Gregor Mayr

# Calibration in Recommender Systems

## BACHELOR THESIS

for the attainment of the degree of

Bachelor of Computer Science

submitted to

**Graz University of Technology**

Supervisor:
Assoc. Prof. DI Dr. Elisabeth Lex

Advisor:
DI Dr. Dominik Kowald, BSc.

Institute of Interactive Systems and Data Science

Graz, Date

## EIDESSTATTLICHE ERKLÄRUNG

## *AFFIDAVIT*

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit/Diplomarbeit/Dissertation identisch.

*I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis/diploma thesis/doctoral dissertation.*

_____ _____

Datum / Date Unterschrift / Signature

# Abstract

As recommender systems are deployed more widely throughout the Web and mobile applications, certain flaws can appear, as more and more users come in contact with them. This has a variety of reasons, one of them being the very complex characteristics of the users themselves, which leads to biases and inaccuracy in recommendation results for some user profiles, as previous research has shown. A consequence is not only the mismatch of categories of items in user profiles and recommendations, called miscalibration, but even more so the uneven distribution of this metric, which we define in this thesis as *unfairness*. In this thesis we look at 3 metrics, the MAE measuring the accuracy of recommendation rankings, the already mentioned miscalibration metric and the popularity lift, which indicates the propagation and amplification of popularity bias through algorithms. We split the user profiles into 3 user groups based on their mainstreaminess/item popularity, called *LowPop*, *MedPop* and *HighPop*, which indicates the similarity of a profile to the aggregated profiles in the dataset, and compare the measurements of our metrics between said groups. We tested multiple datasets ranging from books, music, movies to anime, for a total of 5 datasets. We experimented on multiple collaborative filtering algorithms, which include basic algorithms such as Random and MostPopular, baseline options and neighbourhood- and factorization-based methods. The goal of this thesis is to find the best performing algorithms and to investigate, if all metrics behave similar on all datasets, or if this is not the case, at least find a metric which can provide uniform results on no matter which dataset we used.

# Zusammenfassung

Mit der zunehmenden Verbreitung von Empfehlungssystemen im Internet und bei mobilen Anwendungen kommen durch die steigende Anzahl von Interaktionen mehr und mehr Schwachstellen ans Tageslicht. Dafür gibt es eine Vielzahl von Gründen, einer davon sind die meist sehr komplexen Merkmale der Nutzer dieser Systeme, die bei einigen Nutzerprofilen zu Ungenauigkeiten von Ergebnissen führen, wie frühere Forschungen gezeigt haben. Eine Folge davon ist nicht nur die Nichtübereinstimmung der Kategorien von Artikeln in Benutzerprofilen und Empfehlungen, die als Fehlkalibrierung bezeichnet wird, sondern mehr noch die ungleiche Verteilung dieser Metrik, die wir in diesem Papier als *Unfairness* definieren. In dieser Arbeit betrachten wir dabei drei Metriken: den MAE, der die Genauigkeit der Rankings von Empfehlungen misst, die bereits kurz erwähnte Miscalibration-Metrik, die die Fehlkalibrierung misst, und den Popularitäts-Lift, der die Ausbreitung und Verstärkung von Vorurteilen in Datensätzen anzeigt, die eine Verbindung mit dem Bekanntheitsgrad eines Items aufweisen. Wir teilen die Nutzerprofile in drei Gruppen ein, basierend auf ihrer Mainstreamigkeit/Popularität, welche wir *LowPop*, *MediumPop* und *HighPop* nennen, die die Ähnlichkeit eines Profils mit den aggregierten Profilen im Datensatz angibt, und vergleichen die Messungen unserer Metriken zwischen diesen Gruppen. Wir haben mehrere Datensätze getestet, die von Büchern, Musik, Filmen bis hin zu Anime reichen, was insgesamt 5 Datensätze ausmacht. Es wurden mehrere Algorithmen der kollaborativen Filterung getestet, darunter grundlegende Algorithmen wie Random und MostPopular, Basisoptionen wie Co-Clustering, sowie auf Nachbarschaft und Faktorisierung basierende Methoden wie UserKNN und NMF. Das Ziel unserer Analyse ist es, die leistungsfähigsten und genauesten Algorithmen zu finden und zu untersuchen, ob sich alle Metriken auf allen Datensätzen ähnlich verhalten oder falls dies nicht der Fall ist, zumindest eine Metrik zu finden, die unabhängig vom verwendeten Datensatz einheitliche Ergebnisse liefert.

# Contents

# List of Figures

# List of Tables

# Introduction

Nowadays, recommender systems are a major provider of information on the Web and mobile applications, by providing us, the users, with suggestions of items we might be interested in [Ricci et al., 2010, Jannach et al., 2010]. These items can be many things, such as books, news [Turcotte et al., 2015], music [Kaminskas et al., 2013, Schedl and Hauger, 2015], movies [Chen et al., 2015], but also more complex articles like electronics, automobiles, dating profiles [Brozovsky and Petricek, 2007] and even financial [Felfernig et al., 2007, Musto et al., 2015] or legal advice [Thomas et al., 2020].
The connection between miscalibration, popularity bias and fairness received a lot of attention in recent times [Abdollahpouri et al., 2019, Kowald et al., 2020, Yao and Huang, 2017], but most papers only take a maximum of 2 datasets into consideration [Lin et al., 2020, Abdollahpouri et al., 2020], leaving a lot of room for interpretation. One of the problems is, for example, that one dataset may show a clear connection between two metrics on a certain algorithm, but the other one either has no connection at all or the connection moves to another algorithm. We want to test some hypothesis these papers address in our environment and ask these questions:

- **R1**: Are particular user groups affected differently on the basis of miscalibration?
- **R2**: Is there a connection between MAE and miscalibration?
- **R3**: Has popularity lift an influence on miscalibration?

With an increase of applications, more users are exposed to recommender systems and the data it was trained on may not cover the huge range of characteristics of users [Almazro et al., 2010]. This leads to biases and inaccuracy in recommendation results of certain users, which essentially means, the system is treating some users unfairly [Abdollahpouri et al., 2019]. Fairness in recommender systems is not yet clearly defined and many interpretations exist, we will therefore define it in this thesis as the consistency of a metric over different user groups.

We will assign users to 3 groups using the item popularity of their user profiles, a metric which denotes the quantity of popular and mainstream items in a user profile compared to a group or all profiles, into low, medium, and high popularity groups, except for the LastFM dataset (music), where we use a similar metric to item popularity, called mainstreaminess [Steck, 2011, Schedl and Hauger, 2015]. We will then look at different metrics to find a connection between them, such as the mean absolute error (MAE), miscalibration and popularity lift.

While the MAE is a well-known metric in statistics and recommender systems [Sarwar et al., 2002], used preferably in the latter to measure its accuracy by comparing the ranking of items between the user profile and its recommendations, which is also the case in this thesis. Additionally, the MAE is quite stable against outliers, which makes this metric more preferable than less robust metrics, *i.e.*, RMSE. The miscalibration metric on the other hand measures the consistency of the spectrum of categories of items between the user profile and its recommendations: Say, a user profile of a music recommender contains songs with 70% *pop* and 30% *country*, a recommendation is calibrated if the resulting recommendation also contains the same amount of genres, 70% *pop* and 30% *country* [Abdollahpouri et al., 2020]. We will mainly use this metric to compare the miscalibration between algorithms and user groups.

Finally, popularity lift is defined as the difference of the popularity of items between rated and recommended items, meaning, we get a negative measurement if the average popularity of items in the recommendation result is less than in the user profile and a positive one if we receive suggestions containing items with an overall higher average popularity, essentially popularity lift measures the propagation and amplification of popularity bias in our system [Lee and Hosanagar, 2014]. This metric should give us a good overview about which algorithm is the least affected by popularity bias.

The focus of this thesis is to reproduce and measure the performance of these metrics using several collaborative filtering recommendation algorithms [Schafer et al., 2007]. This will be done on 5 different datasets representing books, movies (2), music and anime, to find out, if the mentioned metrics have a similar behaviour using the same algorithms on different data, such as having similar relative values and the correlations between metrics. The datasets each consist of a list of user profiles containing items users have rated. It should be noted, that the datasets do not

have the same origin and as such implemented different rating systems, additionally, items can have a varying amount of categories attached to them and the total count of categories or genres also varies from dataset to dataset. This may not be a bad feature, as we can determine if this has an influence on the recommendation result of our system, *i.e.*, propagates different biases and influences the fairness of our system.
This thesis will begin with a chapter about related work, then we will talk about our methodology, including datasets and algorithms used, the evaluation metrics and evaluation protocol. Then we will go over the conducted experiments and their results and finally, we will conclude and talk about possible research in the future.

# Related Work

## Recommender Systems

Nowadays, when browsing the web, it is nearly impossible to receive information which was not recommended to you [Ricci et al., 2011]. Most of the time, you are not even aware that a search result was personalized, or a video was recommended. We are already so used to receive personalized data on the web, that articles not recommended to us stand out a lot, as they do not fit in our personalized bubble [Nguyen et al., 2014]. This is made possible through recommender systems, which provide suggestions to the user, helping him to navigate through a large amount of data, preventing the all too common information overload happening in the web [Schafer et al., 2007]. Without recommender systems, for example, buying an article at an online shop, can get very complicated, as the user has to choose between thousands of similar articles, overwhelming him. As one can imagine, if the system can recommend the best fitting article to the user of all those articles, the user would not only be more efficient in his search, but also more satisfied, motivating him to use the system more often, placing him in a positive feedback loop [Mansoury et al., 2020a].
Recommender systems work very similar to recommendations between two people: A friend may recommend a song he has listened to or a movie he watched to you, or alternatively, a book review in a newspaper recommends said book to you. In short, a trusted source recommends an article to you [Jameson et al., 2015]. These so-called articles are called items in recommender systems, and are the main target

a recommendation algorithm has to predict.
Recommender systems try to satisfy two parties: The user which tries to find information and the service providing the information. The user is using the system predominantly for finding good items, but can also have additional interests, like helping or influencing others by adding information, testing the system if it is credible or not, searching for sequences or bundles of items, like TV-Series or playlists, or just browsing for content. The service on the other hand tries to increase the interaction with items, mostly to sell more, but also to improve the popularity with the user by making good suggestions, raising the loyalty of the user in parallel [Ricci et al., 2010].

### User, Items, and Transactions

The scope of functions of recommender systems can be manifold, requiring a range of information and knowledge sources which provide the necessary data to work properly [Park et al., 2012].
To better understand recommender systems, we have to first understand what is needed to make a prediction in the form of a suggestion. A recommender system needs 3 indispensable components, which are: the user which uses the recommendation system, the items we want to suggest to the user, and prior transactions between the user and items in the system.

- **User**: Users, the primary target of recommender systems. As mentioned before, they have a wide range of objectives by using such a system. A user can be modelled in different ways, as some systems use more user information as others, but more on that later, when we go into detail about collaborative filtering and its alternatives. In short, user information can range from a simple list of ratings about items the user has interacted with, to more personal information of the user itself, like *i.e.*, age, gender, education and profession, but also not explicitly mentioned data, for instance the interaction behaviour or relationships with other users. A user model is then generated by the system, resulting into something similar to a simple vector of values, which can be compared and matched to others.

- **Item**: Items are the entities or objects recommended to the user by the recommendation system. Such items can be articles in an online store, people

on a dating platform, songs, movies, books and more, essentially they are the target of the user when using the system. As we can see, the complexity of those mentioned examples can be very diverse, and therefore we separate them into classes of complexity and value. Examples for low complexity are songs, movies, news and books, with higher complexity we can associate electronics and machines, and the highest complexity can contain travel plans, insurance, and jobs. The value of an item can be described as a metric of how much purpose an item has to the user, this can be positive, but also negative. This value also correlates to the cost of finding the item in the system, such as a cognitive cost like search-time or a monetary cost if the service is charged. Recommender systems then use the properties of an item, *i.e.*, genres of a movie or categories in a book, to construct an information structure which can either be used to build a user profile in collaborative filtering or used as a springboard to find similar items in content-based filtering. The information structure can be as simple as a single ID-string or complex like a feature vector with multiple dimensions, depending on the filter method used.
There are many algorithms with many approaches to predict an article, and further chapters will go more into detail about some of them, but the most important feature most have in common, is the ranking of items. The rank of an item is given by the algorithm, which was used to recommend said item to the user. The algorithm most of the time does not only recommend a single item, but a list, in which the items get ranked based on the preference the user has shown before, containing the top $N$ items most probable to satisfy the user.

