://proceedings.neurips.cc/paper_files/paper/2001/file/5531a5834816222280f20d1ef9e95f69-Paper.pdf.

[16] Cynthia Dwork et al. *Fairness Through Awareness*. 2011. arXiv: 1104.3913 [cs.CC].

[17] Batya Friedman and Helen Nissenbaum. "Bias in computer systems". In: *ACM Trans. Inf. Syst.* 14.3 (1996), 330–347. ISSN: 1046-8188. URL: https://doi.org/10.1145/230538.230561.

[18] J.H. Friedman. "Greedy function approximation: A gradient boosting machine". In: *Annals of Statistics* 29 (2001).

[19] Norbert Fuhr. "Optimum polynomial retrieval functions based on the probability ranking principle". In: *ACM transactions on office information systems* 7.3 (July 1989), pp. 183–204. URL: https://doi.org/10.1145/65943.65944.

[20] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. O'Reilly Media, Inc., 2019. ISBN: 1492032646.

[21] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. "Fairness-Aware Ranking in Search & Recommendation Systems with Application to LinkedIn Talent Search". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19. ACM, July 2019. URL: http://dx.doi.org/10.1145/3292500.3330691.

[22] Hans Hofmann. *Statlog (German Credit Data)*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77. 1994.

[23] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques". In: *ACM Trans. Inf. Syst.* 20.4 (Oct. 2002), 422–446. ISSN: 1046-8188. URL: https://doi.org/10.1145/582415.582418.

[24] Kalervo Järvelin and Jaana Kekäläinen. "IR evaluation methods for retrieving highly relevant documents". In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '00. New York, NY, USA: Association for Computing Machinery, 2000, 41–48. ISBN: 1581132263. URL: https://doi.org/10.1145/345508.345545.

[25] Jon Kleinberg and Manish Raghavan. *Selection Problems in the Presence of Implicit Bias*. 2018. arXiv: 1801.03533 [cs.CY].

[26] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. *iFair: Learning Individually Fair Data Representations for Algorithmic Decision Making*. 2019. arXiv: 1806.01059 [cs.LG].

[27] Ping Li, Christopher J. C. Burges, and Qiang Wu. "McRank: Learning to Rank Using Multiple Classification and Gradient Boosting." In: *NIPS*. Ed. by John C. Platt et al. Curran Associates, Inc., 2007, pp. 897–904. URL: http://dblp.uni-trier.de/db/conf/nips/nips2007.html#LiBW07.

[28] Tie-Yan Liu. *Learning to rank for information retrieval*. Jan. 2011. URL: https://doi.org/10.1007/978-3-642-14267-3.

[29] Wei-Yin Loh. "Classification and Regression Trees". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1 (Jan. 2011), pp. 14 –23. DOI: 10.1002/widm.8.

[30] R.D. Luce. *Individual Choice Behavior: A Theoretical Analysis*. Dover Books on Mathematics. Dover Publications, 2012. ISBN: 9780486153391. URL: https://books.google.it/books?id=ERQsKkPiKkkC.

[31] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. 2nd ed. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press. ISBN: 978-0-262-03940-6.

[32] Ramesh Nallapati. "Discriminative models for information retrieval". In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '04. New York, NY, USA: Association for Computing Machinery, 2004, 64–71. ISBN: 1581138814. URL: https://doi.org/10.1145/1008992.1009006.

[33] R. L. Plackett. "The Analysis of Permutations". In: *Journal of the Royal Statistical Society Series C: Applied Statistics* 24.2 (Dec. 2018), pp. 193–202. ISSN: 0035-9254. eprint: https://academic.oup.com/jrsssc/article-pdf/24/2/193/48619663/jrsssc_24_2_193.pdf. URL: https://doi.org/10.2307/2346567.

[34] Tao Qin and Tie-Yan Liu. "Introducing LETOR 4.0 Datasets". In: *CoRR* abs/1306.2597 (2013). URL: http://arxiv.org/abs/1306.2597.

[35] Stephen Robertson and Hugo Zaragoza. "On rank-based effectiveness measures and optimization." In: *Inf. Retr.* 10.3 (2007), pp. 321–339. URL: http://dblp.uni-trier.de/db/journals/ir/ir10.html#RobertsonZ07.

[36] Stephen Robertson and Hugo Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Found. Trends Inf. Retr.* 3.4 (2009), 333–389. ISSN: 1554-0669. URL: https://doi.org/10.1561/1500000019.

