The Effects of Debiasing Methods on the Fairness and Accuracy of Recommender Systems

Author: Filip Čajági[1]
Supervisor: Masoud Mansoury[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Filip Čajági
Final project course: CSE3000 Research Project
Thesis committee: Masoud Mansoury, Nergis Tömen

Abstract

Recommender systems leverage user interactions to predict their preferences and deliver personalized recommendations. Recent years have seen a great increase in their widespread usage in online areas, such as social media, e-commerce and even job applications. However, due to how these systems collect and learn from data, they are vulnerable to various biases, such as popularity bias, which also raises the question of their fairness for both users and providers. Researchers in the area have tried addressing the issue with various debiasing and fairness intervention methods, but these are often studied in separate strands of research, and the trade-off between fairness and accuracy is rarely explicitly evaluated when it comes to debiasing methods.

In this project, we replicate three state-of-the-art debiasing methods and analyze their impact on the fairness and accuracy of recommender models, particularly the trade-off between the two and how it can be controlled using hyper-parameters. We find that while the impact heavily depends on the method and dataset used, in many cases significant improvements can be made to fairness with little to no decrease in accuracy, using the right configuration of hyper-parameters.

1 Introduction

In recent years, recommender systems have become an integral part of most people's everyday lives - video/movie recommendations, social media posts, e-shop listings, even dating apps - whenever we interact with an interface which serves us recommendations, chances are it uses some kind of a recommender system. They leverage users' past interactions to predict their preferences and deliver personalized recommendations, in an attempt to improve the user experience. However, due to how these systems collect and learn from data, they often suffer from various kinds of biases, such as popularity bias, which is one of the most pervasive ones. It causes few items to get over-recommended, while most other items suffer from under-exposure. Furthermore, due to the feedback loop between the system's recommendations and user interactions, biases can reinforce and amplify themselves if left unchecked. The existence of these biases also raises the question of fairness - whether different user or item groups experience recommendations of similar quality and whether all items are exposed equitably.

Researchers in the area have tried addressing these issues with various debiasing methods and fairness intervention techniques. Chen et al. (2020) offer a fairly extensive survey of biases and debiasing techniques in the field in [3], while Zhao et al. (2023) provide a similar survey for fairness in [15]. However, debiasing methods are often studied in terms of improving accuracy and performance, while fairness is often studied in a separate strand of research. This leaves the effects of debiasing methods on the fairness of recommender systems somewhat underexplored, which is what this project aims to address.

In this project, we reproduce three state-of-the-art debiasing methods, and then look at how we can define and measure their accuracy, debiasing effectiveness, and most importantly their effect on fairness, using various metrics, so that we can answer the research question: **"Do debiasing methods contribute to mitigating the fairness issue, or do they primarily improve accuracy without addressing fairness directly?"** We will further investigate it in two sub-questions:

- **RQ1:** How do debiasing methods affect the trade-off between fairness and accuracy in recommender systems?
- **RQ2:** Can varying the hyper-parameters of debiasing methods be used to control their effect on the performance of recommender systems and the trade-off between fairness and accuracy in recommender systems?

The paper will be structured in the following way. Section 2 offers a more detailed look into the background of the field of recommender systems, their biases, and commonly used debiasing methods. Section 3 details the methodology used for the experiments, including the chosen datasets, algorithms, metrics, and the experiment setup. Section 4 then reports on the results of the experiments and analyzes them to answer the two research questions. Section 5 reflects on the ethical aspects of the research and its reproducibility. Finally, Section 7 draws conclusions based on all the previous sections and proposes recommendations for further research in the area.

2 Background

The field of recommender systems and algorithms has become quite broad in the recent years, both in terms of the domains where they are employed, as well as the specific underlying methods used to perform recommendation. The idea of learning from users' past interactions to predict what they might interact with next sounds relatively simple and similar to other branches of machine learning, but the specifics can vary quite a bit based on the problem and the data available. For example, there might be data available for users and items, such as their gender, age or genre, or there might only be records of their interactions, with users and items being simply represented by their IDs. Similarly for interactions, they might be explicit, such as a rating from 1 to 5, or they might be implicit, such as clicking on a recommended item. These differences give rise to a variety of algorithms and models used to perform the recommendation task. As for biases, there have also been multiple types of them identified, but the terminology used in the field is not always consistent [3]. This can result in confusion as to what exactly is being addressed.