- **Transaction**: Recommender systems use the connection between user and items to make suggestions: so-called transactions. The system records these transactions with either explicit or implicit feedback of the user, which is then used by the algorithm of the system to make specific recommendations and calculate the ranking of items.
  If we talk about explicitly given feedback, we think about a rating which was entered by the user. This rating can be differentiated into different types:
  - Numerical ratings, *i.e.*, a star rating between 1 and 5.
  - Ordinal ratings, *i.e.*, a rating in a questionnaire like disagree/agree.

- Binary ratings, which can be a simple yes or no, like or dislike.
- Unary ratings, which tell the system if it has information about the user and an item or not.

On the other hand, if we talk about implicit feedback, the rating is inferred by the system and not the user. This is done by analysing the actions of the user, *i.e.*, the user enters a search-word to look up a certain article, the system then knows what items the user wants to see and can recommend similar items in the future. Another information source for example could be the selected settings when the user explicitly wants to get a recommendation about an article, which can be reused further down the line. This is especially helpful with the cooperation of information tags or metadata associated with items, which the system can exploit to enhance recommendations [Oard et al., 1998].

### Collaborative Filtering

As already brought up in the previous chapter, some recommender systems use a method called collaborative filtering to infer ratings which will be used to make a prediction [Breese et al., 2013]. For this method, 2 approaches are possible: either user-based or item-based collaborative filtering. The basic idea of collaborative filtering is to utilize similarities of collected information of other users to compute a missing rating. As we know, the system explicitly or implicitly collects a rating for each item by making use of transactions, this collection is represented as an enormous matrix containing every user and item, called the utility matrix [Ricci et al., 2010]. This utility matrix is mostly empty, meaning the system has no information about the relation of the user and item in question, and not meaning the rating is low or zero. The objective of the system is to predict said blank elements. These predictions are made through analysing similar user profiles in the utility matrix, with "similar" meaning profiles, which have similar ratings for the same items as our user profile, using the correlation, that users with similar opinions or tastes, will have similar opinions in the future, as long the user in question remains stable in his opinions and convictions.
Looking at similar user profiles, as stated above, is called user-based collaborative filtering. Whereas user-based filtering centres around the user, item-based collaborative filtering switches the role of user and item, and instead of looking at similar

user profiles, we instead search for similar items, which we will then use to infer a rating for the item in question [Sarwar et al., 2001].
A common approach is the use of a similarity function to determine how many users or items are similar to each other, which results in a "neighbourhood" of similar objects. This "neighbourhood" can then be used to predict a rating for an object, be it a user or an item [Sarwar et al., 2000].
Even though it looks like user-based and item-based approaches exclude each other, that is actually not the case. It is very possible to combine them into a dual-approach, netting better results [Wang et al., 2006].
Lastly, collaborative filtering also has some weak points, namely if the user does not provide enough information to start a recommendation, if the system has not enough users to find a match (cold start), if the utility matrix is too sparse, and we are unable to find enough ratings for the target, if no ratings for an item exist at all and finally the problem of popularity bias, which we will discuss in a later chapter [Lee et al., 2004].

### Content-based Filtering

To circumvent some of the problems with collaborative filtering, *i.e.*, a cold start and a sparse utility matrix, we can look at what we already have. Collaborative filtering only looks at transactions between a user and his items, ergo ratings, whereas a content-based approach considers the "content" of the user profile, that is his preferences, and that of items, which we can summarize as categories [Aggarwal et al., 2016].
To be able to recommend items with this approach, it is essential to build up the preferences of the user, as it tries to recommend items similar to them, fundamentally recommending similar items the system knows the user likes. Before we discuss some approaches more in detail, we have to first ascertain what content is in the first place. In short, we can describe content in two different forms: structured and unstructured. Whereas structured content is represented as uniform attributes for every item, essentially metadata like genre, author, type, price or keywords, just to mention a few, unstructured content on the opposite constitutes only of a free-text description.
Content-based approaches are many, but to give some idea what is possible with content-based filtering, here are some examples:

- Dice coefficient: Uses the connection of categories to find unseen items which are similar to the preferences of the user.
- TF-IDF (Term Frequency – Inverse Document Frequency): Analyses a retrieved text to extract words which seem to represent said text, essentially extracting keywords with an assigned value [Wang et al., 2018].
- Relevance feedback: Classifies items as relevant or not through entered keywords in a search by the user. To increase accuracy, the system asks for feedback on the results, extending the content of items [Lops et al., 2011].
- Probabilistic method: Categorizes items into classes and finds similar items through conditional probability.

If we look at the advantages of this method, we see that contrary to collaborative filtering we do not need user profiles of others, solely building on the user profile of the active user. This also leads to transparency in the system, as it is very simple to describe to the user, why the system recommends a certain item to him, simultaneously increasing the trust of the user: *this movie X was recommended to you, because you liked movie Y* [Cramer et al., 2008]. Finally, it solves one of the biggest problems of collaborative filtering, the cold start problem. This problem arises due to the fact of items having no ratings, essentially making it impossible to recommend it to other users. Here shines the content-based approach, as it does not need a rating to make a recommendation, relying on the content of the item itself instead.

But this method of course has its limitations too, for example the "content" of an item may be too short or important information may be missing, also due to the fact that little to no feedback is given by the user, which leads to grading the relevance of an item incorrectly. Additionally, it is possible that some content cannot be extracted automatically, such as multimedia items like videos or songs, resulting in the problem above. Another common problem of such a system is the case of overspecialization, in which due to the algorithms used, the recommended items are too similar or even identical to previous items the algorithm had recommended.

### Knowledge-based Recommendations

There is a major problem with both methods described before, that is, items can have a high complexity [Watson et al., 2016]. If we look at collaborative filtering for example, it struggles with the amount of ratings, as most complex items are either unique or not affordable, resulting in a low number of available ratings. Additionally, most of the time the user wants to explicitly specify extra parameters or preferences, which have to be implemented as constraints into the system, neither supported by content-based nor collaborative filtering. Another important role plays in time, as some items will lose or increase in value dramatically. To list a few of such complex item domains, we have: cars, real estate, finance, electronics, and travel plans [Burke, 2000].
Knowledge-based recommender can be used as a generalization of recommender approaches which are based on the knowledge about item and user profiles and recommendation theme, to list a few:

- **Conversational**: As the name implies, this form of recommender system infers information through "conversation" with the user, also known as a feedback loop [Chen et al., 2013]. As items are, as already mentioned, very complex, it is not possible to match it to a user profile perfectly. Here the conversational nature of this method shines, as the profile can be extended through the additional feedback. It is even possible to start with a blank profile and building it up with every new recommendation.
  Conversational recommenders can be split into two types: constraint-based and case-based. A constraint-based system uses a defined set of rules to make recommendations, retrieving items which fulfil said rules and the requirements set by the user. On the other hand, case-based systems are based on different similarity measures, retrieving items which satisfy a certain similarity to the requirements given by the user, which he normally gives in the form of critiques, such as *nicer*, *costs less money* or *more traditional*.

- **Search-based**: This system uses its very nature to restrict the recommendation results to items with relevance. The process can be described as follows: the user enters a question, to which the system responds with a series of possible items, which in turn get selected by the user. The system can thus compile a set of preferences the user has on the base of which items(s) were clicked on.

Search-based systems build mostly on a constraint-based system, as the choices of the user can best be represented as such [Felfernig and Burke, 2008]. There is also a conjunctive query-based solution, which not only retrieves items which meet all constraints, like the constraint-based approach, but instead tries to identify item tuples that fulfil the criteria of a given query. This query is executed on the item catalogue as a set of selection criteria which are connected conjunctively.

- **Navigation-based**: Navigation is done by looking at local features to determine a path to the intended goal. This is the inspiration of navigation-based recommendation: user feedback in form of critiques is used as markers to specify a change of the direction the recommendation is taking – essentially changing the item ranking on the fly [Chen and Pu, 2012].

### Further Approaches

In recent years, the science community conducted a lot of research in the area of recommender systems, discovering new ways to recommend items and also trying to eliminate weak points of more generic approaches, like collaborative and content-based filtering [Adomavicius and Tuzhilin, 2005, Melville and Sindhwani, 2010]. In this section, we want to mention a few of them, to better understand the possibilities available right now:

- **Group recommenders**: An interesting approach is to look at a group instead of an individual when making recommendations [Masthoff, 2011]. This has several benefits, *i.e.*, you can bundle decisions of all group members to circumvent biases which can arise during a recommendation, leading to sub-optimal results. There are a few differences between single users and group recommenders, such as changes in the interface, the influence of opinions of group members, prevention and solving of conflicts which can arise and the effect of psychology in a group [Felfernig et al., 2018].
  Possible applications of such a system could be recommendations of group activities such as going to a restaurant or watch a movie, but also at work, *i.e.*, a software team planning a release or software requirements.

- **Critiquing-based**: This system is based on knowledge-based recommenda-

tions. It is similar to a conversational or navigation-based system, using critiques as feedback to essentially optimizing its recommendations [Reilly et al., 2004, McCarthy et al., 2010, Mandl and Felfernig, 2012].
Here we want to go a bit more into detail about critiques, as there are many different types:

  - Unit critiques: Critique single attributes or properties of an item.
  - Compound critiques: Critique multiple attributes or properties of an item.
  - Dynamic critiques: Critiques change dynamically depending on available options and prior critiques the user has left.
  - Incremental critiques: Takes the history of critiques and prior interactions into consideration, minimizing the effort put on the user.
  - Experience-based critiques: Critique the recommendation itself.

- **Hybrid recommenders**: Hybrid recommenders combine different strategies into one, the main goal being the elimination of disadvantages some approaches have [Gomez-Uribe and Hunt, 2015]. For example, combining content-based filtering with collaborative filtering, which abolishes the cold start problem of collaborative filtering using different methods, such as a weighted approach, which weights both recommendations and sums them up, calculating a recommendation score [Burke, 2007]. Other methods include a mixed approach which, similar to the previous one, collects individual scores of each recommendation strategy to determine the ranking of an item.
  A hybrid strategy increases the accuracy of an item, as multiple strategies can be combined, making full use of all available data.

- **Advanced algorithms**: Recommendation strategies can be very complex and advanced, having a deep mathematical or machine learning background. We want to mention two examples which come from these backgrounds, namely matrix factorization and deep learning based recommender systems.
  Matrix factorization is part of collaborative filtering, using the user-item utility matrix to decompose it into a product of matrices with lower dimensions [Luo et al., 2014]. This strategy gained recognition during the Netflix price chal-

lenge, as it was very effective, winning at third place. Furthermore, the recommendation results can be further improved by assigning different regularization weights based on the popularity of items and activity of users [Koren et al., 2009].

The deep learning approach can be tailored to very specific data, making it very compelling for major tech companies to use this strategy. We have to differentiate between two types of techniques: deep neural networks, which excel in sequential data processing, and convolutional neural networks, made for non-Euclidean data. Deep learning needs a lot of fine-tuning and can be modified and switched multiple times a day, if we look at Google Search for example. It is therefore mostly only used by big companies, as they have enough data to feed such neural networks [Cheng et al., 2016, Zhang et al., 2019].

## Popularity Bias, Fairness and Calibration

With the increasing rate of recommender systems used on various platforms, these systems get exposed to a huge diversity of users. As users can have very different preferences, a system can experience its limits at some point because of missing or lacking data, exposing multiple problems [Abdollahpouri, 2019, Elahi et al., 2021, Lin et al., 2020, Jain et al., 2015]. We want to mention a few of them, explaining the reasons why they occur and what metrics we can use to measure them, to better understand our system and finally fix them the best we can.