[37] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0136042597.

[38] Yu Shi et al. *lightgbm: Light Gradient Boosting Machine*. R package version 4.3.0.99. 2024. URL: https://github.com/Microsoft/LightGBM.

[39] Ashudeep Singh and Thorsten Joachims. "Fairness of Exposure in Rankings". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, 2219–2228. ISBN: 9781450355520. URL: https://doi.org/10.1145/3219819.3220088.

[40] Ashudeep Singh and Thorsten Joachims. "Policy Learning for Fairness in Ranking". In: *CoRR* abs/1902.04056 (2019). URL: http://arxiv.org/abs/1902.04056.

[41] Julia Stoyanovich, Jannik H. Reimer, and Kai Yang. *FairRank: Quantifying and Mitigating Bias in Rankings*. 2024. URL: https://github.com/DataResponsibly/FairRank.

[42] Julia Stoyanovich, Ke Yang, and H. V. Jagadish. "Online Set Selection with Fairness and Diversity Constraints". In: *International Conference on Extending Database Technology*. 2018. URL: https://api.semanticscholar.org/CorpusID:135391.

[43] Michael Taylor et al. "SoftRank: optimizing non-smooth rank metrics". In: *Proceedings of the 2008 International Conference on Web Search and Data Mining*. WSDM '08. New York, NY, USA: Association for Computing Machinery, 2008, 77–86. ISBN: 9781595939272. URL: https://doi.org/10.1145/1341531.1341544.

[44] Vladimir N. Vapnik. *The nature of statistical learning theory*. Jan. 1995. URL: https://doi.org/10.1007/978-1-4757-2440-0.

[45] Ali Vardasbi, Fatemeh Sarvi, and Maarten de Rijke. "Probabilistic Permutation Graph Search: Black-Box Optimization for Fairness in Ranking". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. New York, NY, USA: Association for Computing Machinery, 2022, 715–725. ISBN: 9781450387323. DOI: 10.1145/3477495.3532045. URL: https://doi.org/10.1145/3477495.3532045.

[46] Qiang Wu et al. "Adapting boosting for information retrieval measures." In: *Inf. Retr.* 13.3 (2010), pp. 254–270. URL: http://dblp.uni-trier.de/db/journals/ir/ir13.html#WuBSG10.

[47] Fen Xia et al. "Listwise approach to learning to rank: theory and algorithm". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, 1192–1199. ISBN: 9781605582054. DOI: 10.1145/1390156.1390306. URL: https://doi.org/10.1145/1390156.1390306.

[48] Ke Yang, Vasilis Gkatzelis, and Julia Stoyanovich. "Balanced Ranking with Diversity Constraints". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 6035–6042. URL: https://doi.org/10.24963/ijcai.2019/836.

[49] Ke Yang, Joshua R. Loftus, and Julia Stoyanovich. "Causal intersectionality and fair ranking". In: *2nd Symposium on Foundations of Responsible Computing, FORC 2021*. Ed. by Katrina Ligett and Swati Gupta. Leibniz International Proceedings

in Informatics, LIPIcs. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2021. DOI: 10.4230/LIPIcs.FORC.2021.7.

[50] Ke Yang and Julia Stoyanovich. "Measuring Fairness in Ranked Outputs". In: *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*. SSDBM '17. New York, NY, USA: Association for Computing Machinery, 2017. ISBN: 9781450352826. URL: https://doi.org/10.1145/3085504.3085526.

[51] Meike Zehlike, Philipp Hacker, and Emil Wiedemann. "Matching code and law: achieving algorithmic fairness with optimal transport". In: *Data Mining and Knowledge Discovery* 34 (Jan. 2020). DOI: 10.1007/s10618-019-00658-8.

[52] Meike Zehlike et al. "FA*IR: A Fair Top-k Ranking Algorithm". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. ACM, Nov. 2017. URL: http://dx.doi.org/10.1145/3132847.3132938.

[53] Meike Zehlike et al. "Fair Top-k Ranking with multiple protected groups". In: *Information Processing & Management* 59.1 (2022), p. 102707. ISSN: 0306-4573. URL: https://www.sciencedirect.com/science/article/pii/S0306457321001916.

[54] Meike Zehlike et al. "FairSearch: A Tool For Fairness in Ranked Search Results". In: *Companion Proceedings of the Web Conference 2020*. WWW '20. ACM, 2020. URL: http://dx.doi.org/10.1145/3366424.3383534.