This section outlines the commonly used types of recommender systems, along with some specific models, and also defines the different types of biases we identified and some debiasing methods used to address them.

2.1 Recommender models

Based on the data that is used and how it is used, we can group recommender models into a few distinct general categories.

- **Collaborative filtering-based models** - These models rely purely on the historical interaction data, such as clicks, purchases, or ratings, to identify patterns of behavior shared among users or items. However, even with this restriction, there are still a few different approaches. Arguably some of the most simple and traditional, but often still effective, are memory-based neighborhood models, usually referred to as ItemKNN or UserKNN. They calculate the similarity of users or items based on their interactions to find the k neareast neighbors and then recommend either items interacted with by similar users, or items similar to the ones they interacted with [1]. A somewhat more nuanced approach is based on matrix factorization. These models calculate lower-dimensional embeddings of latent features for users and items based on their interactions and then generate recommendations by matching these embeddings [5]. Regardless of the specific implementation, collaborative filtering-based models perform well when interaction data is abundant but struggle with the cold-start problem - when new users or items have limited or no interaction data.
- **Content-based models** - These systems make recommendations based on additional features of the items and users. For example, a user who has previously interacted with a science-fiction book might be recommended other items with similar content attributes, such as genre, keywords, or metadata. Unlike collaborative filtering, content-based models do not rely solely on the preferences of other users, making them more resilient to cold-start issues for users. However, they require additional data about users and items, and they may also suffer from overspecialization, repeatedly recommending similar items without exploring diverse options.
- **Knowledge aware (hybrid) models** - These combine multiple sources of data, such as domain knowledge, user-item interactions, and side information (e.g., demographics or knowledge graphs), to enhance recommendations [11]. Hybrid models are designed to overcome the limitations of pure collaborative or content-based methods. For instance, a system may incorporate both user preference patterns and item content features. Knowledge-aware models are particularly useful in domains with rich semantic relationships or in areas where explainability is important. However, they often require extra domain-specific information, such as a knowledge graph, and the quality of the recommendations can vary based on this side information.

2.2 Biases

We can also define several types of biases, based on how and why they arise and propagate themselves. This list is not exhaustive, but these are some of the most pervasive biases that are also most often addressed by debiasing methods.

- **Popularity bias** - It is natural in most domains that some items are more popular than others, and that is not necessarily problematic. However, this usually results in popular items having more interaction data, which in many recommender systems makes these items over-recommended even more than their popularity warrants. This can lead to even more data for these items, amplifying the popularity bias over time through this feedback loop, also known as the Matthew effect [8]. This bias can reduce the diversity of recommendations and limit exposure to niche or long-tail items, which may be equally relevant or of higher interest to some users.
- **Selection bias** - This bias arises from the users' choice when giving explicit interaction feedback, such as ratings. In other words, users choose which items they want to rate, instead of rating all items they interacted with. This results in the data being Missing Not At Random (MNAR), and not being fully representative of the users' real preferences.
- **Position bias** - Users are more likely to interact with items that are displayed in more prominent positions, for example, at the top of a list, regardless of their actual relevance. This creates a confounding factor where interaction data reflects not only user preferences but also the item's position in the interface, making it difficult to learn true relevance.
- **Exposure bias** - Somewhat similar to selection bias, exposure bias arises as users are often only exposed to a small subset of all the items and most recommender models do not distinguish between missing interactions and negative interactions. This means that if a user was exposed to an item but did not interact with it, it is usually treated the same as if the user did not see the item at all.
- **Conformity bias** - Users often tend to behave similarly to others in a group based, for example, based on current social trends, and these interactions may not reflect the user's true preferences. In a way, this bias can be considered the user-side version of the popularity bias, as it contributes to propagating it as well.

2.3 Debiasing methods

The most commonly addressed biases when it comes to debiasing methods seem to be popularity bias and selection bias, possibly because they are the most easily observable. Since these biases can distort both learning and evaluation, debiasing can be applied to increase accuracy in certain scenarios.

One of the earliest proposed methods is based on inverse propensity weighting/scoring **(IPS)** [7]. The propensity of an interaction is essentially its probability of being observed, which, in a simple case, can be estimated by the item's popularity. Using the inverse of this propensity, the impact of each interaction during the training process can be weighed, with unlikely interactions having increased impact and likely interactions (i.e. interactions with popular items) having decreased impact. However, estimating propensity scores is problematic and not always accurate [3]. Depending on the specific implementation and the weighing process, propensity scoring can be used to address both selection bias and popularity bias, but it can also have a detrimental effect to the accuracy of the model.