### Popularity Bias

Bias is a prevalent problem in recommender systems, this is mainly caused by the data it was trained on, which contains these biases and gets propagated to the recommendation result. There are different biases we know of, such as racial biases or biases emerging due to biased categories, *i.e.*, outliers in the dataset, for example, the dataset only contains songs with the genre *Pop*. Popularity bias is not very different from the last mentioned example, as it defines a bias towards popular items, specifically items with popular categories. This form of bias gets further emphasized through the algorithm used in the system, as this factor has a very clear influence on the recommendation result and its popularity bias measure [Nematzadeh et al.,

2017, Friedman and Nissenbaum, 1996], with of course the *MostPopular* algorithm having the greatest impact, as it, as the name suggests, recommends items based on their popularity. Another factor is the disparity of ratings between popular and unpopular items, with popular items gaining a lot more ratings due to their popularity. But just because an item is popular, doesn't mean the user is interested in said item. On the contrary, he may be more interested in less popular niche items. Herein lies the problem, as the data and algorithm propagates bias and more popular items land in user profiles of niche users, amplifying this bias [Lee and Hosanagar, 2014, Abdollahpouri et al., 2020, Jannach et al., 2015]. This bias can be measured, we use a metric called *Algorithmic Popularity Lift*, which measures the difference of popularity of items in user profiles and those recommended, but this will be talked about in further detail in a later chapter.

### Calibration

The term calibration in recommender systems describes the consistency of the spectrum of categories between the user profile and its recommended items. Say, for example, a user profile of a music platform consists of items with 75% of genres being pop, 23% rock and 2% heavy metal, the same spectrum of genres should appear in recommendations.
If the spectrum diverges, we call the recommendation miscalibrated. We will introduce a metric for this miscalibration in a later chapter, simply called the *miscalibration metric*. Miscalibration tells us how personalized a recommendation really is, which helps us test and find good recommendation strategies for our system, paying special attention to the algorithms used, as they can influence this factor the most, as we will discuss later. It is also important to mention, that not all systems benefit from a more personalized recommendation, as it locks the user into a personalized bubble, blinding him from eventual items he might be interested in, such as news or articles in a shop [Brynjolfsson et al., 2006, Hendrickx et al., 2021, Kunaver and Požrl, 2017, Möller et al., 2018].

### Fairness

The definition of fairness in recommender systems is not clearly defined, and numerous attempts have been made to describe it [Kamishima et al., 2012, Yao and Huang,

2017, Narayanan, 2018, Singh and Joachims, 2018, Zafar et al., 2017]. Fairness in recommender systems can be many things and its meaning depends on multiple factors, such as the domain it is operating in, the characteristics of users or user groups, *i.e.*, privacy, and the actual goal of the design of the system itself, for example, a simple book recommender which only purpose is to recommend the most popular items vs. a heavily personalized system.
In summary, fairness describes the consistency of a metric over all users or user groups, Mansoury et al. [Mansoury et al., 2020b] and Eksrand et al. [Ekstrand et al., 2018] for example, describe it as a consistent performance across different user groups, like females receiving more inaccurate recommendations than males. A paper from Sirui Yao and Bert Huang [Yao and Huang, 2017] also define four metrics that address different forms of unfairness in biased data. It also mentions an important key point, which is, that we can measure (un)fairness by looking at the consistency of a metric over users or groups. If we take this fact into consideration, we can use already defined metrics such as the miscalibration metric or popularity lift and look if they are consistent across users or groups, if not, we found an unfair recommendation approach.

# Methodology

This chapter will go into more detail about the technical aspects of this thesis, such as the origin and characteristics of the datasets we will use, the algorithms we want to perform tests on and how the evaluation metrics for our questions are derived, such as the MAE, miscalibration metric and the popularity lift. Finally, we will go over the evaluation protocol, which will describe how the datasets, algorithms, and metrics will be used in our experiments, and last but not least, we will infer more details about the research questions we asked at the beginning of this thesis.

## Datasets

As already mentioned, we will use 5 different datasets to observe how data affects the final recommendation result in a recommendation system. We did not mine these datasets ourselves, but they are made publicly available for researchers by various companies, websites or other researchers. For our purposes, we preprocessed

the available data and split it into several tables:

- `user_events`: This file consists of all user interactions with items, aka, it contains user profiles pertaining to the system. A row consists of the *user ID*, *item ID* and the *preference* (the *rating* of an item). Additionally, a timestamp and category IDs could also be included, with the latter being associated to a special file called `user_events_cats` for efficiency.

- `categories`: The `categories` file includes all categories (genres) in string format with an associated ID, which can mostly be omitted, as the ID is the same as the row number of the category name.

- `low_main_users, medium_main_users, high_main_users`: These files include the IDs of user profiles with their associated mainstreaminess' or item popularity score. Each file contains one thousand user IDs with either *low*, *medium* or *high* mainstreaminess (item popularity). The mainstreaminess of users will be used to determine the *taste* of a user profile in form of a popularity score, *i.e.*, if a user likes popular items his mainstreaminess factor is higher than a user who likes more niche items.

This will be the data structure of our datasets, allowing us to reuse the code for each of them and thus keeping the experimental setup as uniform and free of external biases as possible. Even though the datasets may originate from different sources, which also means that they are used in different systems with unrelated user bases, our setup should be the same for each dataset to ensure that the results of our recommendation framework and setup can be compared to each other.
We will now talk about each dataset a bit more in detail to have a better overview of the origin and time of retrieval, characteristics of the content and underlying preprocessing steps we have taken to conform to the specifications we have defined above. As already mentioned, the item categories of these datasets are not the same and include 4 different types: *books*, *music*, *movies* and *anime*. The datasets are called as follows: *BookCrossing* for the book media type, *LastFM* for the music media type, *MovieLens* and *Yahoo! Movies* for the movie media type, and finally, *MyAnimeList*, for the anime media type.

| Dataset | $\|U\|$ | $\|I\|$ | $\|R\|$ | $\|R\|/\|U\|$ | $\|R\|/\|I\|$ | Sparsity | $R$-range |
|---|---|---|---|---|---|---|---|
| LastFM | $3,000$ | $131,188$ | $1,417,791$ | 473 | 11 | 0.996 | $[1-1,000]$ |
| MovieLens | $3,000$ | $3,667$ | $675,610$ | 225 | 184 | 0.938 | $[1-5]$ |
| BookCrossing | $3,000$ | $223,607$ | $577,414$ | 192 | 3 | 0.999 | $[1-10]$ |
| MyAnimeList | $3,000$ | $9,450$ | $649,814$ | 216 | 69 | 0.977 | $[1-10]$ |
| Yahoo! Movies | $3,000$ | $8,438$ | $78,563$ | 26 | 9 | 0.997 | $[1-5]$ |

Table 1: Dataset statistics about the 5 datasets we use in this thesis. It includes the number of users $|U|$, the number of items $|I|$ and the number of ratings $|R|$. Sparsity describes the ratio between observed ratings $|R|$ to possible ratings $|U| \times |I|$. The $R$-range indicates the range of possible rating values.

### BookCrossing

The BookCrossing dataset[1] as the name suggests, is a collection of user data from the Book-Crossing community [Ziegler et al., 2005], which operates mainly on the Book-Crossing website[2]. The website describes itself as a smart social networking site and the core idea is to give every book a unique identity with which it is possible to track how a book is passed from reader to reader, building a network between readers who share books. These books exist in the real world and users can record their interaction with a book through the website or app. Users are also able to add their own books by marking them with BCIDs (BookCrossing Identity Numbers) and are then able to follow its journey across the world. There are currently 1,927,984 BookCrossers and 13,665,782 books travelling throughout 132 countries, with top countries being the USA (29%), Germany (16%), UK (13%), the Netherlands (11%), Finland (10%) and Canada (8%) (10/21).
The dataset itself was collected by Cai-Nicolas Ziegler in a 4-week crawl in 2004 and contains 278,858 anonymized users providing 1,149,780 ratings about 271,379 books. It is composed of 3 tables:

- *BX-Users*: This table contains an ID for each individual user and, if provided, additional demographic data such as age and location.
- *BX-Books*: BX-Books contains all available books and are identified by their ISBN. Books with an invalid ISBN have already been removed from the dataset. Additionally, the table includes various information such as the title, author,

[1] `http://www2.informatik.uni-freiburg.de/~cziegler/BX/`
[2] `https://www.bookcrossing.com/`

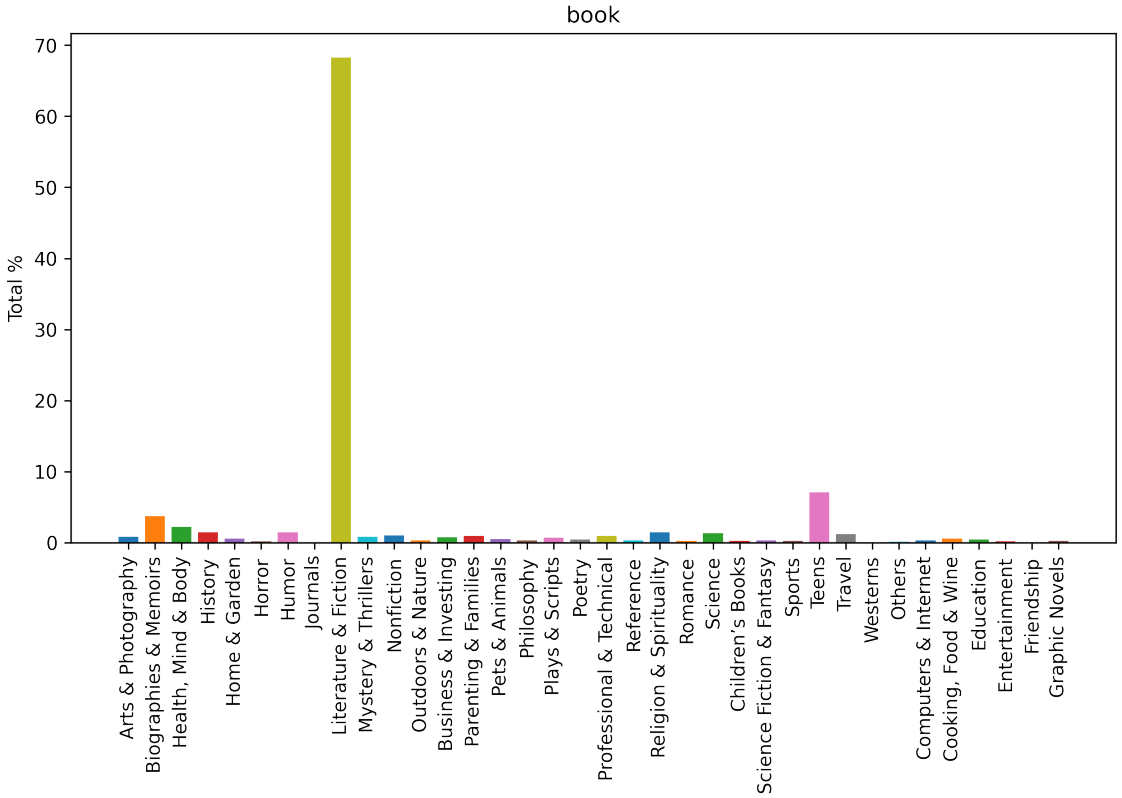



Figure 1: Genre spectrum of book ratings, measured in percentage points. The genre *Literature & Fiction* is very predominant, with a distribution of almost 70%. This is because most books shared through BookCrossing fall under this genre.

date of publication, publisher and URLs to cover images for the book from Amazon.

- *BX-Book-Ratings*: This is the most important table, as it contains the user profiles in the form of rows with a user ID, book ISBN and rating between 0 and 10, whereas 0 means the rating was taken implicitly and 1 to 10 explicitly. A user profile consists of one or more of such rows, essentially creating a list of books a user has interacted with combined with a rating of how much the user liked the book (or in the implicit case, just denotes the user has interacted with the book).

Additionally, we complemented the dataset with an external table from Ruchi Bhatia on Kaggle[3] which adds 3 auxiliary columns: Category, Language and Summary. The

[3] `https://www.kaggle.com/ruchi798/bookcrossing-dataset`

one column most important to us is the Category column, as it supplements exactly the information needed for our experiments.
Finally, we applied a few preprocessing steps to the dataset, namely we excluded all user profiles with less than 50 ratings and more than 2000 and set all implicit ratings to the median of 5. The final subset we use for our experiments consists of 577,414 ratings for 223,607 books from 3000 users, 1000 per user group (Low, Medium, High), with an average profile size of 192 items.

**LastFM**

The LFM-1b dataset[4] was acquired through the effort of Markus Schedl [Schedl, 2016] who proceeded to obtain a large dataset of over 1 billion listening events through the LastFM API. LastFM is an online music tracking and recommendation service[5] which allows you to link your favourite music streaming services to it, and LastFM in turn keeps track of what you are listening and gives you various stats in real time, weekly reports and access to your listening history for example. Furthermore, it uses your listening history to recommend users new music and events. LastFM also makes it possible to find users with similar tastes in music and connect with them, adding a social aspect to its profile.
The LFM-1b dataset was fetched between January 2013 and August 2014 and contains 1,088,161,692 listening events from 120,322 users, rating songs of 3,190,371 artists. The users are 72% male and 28% female and come mainly from the US (19%), Russia (9%), Germany (8%), Ukraine (8%), Poland (8%) and Brazil (7%). It is composed of 6 tables:

- *users*: A table containing the unique IDs of users, age and gender, the amount of times a user has listened to songs and the timestamp when each user registered for this service.

- *users additional*: This table contains additional data of each user, such as the novelty score according to [Müller et al., 2011] which defines a percentage value of how many new artists were listened to, and was averaged over time periods of 1, 6 and 12 months, the mainstreaminess score, averaged over 1 and 6 months, a year and total time. Additionally, the table also contains the total

[4] `http://www.cp.jku.at/datasets/LFM-1b/`
[5] `https://www.last.fm/`

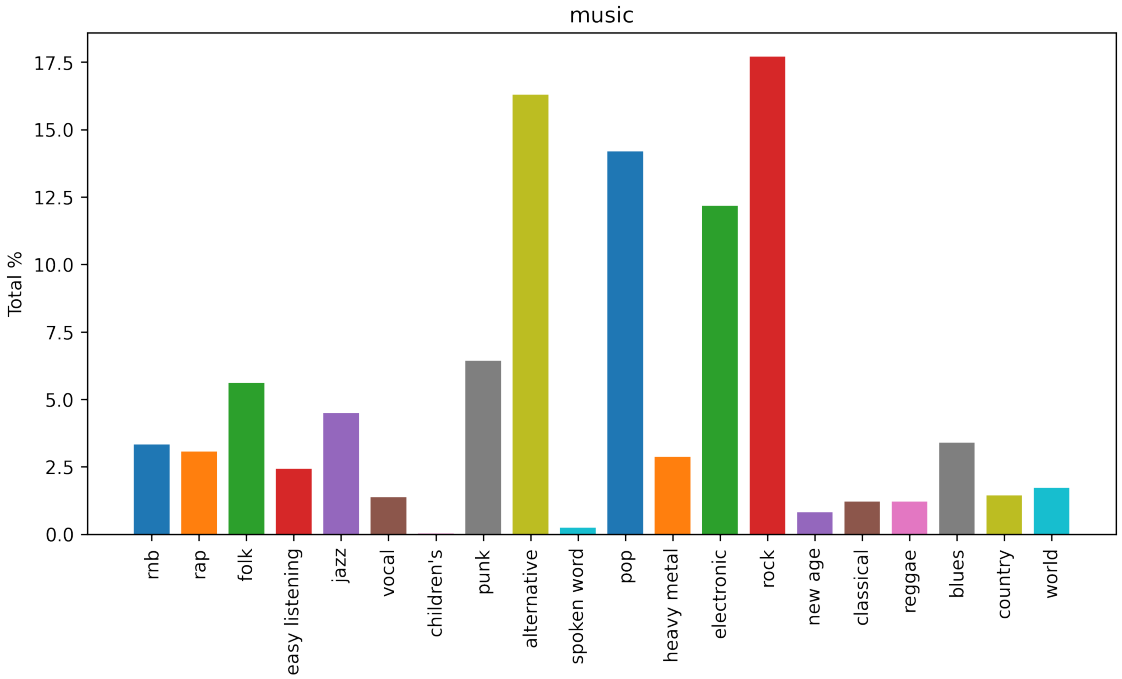



Figure 2: The percentage amount of listening events pertaining to each genre. As a listening event can have multiple genres, the total count exceeds the number of listening events. Takes only the listening events of the subset of the LFM-1b dataset into consideration.

count of listening events, distinct tracks and artists and the average number of listening events per week pertaining to each user.

- *artists*: Holds a unique ID and name for each artist.
- *albums*: A collection of all albums with ID, name and artist ID for each item.
- *tracks*: This table lists tracks with an ID, name and the artist ID of its creator.
- *LEs*: The table containing all listening events of all users. A listening event is composed of a user ID, artist ID, album and track ID and the timestamp it was generated.

For our subsequent experiments we use a subset[6] of the actual LFM-1b dataset which takes the work of Kowald et al. [Kowald et al., 2020] into consideration, creating three user groups for each 1000 users, split into low, medium, and high

[6] `https://zenodo.org/record/3475975`

mainstream users, reducing the amount of events to 1,755,361 and 352,805 music artists. This dataset does not include explicit ratings by the users, therefore we will use the number of times a user has listened to a song as implicit ratings, normalized between 1 and 1000. After cleaning up the subset, we are left with 1,417,791 events on 131,188 songs, with an average profile size of 472 items per user.

**MovieLens**

The MovieLens datasets[7] are widely known and were first released in 1998 by F. Maxwell Harper and Joseph A. Konstan at the University of Minnesota [Harper and Konstan, 2015]. They are collected through MovieLens, a movie recommendation service, and datasets with multiple sizes exist, such as 100 thousand, 1 million, 10 and 20 million. MovieLens is a research project consisting of an online web-service run by GroupLens Research at the University of Minnesota and uses a collaborative filtering approach to recommend movies to its users based on their ratings of prior movies they already watched. Additionally, it provides information pertaining to movies, such as average ratings, genres the movies confers to, images, a short description, tags, etc..
For our research purposes, we will use the 1M dataset with 1,000,209 user ratings of 6,040 users about 3,706 movies. The dataset was collected between April 2000 and February 2003 and has a rating density of $4,47\%$ with ratings between 1 and 5 stars. The MovieLens 1M is composed of 3 tables:

- *users*: The users' table contains demographic information such as gender, age and occupation about the users, which was provided by the user base. Each user is identifiable through a unique ID.

- *movies*: This table includes a simple set of information about each movie, consisting of only the title and a list of genres, identifiable with an ID. The titles are exactly the same as the titles provided by the IMDb, with the additional year of release. Be advised that some IDs may not correspond to an actual movie, as they were related to duplicates and/or test entries.

- *ratings*: The ratings' table lists the rating events of 6040 users about 3952 movies. Each rating is between 1 and 5 stars, with only whole stars increments,

[7] `https://grouplens.org/datasets/movielens/1m/`

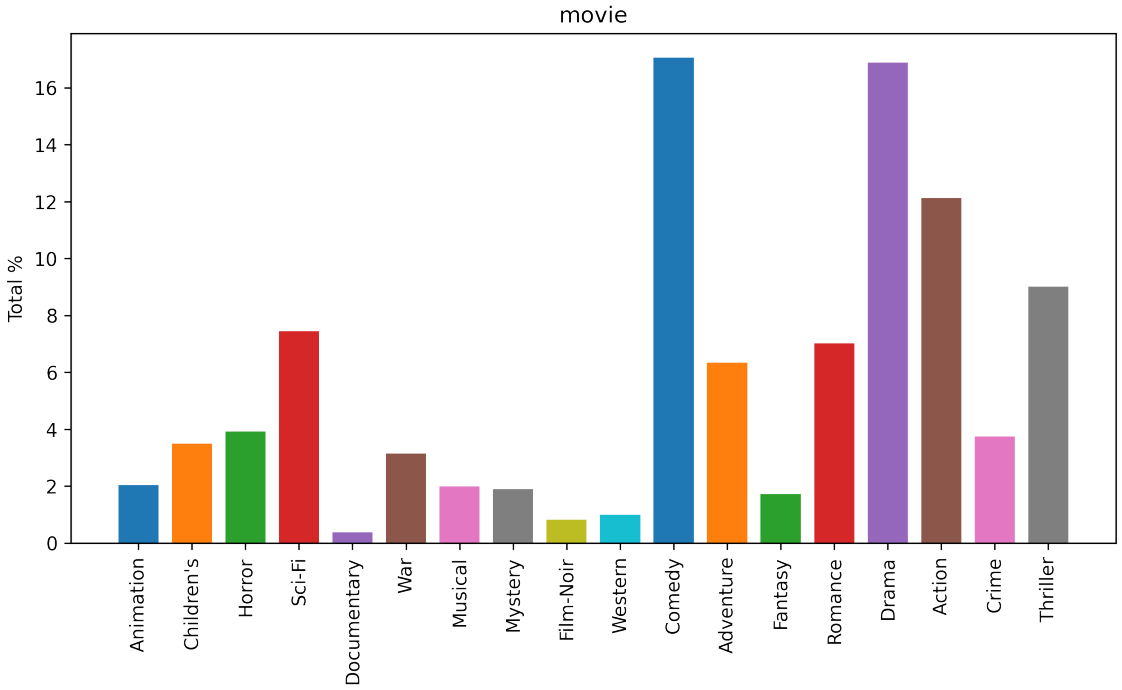



Figure 3: Genre distribution of movie ratings from the MovieLens dataset subset. As seen, most ratings fall under the genres *Comedy*, *Drama* and *Action*, which makes them the most popular in user profiles.

and have an associated timestamp. Each user has at least rated 20 movies.

This dataset is already very complete and is understandably very popular in the research community. For our aims, we will restrict the threshold for the minimal and maximal number of ratings per user to 50 and 2000, respectively, to have a similar set of data as the other datasets. The resulting subset consists of 675,610 ratings about 3667 movies, from 3000 users, split in 3 groups (Low, Medium, High) with an average profile size of 225.

### Yahoo! Movies

The Yahoo! Movies dataset[8] is provided as part of the Yahoo! Research Alliance Webscope program and is based on data generated by Yahoo! Movies on or before November 2003, with some modifications and additions by Yahoo! Research. Yahoo! Movies is part of the Yahoo! network since May 1998 and provides information

[8] `https://webscope.sandbox.yahoo.com/catalog.php?datatype=r`

about movies, such as release dates, trailers, box office and theatre information. It capacitates the user to read reviews of others and write their own, get personalized recommendations for movies, purchase movie tickets and create and share lists of their favourite movies.