More recent debiasing methods seem to be increasingly using causal and counterfactual reasoning in order to address popularity bias directly. Zhang et al. (2021) observe that item popularity acts as a confounder between exposure and user feedback: an item's high popularity both makes it more likely to be shown and more likely to be clicked, regardless of user interest [12]. Their method, called Popularity-bias Deconfounding and Adjusting **(PDA)**, first deconfounds popularity during training and then adjusts recommendations based on popularity at inference. They claim that "not all biases in the data are bad – some items demonstrate higher popularity because of their better intrinsic quality" [12]. The idea behind the method is that the model learns users' genuine interests without overfitting popular items, and then still promotes inherently good items when generating recommendations.

Wei et al. (2021) propose a model-agnostic counterfactual reasoning **(MACR)** approach to eliminate popularity bias [8]. During training, MACR uses multi-task learning to simultaneously train a main recommender model, based on user-item interactions (such as MF), and two supporting modules based solely on users and items, respectively. It then calculates a counterfactual score by subtracting the popularity-driven supporting modules from the main one, addressing both the popularity and conformity biases. This approach is model-agnostic – it can be applied on top of any base recommender – and has been shown to boost accuracy on long-tail items while maintaining overall performance [8].

Other approaches have also been proposed, but they still often use similar ideas of propensity and causal or counterfactual reasoning.

3 Methodology

In order to evaluate the effects of debiasing methods on the fairness and accuracy of recommender systems, we reproduce several state-of-the-art methods from papers described in Section 2.3 and analyze them in terms of various performance and fairness metrics. This section details the methodology used for the experiments, including the chosen datasets, algorithms, metrics, and the experiment setup.

3.1 Datasets

For the training of the models, we chose two real-world datasets from different domains.

- MovieLens 1M (ML) [4] - Contains around 1 million interactions from the MovieLens platform, between users and movies recommended to them, in the form of ratings. The recency of the dataset may be questionable, as it was released in 2003, however, it is a stable dataset commonly used for benchmarking recommender systems.
- Book-Crossing (BX) [16] - A similar-sized dataset in terms of interactions from the Book-Crossing community, which contains data about books and users' ratings for them. However, the main difference from the MovieLens dataset is the sparsity, as this dataset contains many more users and items, resulting in a sparsity of around 99.997%, increasing the diversity and generalization of our experiments, as real world data is often even more sparse.

Tables 1 and 2 show the specifics of the datasets.

Property	**MovieLens 1M**
Users	6,040
Items	3,706
User-item interactions	1,000,209
Sparsity	95.534%
User attributes	ID, gender, age, occupation
Item attributes	ID, title, release_year, genres
Interaction attributes	userID, movieID, rating, timestamp

Table 1: MovieLens 1M Dataset Statistics and Attributes

Property	**Book-Crossing**
Users	105,283
Items	340,556
User-item interactions	1,149,780
Sparsity	99.997%
User attributes	ID, location, age
Item attributes	ID, title, author, pub_year, publisher
Interaction attributes	userID, itemID, rating

Table 2: Book-Crossing Dataset Statistics and Attributes

3.2 Data pre-processing

In order to increase consistency and reduce training times, we employ a simple filtering strategy for the data - we only retain users and items which have at least 5 interactions, also known as 5-core filtering. We then split the interactions into training (80%), validation (10%) and test (10%) sets, and with this 5-core filtering we can ensure that each user has some interactions in each set. While this filtering does not have much effect on the MovieLens 1M dataset due to its relatively low sparsity and already filtered users, it does reduce the Book-Crossing dataset roughly by half in terms of the interactions.

Additionally, we want to investigate user-side fairness for advantaged and disadvantaged groups of users based on a given sensitive attribute, which will be the gender attribute in the ML dataset and the age attribute in BX. Therefore, we also filter the BX dataset to only retain users with ages 1 to 100, as there are some users with null age and some outliers with age > 200. We then split the users of Book-Crossing into 2 age groups - above (minority) and below 50 years old. Table 3 shows how the datasets change after filtering, while Figrue 1 shows the user demographics for the datasets in terms of the mentioned groups.