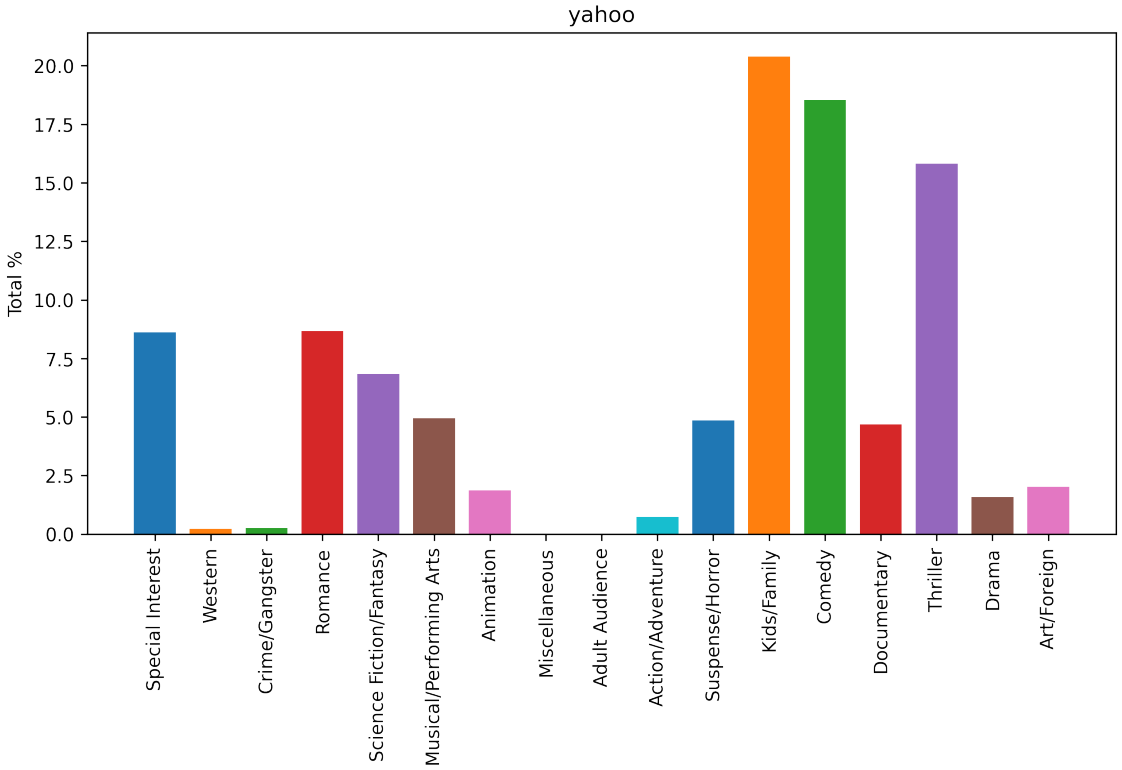



Figure 4: Genre distribution of movie ratings from the Yahoo! Movies dataset subset. Most popular is *Kids/Family*, followed by *Comedy* and *Thriller*. Compared to the MovieLens dataset, Yahoo! Movies viewers are a lot less interested in *Drama* and *Action*.

The Yahoo! Movies dataset consists of multiple tables, including a train and test set, user demographics, item descriptions and mapping tables for MovieLens and EachMovie:

- *movie-ratings-train*: This table consists of 211,231 ratings from 7,642 users rating 11,915 movies. The rating ranges from 1(F) to 13(A+) or converted to 1 to 5. All users have rated at least 10 items and all items are rated by at least one user, with the average number of ratings per user being 27.64. Unfortunately, the rating density is quite low, with only 0.23%. Each rating

is composed of a user ID, movie ID, the rating and its converted counterpart.

- *movie-ratings-test*: The test table is a subset of the training table. The test data was gathered chronologically after the training data and contains 2,309 users, 2,380 items, and 10,136 ratings. All test users and items are also included in the training set, and the average ratings per user is 4.39. Each rating is comprised exactly the same as the training set.

- *user-demographics*: This table contains simple demographic information about each user, comprised of the year of birth and gender, identifiable through the unique user ID.

- *movie-content-descr*: The content description table lists all movies with their descriptive content information. We will mention a few important fields, as the content of each movie consists of a multitude of fields and are not important for our experiments, such as the movie ID identifying each movie, title, release date, list of genres, the average rating of this item among users in the training data, the number of users in the training data who rated this item and the global non-personalized popularity (GNPP).

- *mapping-to-movielens*: A mapping of the Yahoo! Movies dataset movie IDs to the MovieLens movie IDs. Each row consists of the Yahoo! Movies movie ID, MovieLens movie ID, and the movie title. Yahoo! states, that the mapping may be incomplete or incorrect, this has to be taken into consideration.

- *mapping-to-eachmovie*: Similar to the MovieLens mapping, this mapping links the Yahoo! Movies dataset movie IDs to EachMovie IDs. The EachMovie dataset was created by the Digital Equipment Corporation's Systems Research Center containing 2,811,983 ratings entered by 72,916 users for 1628 different movies, but as of October 2004, the dataset has been discontinued. The MovieLens dataset was originally based on this dataset. Each row consists similar to the mapping above of the Yahoo! Movies movie ID, EachMovie movie ID, and the movie title and may have the same issue of incompleteness or incorrectness.

It is without question, that the most important table for us is the table containing the training data. For our purposes, we only require 3000 users, 1000 for each user group. This reduces the dataset to 78,563 ratings for 8438 items, with an average

of 26 items per user profile. We also only use the reduced rating from 1 to 5 for a better comparison to other datasets, and only include users with at least 10 ratings and 2000 at most.

### MyAnimeList

The MyAnimeList dataset[9] was collected by the user CooperUnion on `kaggle.com` and is available for download there. The dataset was acquired from recommendation data at `myanimelist.net`, a large anime and manga database with a recommendation service, where users can create personalized lists to organize and track what titles a user has already seen, and make appropriate recommendations based on the rating values the user assigned to the items in the list.

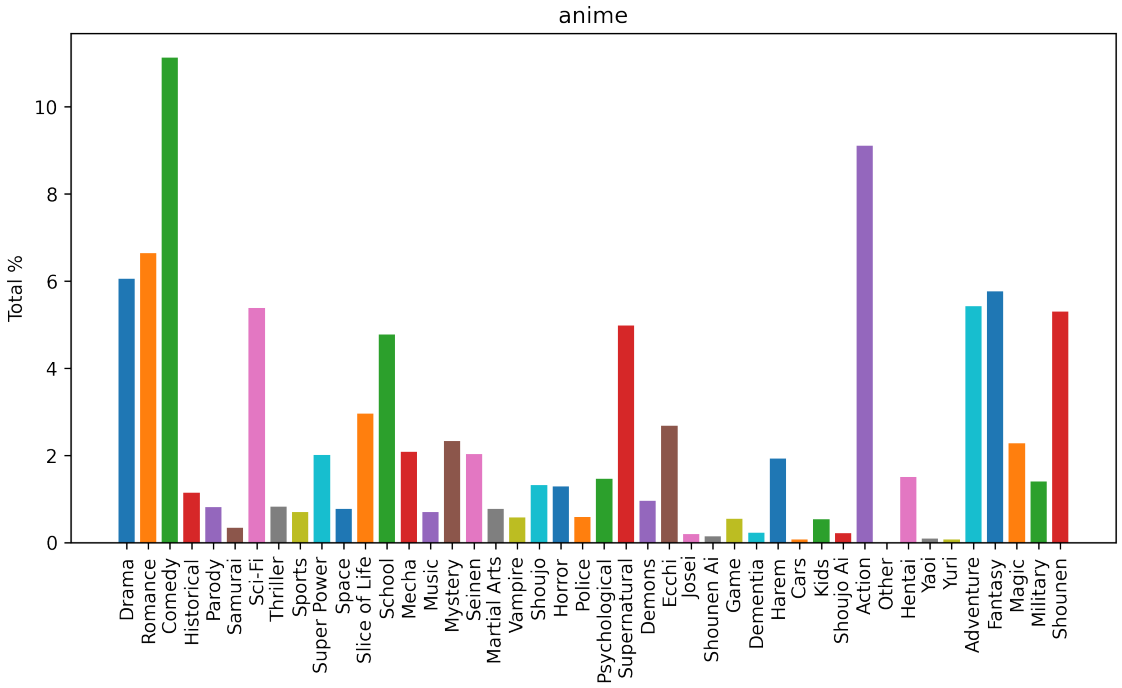



Figure 5: Distribution of genres of anime ratings from the MyAnimeList dataset subset. The most popular genres of this subset are *Comedy* and *Action*.

The dataset provides over 7,8 million ratings from 73,516 users on 12,294 anime. It contains 2 tables:

[9] `https://www.kaggle.com/CooperUnion/anime-recommendations-database`

- *anime*: This table contains some basic information on each anime show. Each anime is identifiable through a unique ID and name and can be categorized to one or more genres. Additionally, columns for the type of show, number of episodes, average ratings given by users and number of member, *i.e.*, how many users have the anime in their list, exist.

- *ratings*: The ratings' table holds the ratings users have assigned to anime. Each user has an assigned ID and can give an explicit rating between 1 and 10, or implicit of -1 to an anime, distinguishable through an assigned anime ID.

Like previous datasets, we will limit the amount of users who reach a rating threshold of at least 50 and at most 2000 ratings, reducing the number of total ratings to 649,814 about 9450 items, with an average profile size of 216. Additionally, implicit ratings will be set to 5. Like all other datasets, the amount of users is exactly 3000, split into 3 groups (Low, Medium, High).

## Algorithms

Probably the most essential component of a recommender system is the underlying algorithm, which generates the predictions for the user. For our experiments, we use a variety of algorithms to discern how much they propagate bias, how accurate their recommendations are and what miscalibration we have to expect using them. A comparison will be made between them to determine the best option for a recommender system. For reproducibility, we will use a Python-based open-source recommendation toolkit called Surpise[10], which provides several algorithms representing various recommendation strategies such as collaborative and content-based filtering or even advanced strategies like non-negative matrix factorization (NMF) and singular value decomposition (SVD). We will limit the number of algorithms to compare in our experiments to 7, namely *Random* as a baseline or guideline, *Most Popular* as an extreme, several collaborative/content-based filtering algorithms such as *UserItemAvg*, *UserKNN*, *UserKNNAvg* and *Co-Custering*, and *NMF* as an advanced strategy.

[10] `http://surpriselib.com/`

### Random

This is probably one of the most simplest algorithms to implement for recommendations, using a pseudo random number generator to select a single or collection of items out of all available items. As mentioned, this algorithm can be used as a baseline for our experiments, as results should be completely random and thus unaffected by any biases existing in the data. This means, that results should have little to no popular bias, not be personalized, meaning highly miscalibrated, and, as the rating is chosen randomly, very inaccurate. For our experiments, we will simply choose $n$ random items per user for their recommendations, with the order of occurrence being the rank of an item.

### MostPopular

The MostPopular algorithm is widely used and part of many simplistic recommender systems. Simplistic, because MostPopular does not recommend personalized items and has therefore no need for any calculations at all, instead simply recommending the *most popular* items in a system. The popularity of an item can be defined in many ways, but in most cases refers to the highest rated or most viewed items. This, of course, has the negative side effect that such recommended items are heavily biased and may not meet everyone's taste, which could lead to users not using the recommendation service or search for a better alternative. For our experiments, we will recommend the top $n$ items with the highest number of ratings to each user, with the descending order of placement in the distribution list being the rank of each item.

### UserItemAvg

It is quite normal for collaborative filtering based recommender systems to see data exhibit large user and item effects, *i.e.*, some users give consistently higher or lower ratings than others and some items receive regularly higher ratings than others. This can be adjusted by taking these effects into consideration by calculating a *baseline estimate*, which remedies such effects:

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i,$$

where we essentially wrap the rank prediction $\hat{r}_{ui}$ into a baseline estimate $b_{ui}$, with the parameters $b_u$ and $b_i$ being the observed deviations of user $u$ and item $i$, respectively, from the average. This algorithm is available as `BaselineOnly` in the Surprise package, and as the name suggests, can be used as a baseline algorithm for collaborative filtering algorithms.

### UserKNN and UserKNNAvg

Both UserKNN and UserKNNAvg are basic collaborative filtering algorithms directly derived from a basic nearest neighbours approach [Fix and Hodges, 1989]. As the name suggests, kNN clusters similar data into $k$ clusters by calculating the distance between a target and all other entries, for example user profiles, and then ranks the distance and returns the $k$ nearest neighbour user profiles. This, of course, can also be done with items, meaning, we could find the $k$ nearest neighbours of a movie, book or song. Interesting for us is the use of a kNN algorithm to predict a rating for an item. The `KNNBasic` class in the Surprise package implements such a use case, with the prediction $\hat{r}_{ui}$ set as:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u, v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} sim(u, v)}$$

for item based kNN, or

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} sim(i, j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} sim(i, j)}$$

for user based kNN. `KNNBasic` merely implements the most basic kNN approach we named UserKNN, whereas the class `KNNWithMeans` additionally takes the mean ratings of each user into account:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} sim(u, v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u, v)}$$

for item based kNN, or

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} sim(i, j) \cdot (r_{uj} - \mu_i)}{\sum_{j \in N_u^k(i)} sim(i, j)}$$

for user based kNN, and will represent UserKNNAvg. For our consequent experiments, we will use the user based option for both algorithms.