Property	**MovieLens 1M**	**Book-Crossing**
Users	6,040	10,288
Items	3,416	28,171
User-item interactions	999,611	399,240
Sparsity	95.157%	99.862%

Table 3: Dataset Statistics after Filtering

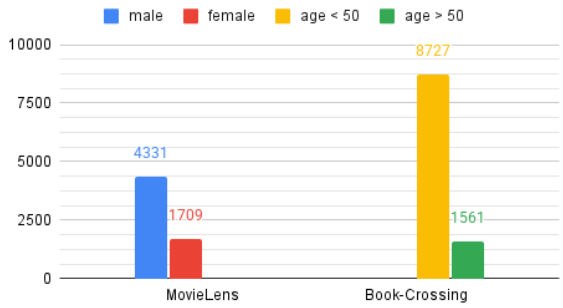

Figure 1: User demographics for the MovieLens and Book-Crossing datasets

3.3 Algorithms

We chose the following specific algorithms for evaluation:

- Baselines - these models do not have debiasing methods applied to them
  - Most Popular - A simple model that recommends the most popular items (items most interacted with), regardless of the user
  - Random - A simple model that recommends random items
  - ItemKNN/UserKNN - A traditional k-nearest-neighbors based approach. It calculates the similarity between users/items based on their interactions and recommends users either items interacted with by similar users, or items similar to the ones they interacted with. The specific implementation is based on [1]
  - MF (Matrix factorization) - A more nuanced approach that calculates lower-dimensional embeddings of latent features for users and items based on their interactions and then generates recommendations by matching these embeddings. The specific implementation is based on [5]
- Debiasing methods - applied on the MF model.
  - IPS (Inverse Propensity Scoring) - A method that weighs the effect of interactions on the training process based on how likely they are to occur, with unlikely interactions receiving a higher weight and therefore affecting the model more. The specific implementation is based on [7]
  - PDA (Popularity-bias Decounfounding and Adjusting) - A method that aims to decouple the effect of item popularity during training and then inject it back into the recommendations, in order to eliminate the amplification of the bias, while still leveraging the popularity for better recommendations. The implementation is based on [12]
  - MACR (Model-Agnostic Counterfactual Reasoning) - A method that simultaneously trains a regular MF model, and two supporting modules only based on the items and users respectively, and then generates recommendations by using counterfactual inference to essentially subtract the supporting modules from the main one. The implementation is based on [8]

3.4 Metrics

The results will be evaluated in terms of the following metrics for accuracy and fairness. Here, @K refers to the metric being evaluated on a list of the top K recommendations generated by the model, for example, in most of our experiments K=10. Most of the metrics are standard, as defined and implemented in RecBole [14] (see Section 3.5).

Accuracy

- Recall@K - fraction of relevant items recommended out of all relevant items
- Precision@K - fraction of relevant items recommended out of all recommended items
- nDCG@K (normalized discounted cumulative gain) - measure of ranking quality that accounts for the position of the hit by assigning higher scores to hits at top ranks
- Hit@K - if there is at least one relevant item in the list, it is a hit

Fairness

- Item-side fairness:
  - Coverage@K - coverage of recommended items over all items
  - TailPercentage@K - percentage of long-tail items (bottom 20% in terms of total interactions) in recommended items
  - PopularPercentage@K - percentage of the most popular items (top 10%) in recommended items
  - GiniIndex@K - measures the diversity of recommended items (1 = same items recommended to all users, 0 = all items recommended equally)
- User-side fairness:
  - NonParity Unfairness - the absolute difference between the overall average ratings of users of different groups as described in [10]
  - nDCG_Maj@K - the nDCG metric specifically for the majority group (male for the ML dataset, age $<$ 50 for BX)
  - nDCG_Min@K - the nDCG metric specifically for the minority group (female for the ML dataset, age $>$ 50 for BX)
  - nDCG_diff@K - disparity of the nDCG metrics between the 2 groups, computed as (1 - nDCG_Min / nDCG_Maj)

3.5 Experimental setup

All of the experiments were conducted in Python through the RecBole framework [14] [9] [13], which is a framework for recommender systems based on PyTorch. Additionally the Weights & Biases [2] framework was used to log and visualize results, and to tune hyperparameters of the models. The

Method	MovieLens 1M				Book-Crossing			
	Recall@10	Precision@10	nDCG@10	Hit@10	Recall@10	Precision@10	nDCG@10	Hit@10
ItemKNN	0.1612	0.1992	0.2549	0.7366	0.0611	**0.0169**	0.0486	0.1307
UserKNN	**0.1807**	**0.2032**	**0.2682**	**0.7707**	**0.0664**	0.0167	**0.0493**	**0.1362**
Random	0.0032	0.0056	0.0061	0.0520	0.0004	0.0002	0.0003	0.0017
Pop	0.0691	0.1020	0.1218	0.4949	0.0192	0.0058	0.0161	0.0519
MF	0.1579	0.1996	0.2520	0.7354	0.0328	0.0091	0.0230	0.0778
MF-IPS	0.1233	0.1596	0.1986	0.6591	0.0255	0.0076	0.0190	0.0663
MF-PDA	**0.1678**	**0.2074**	**0.2673**	**0.7517**	0.0264	**0.0085**	0.0200	0.0706
MF-MACR	0.1560	0.1946	0.2466	0.7286	**0.0347**	0.0081	**0.0226**	**0.0736**

Table 4: Accuracy metrics for baseline and debiasing recommender methods on MovieLens 1M and Book-Crossing datasets.

Method	MovieLens 1M				Book-Crossing			
	Gini@10	ItemCov@10	Tail%@10	Pop%@10	Gini@10	ItemCov@10	Tail%@10	Pop%@10
ItemKNN	0.9097	0.3878	0.0002	0.8315	0.7023	0.7096	**0.2551**	0.3072
UserKNN	0.9548	0.2075	0.0000	0.9536	0.9729	0.2024	0.0040	0.9026
Random	**0.1386**	**0.9997**	**0.2095**	**0.0831**	**0.2894**	**0.9743**	0.2016	**0.0989**
Pop	0.9939	0.0293	0.0000	1.0000	0.9996	0.0012	0.0000	1.0000
MF	0.9051	0.4287	0.0001	0.8124	0.9900	0.1111	0.0004	0.9434
MF-IPS	**0.8794**	**0.6043**	0.0006	**0.7733**	0.9938	0.1116	0.0026	0.9271
MF-PDA	0.9279	0.2426	0.0000	0.8945	0.9982	0.0137	0.0000	0.9589
MF-MACR	0.8821	0.5505	**0.0059**	0.7768	**0.9661**	**0.2502**	**0.0105**	**0.8820**

Table 5: Item-side fairness metrics for baseline and debiasing recommender methods on MovieLens 1M and Book-Crossing datasets.

Method	MovieLens 1M				Book-Crossing			
	NonParity	nDCG_Maj	nDCG_Min	nDCG_Diff	NonParity	nDCG_Maj	nDCG_Min	nDCG_Diff
ItemKNN	**0.0008**	0.2712	0.2136	0.2124	**0.0004**	**0.0509**	0.0356	0.3006
UserKNN	0.0055	**0.2854**	**0.2245**	0.2134	0.0033	0.0506	**0.0421**	0.1680
Random	**0.0008**	0.0066	0.0050	0.2424	0.0046	0.0002	0.0009	*-3.500*
Pop	0.0077	0.1348	0.0891	0.3390	0.0073	0.0165	0.0140	**0.1515**
MF	0.0032	0.2668	0.2144	**0.1964**	0.0380	0.0240	0.0178	0.2583
MF-IPS	0.0045	0.2110	0.1673	0.2071	0.0377	0.0192	0.0182	**0.0521**
MF-PDA	0.0053	**0.2826**	**0.2286**	**0.1911**	0.0079	0.0207	0.0160	0.2270
MF-MACR	**0.0008**	0.2620	0.2075	0.2080	**0.0019**	**0.0228**	**0.0211**	0.0746

Table 6: User-side fairness metrics for baseline and debiasing recommender methods on MovieLens 1M and Book-Crossing datasets.

random seed used for all random generation was 42, to ensure reproducibility of results. All the configuration settings are provided in the config directory of the repository provided in Appendix A.

4 Results

This section contains the full results of the experiments in terms of the metrics described in Section 3.4. The results are evaluated on the test set part of the datasets, after the training has finished using the training and validation sets, described in Section 3.2

This section is split into two parts based on the two research sub-questions.

4.1 RQ1: Accuracy and Fairness Trade-off

Tables 4, 5 and 6 show the full results on the MovieLens 1M and Book-Crossing datasets for the accuracy, item-side fairness and user-side fairness metrics respectively.