### Co-Custering

Co-Custering is a technique developed for dynamic real-time collaborative filtering that can adapt to a rapid increase of new user, item and rating data, like it is happening in a major online recommender system, where techniques such as correlation and SVD based methods are unable to keep up, as they are computationally expensive and can be deployed only in static off-line settings [George and Merugu, 2005]. The `Co-Clustering` class in the Surprise package implements the proposed algorithm in [George and Merugu, 2005] by assigning clusters $C_u$ and $C_i$ to users $u$ and items $i$ and co-clusters $C_{ui}$ to both, respectively. The prediction $\hat{r}_{ui}$ is then made:

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i}),$$

where $\overline{C_{ui}}$, $\overline{C_u}$ and $\overline{C_i}$ are the average ratings of the c-cluster $C_{ui}$, the users $u$ cluster and the items $i$ cluster in that exact sequence. Important to mention is that if the user or item is not known, the prediction is either $\hat{r}_{ui} = \mu_i$ if the user is unknown or $\hat{r}_{ui} = \mu_u$ if the item is unknown. Finally, if both user and item are not known, the prediction is equal to the mean $\mu$.

### NMF

Non-negative Matrix Factorization (NMF) is the process of factorizing a matrix $\boldsymbol{V}$ into two matrices $\boldsymbol{W}$ and $\boldsymbol{H}$, with all three matrices having no negative elements, which makes them easier to investigate. NMF has a huge range of applications, such as astronomy, computer vision, and interesting for us, recommender systems. We will use the `NMF` class of the Surprise package, which implements a collaborative filtering algorithm based on NMF. It follows the implementation described in [Luo et al., 2014], with the prediction $\hat{r}_{ui}$ calculated as follows:

$$\hat{r}_{ui} = q_i^T p_u,$$

or extended with bias,

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u.$$

All factors have to be positive and are optimized through stochastic gradient descent. The algorithm itself is highly dependent on initial values, meaning, both factors for user and item have to be uniformly initialized. The biased version may yield better accuracy, but this can cause an increased likelihood of overfitting, which have to be compensated through the reduction of the number of factors or increase of regularization. For our purposes, we will only use the baseline version, as this should suffice for our experiments.

## Evaluation Metrics

We will use three evaluation metrics in this thesis, namely the *Mean Absolute Error* (MAE), representing the accuracy of the recommendation result, the *Miscalibration metric*, which measures the discrepancy of the category spectrum between user profile and recommendation, and the *Popular Lift*, which measures the popular bias in a recommendation. This section will talk about the origin of these metrics, how they are calculated, and what features and characteristics they possess to be of interest for our research questions.

### Mean Absolute Error

The Mean Absolute Error (MAE), as the name suggests, is a measure in statistics to define the average error between a pair of observations, or, in the case of recommender systems, between a prediction and the true value in the context of item ranking:

$$MAE = \frac{\sum_{\hat{r_{ui}} \in \hat{R}} |r_{ui} - \hat{r_{ui}}|}{|\hat{R}|},$$

where $\hat{R}$ is the number of ratings, $\hat{r_{ui}}$ is the prediction, and $r_{ui}$ the true value. The MAE of a recommendation gives a clearer view on how accurate, or better said, *inaccurate*, a system is, as the MAE actually displays the inverse of the accuracy and instead shows the *error* of a predicted rating, meaning, the higher the error rate, the higher the inaccuracy of a recommendation. A special characteristic of this metric is how it is unaffected to the bias of extreme error values, in other words, prediction outliers have a small effect on the overall accuracy of a prediction. To place a higher importance on outliers, we can look at the related Root Mean Squared Error (RMSE), which disproportionately penalizes large errors as the error

term is squared. But our focus does not lie on the influence of outliers on the overall prediction and thus for our purposes, the MAE should suffice.

### Miscalibration Metric

Based on the work of *Lin et al.* [Lin et al., 2020], which uses a metric proposed by *Steck* [Steck, 2018], this thesis makes use of said miscalibration metric. To find out how much a system is miscalibrated, we have to compare two essential distributions based on the distribution of categories (genres) $c$ for each item $i$ stated as $p(c|i)$:

- $p(c|u)$: The distribution over categories $c$ of item set $\mathcal{H}_u$ which the user $u$ has interacted with.
$$p(c|u) = \frac{\sum_{i \in \mathcal{H}_u} w_{u,i} \cdot p(c|i)}{\sum_{i \in \mathcal{H}_u} w_{u,i}},$$
where $w_{u,i}$ is the weight of an item $i$ in the set $\mathcal{H}_u$, in our case a star ranking or how often an item was clicked (played). But it is also possible to use other ranking strategies like most recent play-time or overall watch-time.

- $q(c|u)$: The distribution over categories $c$ of item set $\mathcal{I}_u$ which were recommended to the user $u$.
$$q(c|u) = \frac{\sum_{i \in \mathcal{I}_u} w_{r(i)} \cdot p(c|i)}{\sum_{i \in \mathcal{I}_u} w_{r(i)}},$$
with $\mathcal{I}_u$ as the set of items which were recommended to the user and $w_{r(i)}$ as the weight specified by the prediction rank of the recommended item list.

Using the two distributions, we can use the Kullback-Leibler (KL) Divergence to compute a divergence metric between $p$ and $q$, states as:

$$KL(p||\hat{q}) = \sum_{c \in C} p(c|u) \log \frac{p(c|u)}{\hat{q}(c|u)},$$

where $p(c|u)$ is our target distribution (the values we want to reach), $C$ is the set of categories and $\hat{q}(c|u)$ is the modified distribution of $q(c|u)$, which is necessary because of the divergence behaviour which occurs when a category $c$ is $q(c|u) = 0$ and $p(c|u) > 0$. To circumvent this, we use:

$$\hat{q}(c|u) = (1 - \alpha) \cdot q(c|u) + \alpha \cdot p(c|u),$$

where we carry on like *Lin et al.* [Lin et al., 2020] and set $\alpha = 0.01$ for our experiments, as $0 < \alpha < 1$, such that $q \approx \hat{q}$.

This metric, as mentioned in both *Lin et al.* [Lin et al., 2020] and *Steck* [Steck, 2018] has several favourable properties which make it a preferable solution for the problem at hand:

- The metric is zero in case of a perfect calibration, our for our purposes has no miscalibration: $p(c|u) = \hat{q}(c|u)$.
- It diverges faster if the user has little interaction with a category, i.e., the user interacts with an item 4% of the time, but the item only gets recommended 3% will have a higher metric in contrast to high interaction count like 40%, but the item gets only recommended 39% of the time.
- Based on the last behaviour, the metric is more sensitive if the item is recommended less than it was interacted with, meaning if the interaction count was 20%, a recommendation of 21% provides a better metric than a recommendation of 19%.

These properties ensure that even items with a small interaction count are not neglected, which leads to a more diverse recommendation list.

### Popularity Lift

A good way to measure popular bias in recommendations is through the use of the Popular Lift metric. Popular bias, as already mentioned in an earlier chapter, is the effect of over-recommending popular items due to their high number of ratings they possess, skewing the overall rating data in their favour. Popular items, as the name suggests, are popular among the user base and therefore algorithms, like MostPopular, take advantage of this fact. But this of course does not cater to everyone's needs, as some users prefer a more niche subset of items, which have commonly fewer ratings than popular items. A good recommender system must be able to address the needs of such users as well. To determine which algorithms fit in such a recommender system, *i.e.*, algorithms which propagate and amplify the already existing disparity in the rating data the least, we introduce a metric to measure the propagated and, and in term, amplified popularity bias, called *Popularity Lift*. This

is done by splitting the user base into different user groups $G$ and calculating the average item popularity for the groups' of user profiles and their recommendations:

$$GAP_p(G) = \frac{\sum_{u \in G} \frac{\sum_{i \in \Gamma_u} \theta_u(i)}{|\Gamma_u|}}{|G|},$$

for user profiles and

$$GAP_q(G) = \frac{\sum_{u \in G} \frac{\sum_{i \in \Lambda_u} \theta_u(i)}{|\Lambda_u|}}{|G|},$$

for recommendations, where $\theta_u(i)$ is the value of popularity for an item $i$, *i.e.*, the ratio of users who rated the item $i$, $\Gamma_u$ and $\Lambda_u$ being the item sets of both user profiles $p$ and recommendations $q$, respectively. The Popularity Lift (PL) of a group $G$ is then defined as:

$$PL(G) = \frac{GAP_q(G) - GAP_p(G)}{GAP_p(G)}.$$

A positive Popularity Lift indicates an amplification of the popularity bias propagated through an algorithm, a negative value instead appears as a result of a lesser concentration of popular items in the recommendation than the concentration in the user profile. An ideal value for the Popularity Lift is 0, meaning, the algorithm neither reduces nor amplifies the propagated popular bias of the rating data. The Popularity Lift does not measure the already existing popular bias in the rating data, only its amplification through an algorithm.

## Evaluation Protocol

The focus of this thesis lies on how different evaluation metrics compare to each other in context of different datasets and algorithms used for recommendations. To keep a certain level of standardization of our results, it is imperative to keep the setup as similar as possible, which means we will use a standard 80/20 split between training and test set and standard hyperparameters for our algorithms, which range from baseline, KNN-based to Matrix Factorization-based approaches, which was discussed in the previous chapter. Similar, we will analyse 5 different datasets by extracting 3000 users from each user-base, who represent 1000 users with low, 1000 users with medium and 1000 users with high mainstreaminess or item popularity,

respectively.
To determine, if one or more particular users receive less calibrated recommendation results (RQ1), we will use the discussed user groups of low, medium, and high mainstreaminess/item popularity and compare them over different algorithms. We expect the low mainstream users, *i.e.*, niche users, to have particularly miscalibrated results and users with high mainstreaminess to receive the best recommendation results, based on miscalibration. We also want to observe how much the dataset can influence the result, therefore we will run this experiment on 5 different datasets.
Additionally, we want to see if there is a connection between MAE and miscalibration, *i.e.*, if MAE and miscalibration correlate in any way (RQ2). Hyperparameters are used to tweak an algorithm, increasing its accuracy, which we measure with the MAE. Our focus does not lie on increased accuracy, but more so on the difference in accuracy results between different user groups. We want to find out, if MAE experiences a similar behaviour in its measurements as the values of the miscalibration metric. This can be done by simply comparing both MAE and miscalibration with each other and see, if they return similar differences in their results.
Finally, we take a closer look at the popularity bias and if it has any influence on miscalibration (RQ3). Like already mentioned, we can measure popularity bias through the Popularity Lift metric, which measures the deviation from the average amount of popular items contained in a user profile. We compare this metric to the values of the miscalibration metric and look if correlations between the two metrics appear. We can, similarly to the experiments above, split users into the same user groups and identify if they exercise a coinciding behaviour. For both RQ2 and RQ3, we can also calculate the Pearson coefficient between metrics for each dataset, which gives us a numerical value representing the linear relationship between metrics.

# Experiments and Results

To evaluate the performance of the recommendation algorithms we stated above, we conducted several experiments. In these experiments we calculated the evaluation metrics *MAE*, *Miscalibration* and *Popularity Lift*, consecutively. We executed each algorithm on all five datasets we specified, and then compared their results.

## Setup

We conducted All experiments on one computer using Jupyter[11] Notebook for Python[12], a web-based and interactive development tool, which allows us to run Python code step by step, making it possible to see the result of each individual calculation step. As already mentioned, we used `Surprise`, a recommendation library, to calculate the predictions for our users using *Baseline*, *UserKNN*, *UserKNN with means*, *NMF* and *Co-Clustering*. *Random* and *MostPopular* were calculated through `numpy`[13], a computing library, which provides several useful functions which provide us with the necessary functionality to implement these algorithms.
We did all calculations *per* dataset, meaning, we executed all calculation steps sequentially for one dataset and after collecting the results, were executed again for the next dataset.

## Results

We can split the experiment into three parts based on the calculation of each evaluation metric: *MAE*, *Miscalibration* and *Popularity Lift*. Additionally, we assigned users of each dataset to one of three user groups, based on mainstreaminess/item popularity of their profile: *LowPop*, *MedPop* or *HighPop*. We will use the abbreviation *Pop*, as only the LastFM dataset uses mainstreaminess instead of item popularity. Finally, we did the calculations on the same setup throughout the experiments, which reduces external influences such as system bias to a minimum.

In Table 2 we can see the immediate results of our calculations. We did not calculate the MAE for both *Random* and *MostPopular*, as they are not deployed through the Surprise library, and are instead based on `numpy` functions. If we look at the results a bit more in detail, we find many similarities between datasets:

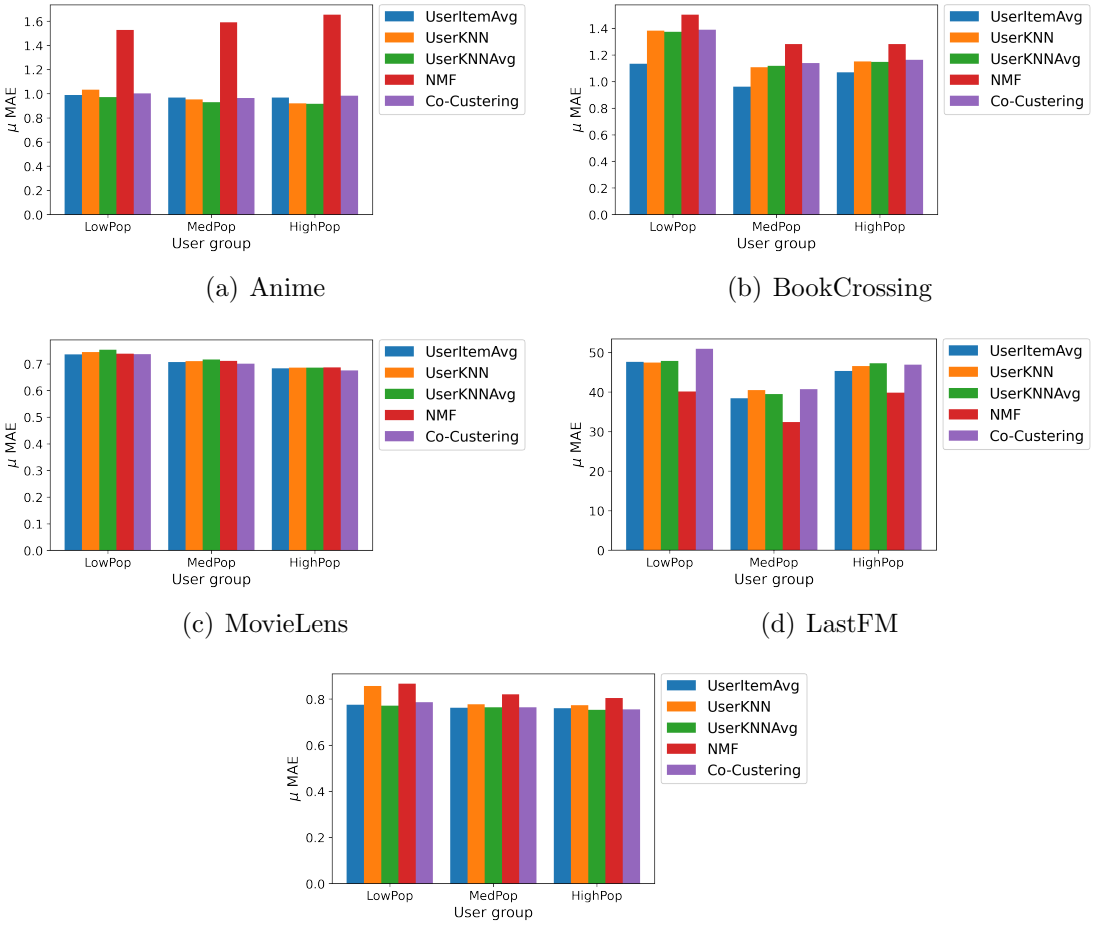
[11] `https://jupyter.org/`
[12] `https://www.python.org/`
[13] `https://numpy.org/`

| | *Users* | Random | | | MostPopular | | | UserItemAvg | | | UserKNN | | | UserKNNAvg | | | NMF | | | Co-Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Metrics*** | | *MAE* | *MC* | *PL* | *MAE* | *MC* | *PL* | *MAE* | *MC* | *PL* | *MAE* | *MC* | *PL* | *MAE* | *MC* | *PL* | *MAE* | *MC* | *PL* | *MAE* | *MC* | *PL* |
| **Anime** | *LowPop* | – | **1.538** | -0.704 | – | 1.154 | 5.653 | 0.990 | 0.965 | 1.112 | 1.032 | 0.922 | 0.805 | 0.973 | 0.903 | 0.605 | **1.528** | 0.932 | 0.833 | 1.002 | 0.911 | 1.114 |
| | *MedPop* | – | 1.585 | -0.842 | – | 0.737 | 2.388 | **0.968** | 0.734 | 0.413 | 0.952 | 0.723 | 0.307 | 0.931 | 0.719 | 0.262 | 1.592 | 0.714 | 0.341 | **0.963** | 0.708 | 0.431 |
| | *HighPop* | – | 1.644 | **-0.889** | – | **0.629** | **1.490** | 0.968 | **0.646** | **0.144** | **0.920** | **0.628** | **0.108** | **0.917** | **0.638** | **0.096** | 1.653 | **0.632** | **0.121** | 0.982 | **0.632** | **0.157** |
| **BookCrossing** | *LowPop* | – | 2.906 | -0.747 | – | 2.182 | 34.144 | 1.134 | 2.426 | 0.543 | 1.383 | 2.522 | 0.088 | 1.373 | 2.479 | 0.201 | 1.502 | 2.617 | -0.152 | 1.389 | 2.472 | 0.125 |
| | *MedPop* | – | 2.523 | -0.884 | – | 1.502 | 14.647 | **0.961** | 1.586 | 0.516 | **1.108** | 1.654 | **0.008** | **1.118** | 1.648 | **0.029** | **1.281** | 1.727 | **-0.156** | **1.140** | 1.654 | **0.001** |
| | *HighPop* | – | **2.335** | **-0.927** | – | **1.009** | **8.759** | 1.069 | **1.152** | **0.290** | 1.151 | **1.173** | 0.066 | 1.147 | **1.181** | 0.068 | 1.282 | **1.183** | 0.015 | 1.163 | **1.189** | 0.049 |
| **MovieLens** | *LowPop* | – | **1.249** | -0.557 | – | 1.219 | 2.988 | 0.736 | 0.779 | 0.697 | 0.744 | 0.756 | 0.653 | 0.753 | 0.777 | 0.536 | 0.738 | 0.785 | 0.573 | 0.736 | 0.751 | 0.711 |
| | *MedPop* | – | 1.311 | -0.686 | – | 1.009 | 1.925 | 0.707 | 0.715 | 0.415 | 0.710 | 0.689 | 0.389 | 0.717 | 0.692 | 0.334 | 0.711 | 0.714 | 0.361 | 0.701 | 0.699 | 0.426 |
| | *HighPop* | – | 1.358 | **-0.749** | – | **0.825** | **1.289** | **0.683** | **0.633** | **0.238** | **0.686** | **0.617** | **0.227** | **0.686** | **0.620** | **0.205** | **0.686** | **0.628** | **0.217** | **0.675** | **0.622** | **0.242** |
| **LastFM** | *LowPop* | – | 1.306 | -0.940 | – | 1.084 | 6.261 | 47.644 | 0.517 | **1.309** | 47.461 | 0.563 | **0.221** | 47.890 | 0.536 | **0.118** | 40.144 | 0.592 | **0.025** | 50.925 | 0.522 | **0.724** |
| | *MedPop* | – | **1.185** | -0.943 | – | 0.911 | 5.735 | **38.439** | 0.473 | 1.621 | **40.484** | 0.522 | 0.282 | **39.505** | 0.520 | 0.156 | **32.401** | 0.585 | 0.031 | **40.744** | 0.505 | 1.014 |
| | *HighPop* | – | 1.203 | **-0.953** | – | **0.843** | **4.774** | 45.357 | **0.407** | 1.361 | 46.555 | **0.487** | 0.292 | 47.284 | **0.465** | 0.171 | 39.876 | **0.518** | 0.294 | 46.915 | **0.434** | 0.966 |
| **Yahoo! Movies** | *LowPop* | – | **1.678** | -0.920 | – | 1.601 | 7.572 | 0.775 | **1.405** | **-0.020** | 0.857 | **1.411** | **-0.043** | 0.771 | **1.404** | **-0.035** | 0.866 | **1.408** | **-0.033** | 0.786 | **1.412** | **-0.038** |
| | *MedPop* | – | 1.876 | -0.980 | – | 0.886 | 1.081 | 0.763 | 1.599 | -0.015 | 0.777 | 1.600 | -0.016 | 0.764 | 1.599 | -0.015 | 0.821 | 1.602 | -0.015 | 0.765 | 1.599 | -0.016 |
| | *HighPop* | – | 1.874 | **-0.981** | – | **0.879** | **1.074** | **0.760** | 1.563 | -0.017 | **0.773** | 1.563 | -0.017 | **0.754** | 1.564 | -0.017 | **0.805** | 1.565 | -0.017 | **0.755** | 1.564 | -0.017 |

Table 2: Calculation results of *MAE*, *Miscalibration (MC)* and *Popularity Lift (PL)* for all user groups per dataset for each algorithm. The highlighted values are always the smallest of each set.

### MAE

Looking at the MAE, we can see that it performs better for MedPop and HighPop users for all datasets and algorithms, which means MedPop and HighPop users receive more accurate recommendations than LowPop users. Additionally, to Table 2, Figures 6 and 7 provide comparisons of MAE differences among algorithms and datasets. Looking at the figures, we see a rather diverse behaviour between datasets and algorithms. In the *Anime* dataset, the MAE for the factorization-based algorithm stands out the most, as it is the only one which rises with a higher mainstreaminess, whereas the others drop. Moreover, its value is overall $40\% - 60\%$ higher than the rest. The neighbourhood-based methods on the other hand seem to perform the best. For *BookCrossing*, similar to the *Anime* dataset, the worst performing algorithm is again the factorization-based *NMF*. Contrary to the previous dataset, we can see the MedPop user group performing slightly better than the HighPop users, with the baseline algorithm *UserItemAvg* having the best results. The *MovieLens* dataset shows the most uniform values between datasets, with the *Co-Clustering* algorithm performing the best and the neighbourhood-based methods the worst. In this dataset, we see a slight but steady decrease in MAE the higher the mainstreaminess gets. Similar to *BookCrossing*, the *LastFM* dataset performs best for the MedPop group, this time even more clearly. But contrary to *BookCrossing*, we observe the then worst performing *NMF* now bring about the best results between algorithms, with *Co-Clustering* and neighbourhood-based methods performing the worst. Lastly, looking at the *Yahoo! Movies* dataset, we see a similar scenario in the *MovieLens* dataset, with the MAE reducing as the mainstreaminess gets higher. We observe on average higher values for *UserKNN* and *NMF*, with *UserKNNAvg*