Discussion

Looking at the results in terms of the accuracy in Table 4, there are various notable patterns. The traditional ItemKNN and UserKNN models seem to be performing the best in terms of all the accuracy metrics. However, these are memory-based approaches which are not very efficient, especially for larger datasets. The random model is, as expected, the worst performing method, but it can still give us some insight as a baseline, so that it is clear that our models are at least working. We can also notice the relative difference in accuracy between the two datasets across all the models, due to the fact that the Book-Crossing dataset is much

sparser, making the recommendation task much more difficult. The Pop model, which always recommends the most popular items to all users, is performing considerably better than the random model, though still much worse than all the more complicated methods, showing that the recommendation task is not as simple as just using the popularity. The matrix factorization (MF) model is achieving more comparable results to the neighborhood-based models, while learning lower-dimensional embeddings for users and items, improving the efficiency of recommendations even for large datasets.

As for the debiasing methods, inverse propensity scoring (IPS) seems to be underperforming, especially on the MovieLens dataset, which might indicate problems with estimating the propensity solely based on item popularity. The popularity-bias deconfounding and adjusting (PDA) model, on the other hand, shows a small but still statistically significant improvement over the basic MF model on the MovieLens dataset, showing that debiasing methods can indeed improve accuracy in some scenarios. The counterfactual reasoning (MACR) model performs comparably to the standard MF model in terms of accuracy on the ML dataset and even better on the BX dataset, which is significant as it also offers considerable improvements in terms of fairness.

That leads us to the item-side fairness metrics in Table 5. Here, the Random model can serve as a baseline for a fully fair model. In order to analyze the trade-off between fairness and accuracy for the debiasing methods, we can choose nDCG as the main accuracy metric and item coverage (ItemCov) as the main item-side fairness metric, since the others seem to follow similar patterns. On the MovieLens dataset, PDA offers a 6% increase in nDCG at the cost of a significant 43% decrease in ItemCov, likely due to the fact that it leverages item popularity during recommendation. IPS, on the other hand, decreases nDCG by 21%, while increasing ItemCov by 41%, since less popular items have greater impact on the training. Lastly, MACR provides a nice middle ground in the trade-off with only a 2% decrease in nDCG, but a 28% increase in ItemCov. Furthermore, the results on the Book-Crossing dataset are somewhat similar but also notably different in some cases. Here, PDA decreases both nDCG and ItemCov, showing that it is not necessarily always a trade-off and in certain scenarios debiasing methods can be detrimental for both fairness and accuracy. However, MACR again gives a similar 2% decrease in accuracy, but a much more significant 225% increase in ItemCov. As for the other metrics, the TailPercentage metric is somewhat worrying, since the very low results across most models in both datasets mean that the bottom 20% of items in terms of the interaction amount receive barely any exposure, even though they are likely relevant at least to some users.

The user-side fairness of the debiasing methods is also worth acknowledging. In both datasets, it appears that the majority group of users (male for MovieLens, and age < 50 for Book-Crossing) receive significantly better recommendations than the minority group, regardless of the model used. The fact that this is also true for the Pop model, which simply recommends the most popular items to all users, regardless of their gender or age, might seem counterintuitive at first, but it highlights a possible caveat of the task. The problem is that the majority group, for example male users in the ML dataset, seem to enjoy similar movies, which are to some extent different than those preferred by female users. This causes the male-preferred movies to become more popular in general and increases the recommendation quality for male users, while reducing the quality for female users. Additionally, for the MovieLens dataset, it seems that none of the debiasing methods have much effect on the nDCG_Diff. However, in the Book-Crossing dataset it is clear that both the IPS and MACR methods significantly reduce nDCG_Diff by increasing the recommendation quality for the minority groups (nDCG_Min).

RQ1 Answer: Overall, it is clear that debiasing methods can have a significant impact on the trade-off between fairness and accuracy of recommender models, but the specifics heavily depend on the method as well as the dataset used. Both accuracy and fairness can be increased, usually at the cost of the other, although in some cases both can also be increased or decreased together. Out of the three debiasing methods evaluated, MACR performed the best in terms of the trade-off, with minimal impact to accuracy and significant increases in both item-side and user-side fairness.

4.2 RQ2: Debiasing method hyper-parameters

To analyze the effects of varying the hyper-parameters of debiasing methods on the fairness, accuracy and their trade-off, we need to look at each method separately as they each have different parameters controlling to what extent the debiasing is applied.

IPS

For IPS we can introduce a parameter "propensity_weight" (between 0 and 1) that scales the extent to which the effect of a particular user-item interaction on the training of the model is based on the item's popularity. Here propensity_weight = 0 means that the propensity is not considered at all, resulting in the same model as the standard MF, while propensity_weight = 1 means that an item with 1/10 of the popularity of the most popular item will have a 10 times larger effect than the most popular item.

Figure 2 shows the effect of propensity_weight on the item-side fairness (ItemCov) and accuracy (nDCG) of the MF model on the MovieLens dataset. We can see that increasing the weight increases item coverage but decreases the nDCG, as expected based on the previous results. The effect is most noticable near propensity_weight = 1, as for example, propensity_weight = 0.5 has almost no effect on the model. Additionally, while not shown in the figure, propensity_weight had no noticable effect on user-side fairness in this case, although as shown in Section 4.1 this would not be the case for the Book-Crossing dataset.

PDA

Part of the PDA debiasing method applies L2 regularization on the user and item embeddings, which has the parameter regularization_weight, which we can control. Figure 3 shows the effect of regularization_weight on the fairness and accuracy of the MF model on the MovieLens dataset. It seems that in our case, the default value of 0.001 has no significant effect on the model, while increasing the value is only detrimental

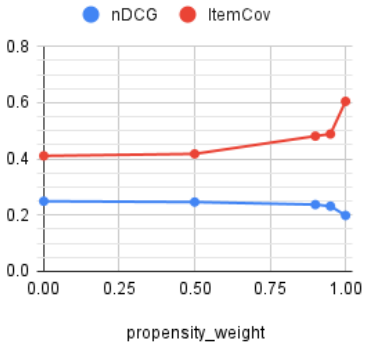

Figure 2: The effect of propensity_weight on the fairness and accuracy of the model

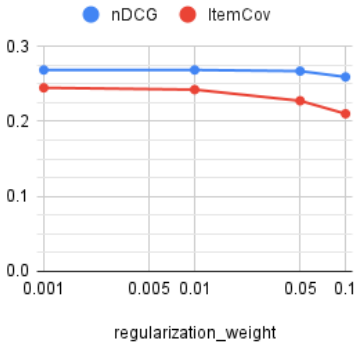

Figure 3: The effect of regularization_weight on the fairness and accuracy of the model

to both fairness and accuracy. However, this regularization might be important for larger models or sparser datasets.

MACR

The MACR model has a few controllable hyper-parameters, although the most important one for controlling the trade-off between fairness and accuracy is the C parameter described in [8]. It controls the degree to which the counterfactual inference based on the user and item modules is applied to the model during recommendation, with higher values resulting in more score being subtracted by the submodules.

Figure 4 shows the effect of C on the fairness and accuracy of the MF model on the MovieLens dataset. The effect is similar to propensity_weight of the IPS method, but with a far more significant impact on item coverage. In this case, however, the user-side fairness is also slightly negatively impacted, contributing to the fairness and accuracy trade-off.

RQ2 Answer: The hyper-parameters of debiasing methods can be used to tune the trade-off between fairness and accuracy based on the needs of the developer, with some methods being more flexible than others. Out of the three analyzed debiasing methods, MACR shows the highest flexibility in terms of controlling the trade-off and increasing item-side fairness.

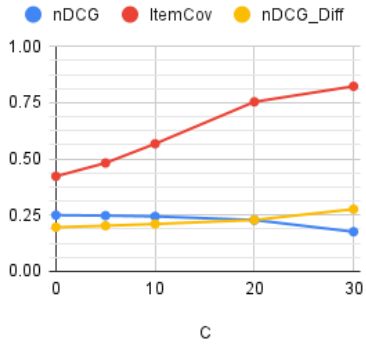

Figure 4: The effect of C on the fairness and accuracy of the model

5 Responsible Research

Since this is a project about fairness, ethics are at the core of the project topic itself; however, they should also be at the core of the research process. In general, we try to follow the TU Delft Code of Conduct [6] while performing research. All the sources and references used for the project are freely accessible and cited appropriately, in the IEEE format. The datasets used in the experiments (i.e. MovieLens 1M [4] and Book-Crossing [16]) are cited, publicly available and, according to their sources, obtained ethically as well. The code used to run the experiments is based on the open-source RecBole framework [14] [9] [13], however in most cases some adjustments and additions had to be made, so the code along with all the config files will be available through a GitHub repository in Appendix A, to enable reproducibility. Additionally, all configuration needed to reproduce the results is described in detail in Section 3 and any randomization (i.e. initialization or data splitting) was done using the set random seed 42, as set in the config files. It should therefore be theoretically possible to fully reproduce the exact same results without any major code modification.

6 Conclusions and Future Work

The purpose of this project was to analyze the effects of debiasing methods on the fairness and accuracy of recommender systems, particularly the trade-off between the two and how it can be controlled using hyper-parameters. To do so, three state-of-the-art debiasing methods were reproduced and evaluated: inverse propensity scoring (IPS), popularity-bias deconfounding and adjusting (PDA) and model-agnostic counterfactual reasoning (MACR), along with five baselines for comparison, on two different datasets: MovieLens 1M (ML) and Book-Crossing (BX).

Overall, we found that debiasing methods can have a significant impact on the trade-off between fairness and accuracy of recommender models, although the specifics heavily depend on the used method as well as dateset. As discussed in Section 4.1, both accuracy and fairness can be increased, though usually at the cost of the other, in certain cases, however, both can also be increased or decreased together. Of the three debiasing methods evaluated, MACR performed the best in terms of the trade-off, with minimal impact to accu-

racy (around 2% decrease) and significant increases in both item-side and user-side fairness (up to 225% increase in item coverage).

Additionally, in Section 4.2 we confirmed that hyperparameters of debiasing methods can be used to tune the trade-off between fairness and accuracy based on the needs of the developer, with some methods being more flexible than others. Of the three methods, MACR once again showed the most flexibility in terms of controlling the trade-off and increasing item-side fairness, making it arguably the ideal debiasing method for our datasets.

However, this project only evaluated three debiasing methods, all applied to the same matrix factorization model, and only on two datasets, which is a somewhat limited scope. Balancing the trade-off between fairness and accuracy is still a rather underexplored area, and future work could focus on analyzing other debiasing methods, using different datasets, and even other recommender models, such as content-based or knowledge-aware models. When doing so, we recommend using similar metrics for fairness and accuracy, as they showed a good overview of the performance and trade-offs of the models.

A Appendix - Code

All the code and configuration settings used for the project can be found in the GitHub repository at https://github.com/fcajagi/TUD-CSE3000-RecBole

It is a modified version of the RecBole framework.

References

[1] Fabio Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *RecSys 2013 - Proceedings of the 7th ACM Conference on Recommender Systems*, pages 273–280, 2013.

[2] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[3] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *28th Text REtrieval Conference, TREC 2019 - Proceedings*, 10 2020.

[4] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets. *ACM Transactions on Interactive Intelligent Systems*, 5:1–19, 1 2016.

[5] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42:30–37, 8 2009.

[6] Sabine Roeser and Samantha Copeland. Tu delft code of conduct. Technical report, Technische Universiteit Delft, 2020.

[7] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. Technical report, 2016.

[8] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1791–1800. Association for Computing Machinery, 8 2021.

[9] Lanling Xu, Zhen Tian, Gaowei Zhang, Lei Wang, Junjie Zhang, Bowen Zheng, Yifan Li, Yupeng Hou, Xingyu Pan, Yushuo Chen, Wayne Xin Zhao, Xu Chen, and Ji-Rong Wen. Recent advances in recbole: Extensions with more practical considerations, 2022.

[10] Sirui Yao and Bert Huang. Beyond parity: Fairness objectives for collaborative filtering. 5 2017.

[11] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 353–362. ACM, 8 2016.

[12] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. Causal intervention for leveraging popularity bias in recommendation. In *SIGIR 2021 - Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 11–20. Association for Computing Machinery, Inc, 7 2021.

[13] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, et al. Recbole 2.0: Towards a more up-to-date recommendation library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4722–4726, 2022.

[14] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *CIKM*, pages 4653–4664. ACM, 2021.

[15] Yuying Zhao, Yu Wang, Yunchao Liu, Xueqi Cheng, Charu Aggarwal, and Tyler Derr. Fairness and diversity in recommender systems: A survey. 7 2023.

[16] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web - WWW '05*, page 22. ACM Press, 2005.