(a) Anime

(b) BookCrossing

(c) MovieLens

(d) LastFM

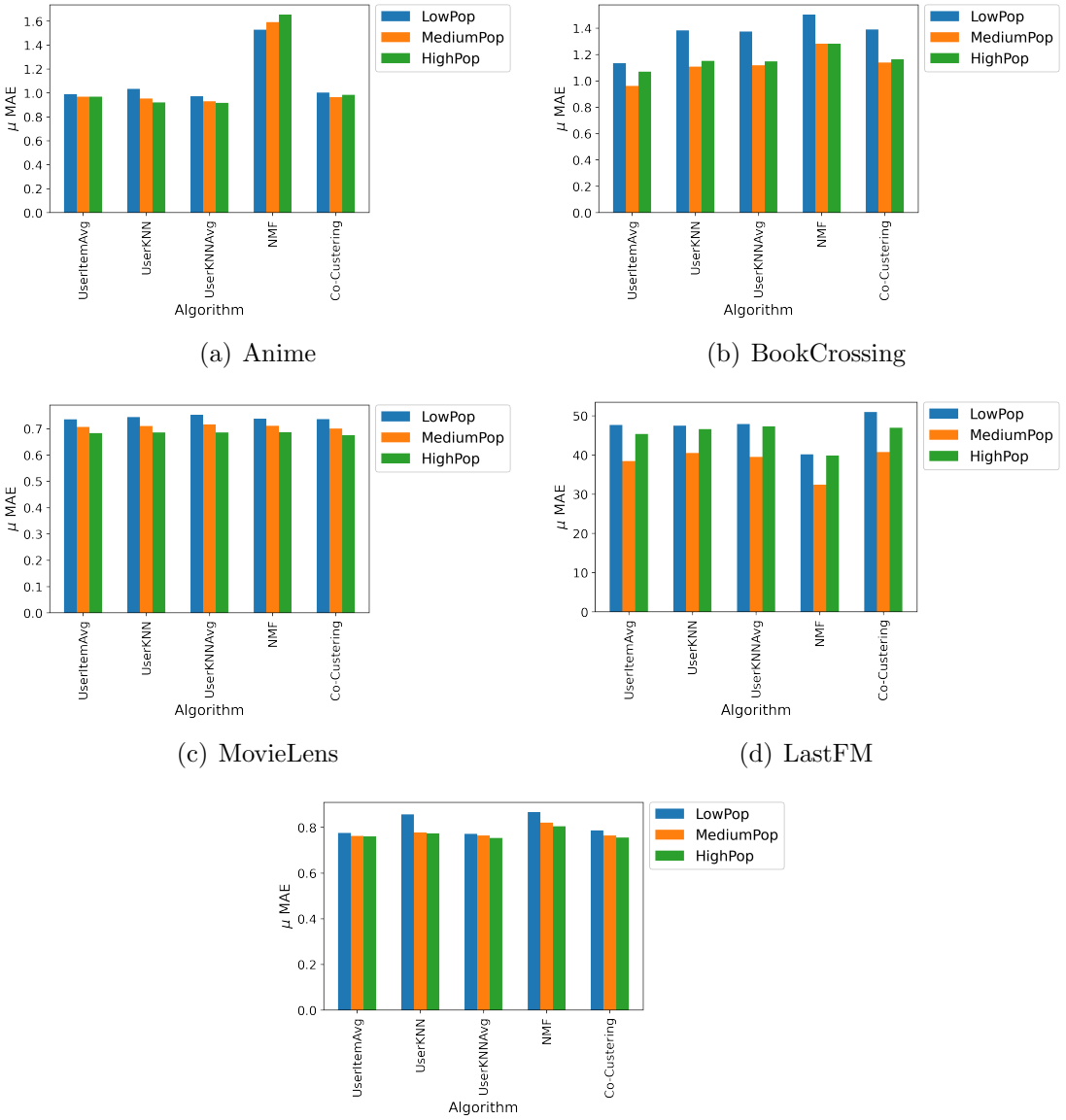
(e) Yahoo! Movies

Figure 6: MAE of each dataset, algorithm, and user group. The bars are grouped by user group and display the values for the algorithms *UserItemAvg*, *UserKNN*, *UserKNNAvg*, *NMF* and *Co-clustering*.

performing the best. For our purposes it is enough to look at the relative interval of values, as due to the nature of the MAE, values can differ greatly from dataset to dataset, *i.e.*, *LastFM*, and as we are deploying algorithms with their standard hyperparameters, this metric can of course be optimized further, but which is not the purpose of this thesis.

### Miscalibration Metric

The Miscalibration metric shows even clearer results, with the exception of the Yahoo! Movies dataset, all the other datasets display the lowest values for High-




(a) Anime

(b) BookCrossing

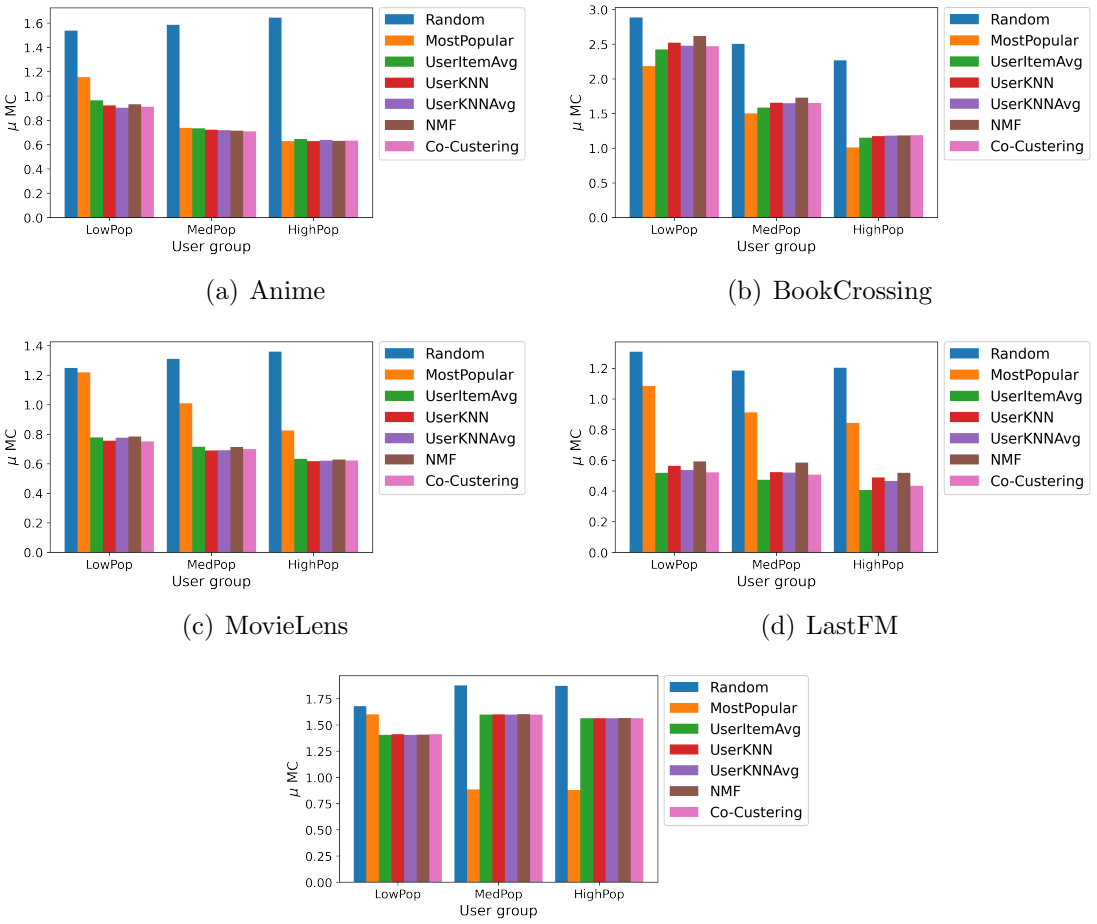
(c) MovieLens

(d) LastFM

(e) Yahoo! Movies

Figure 7: MAE of each dataset, algorithm, and user group. The bars are grouped by algorithms and display the values for each user group: *LowPop*, *MedPop* and *HighPop*.

Pop users. Miscalibration measures the inconsistency between the genre spectrum of user profiles and their recommendations, meaning, HighPop users have the most calibrated recommendations of all groups. Figures 8 and 9 provide us with benchmarks between the individual datasets. Starting with the *Anime* dataset, we can recog-




(a) Anime

(b) BookCrossing

(c) MovieLens

(d) LastFM

(e) Yahoo! Movies

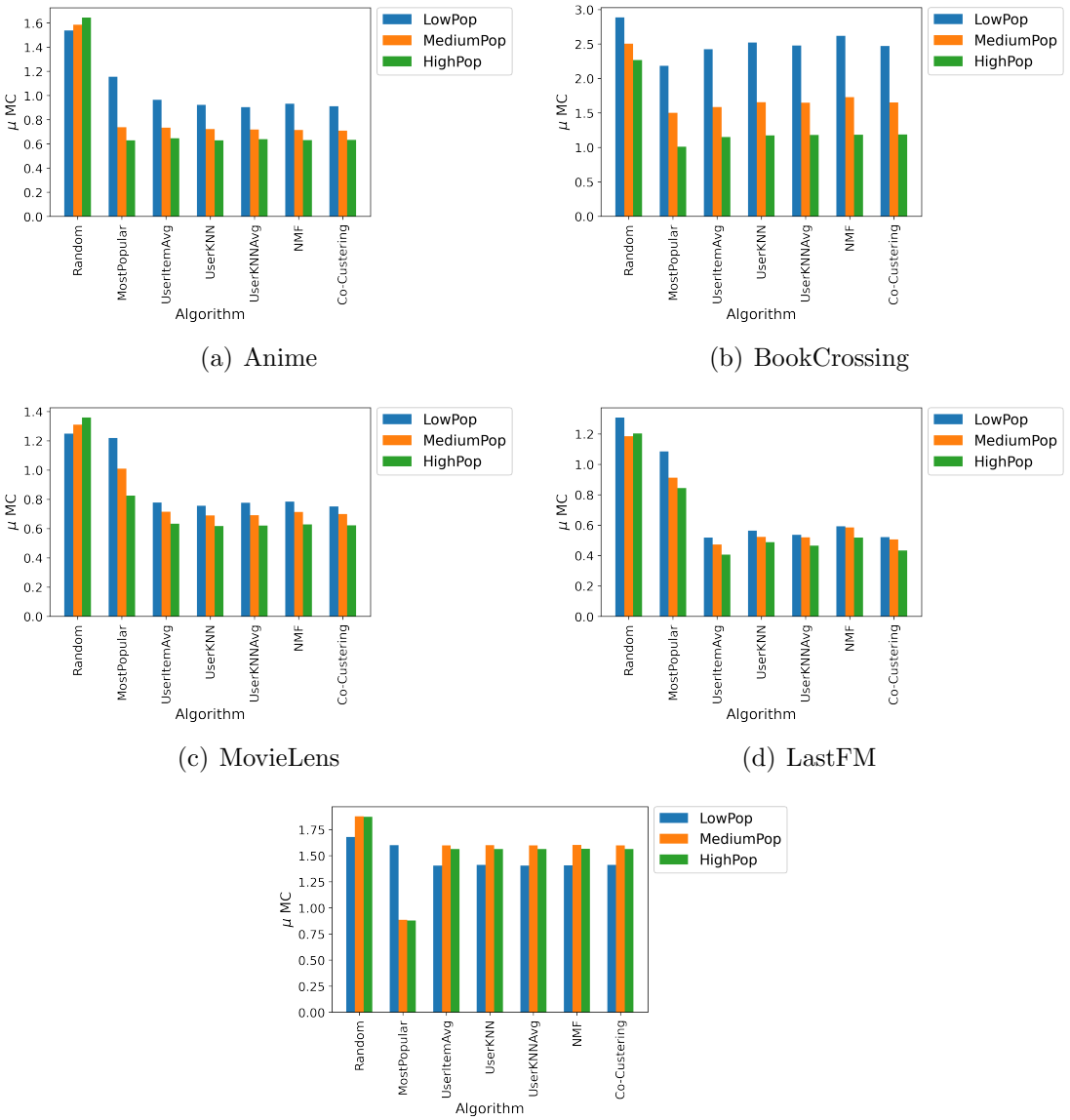
Figure 8: Miscalibration of each dataset, algorithm, and user group. The bars are grouped by user group and display the values for the algorithms *UserItemAvg*, *UserKNN*, *UserKNNAvg*, *NMF* and *Co-clustering*.

nize very similar values between algorithms, with MostPopular performing slightly worse for LowPop users than the rest. The best values can be found for HighPop users and the worst for LowPop users. *BookCrossing* has a similar behaviour, with again HighPop users performing the best and LowPop users the worst. Due to the genre distribution of this dataset (almost 70% for *Literature & Fiction*), the best performing algorithm for this dataset is actually MostPopular, whereas the factorization-based algorithm *NMF* performs the worst and the neighbourhood-based methods also performing worse than the baseline. As for the *MovieLens* dataset, we observe a similar behaviour to the *Anime* dataset, with lower values the

higher the mainstreaminess, and MostPopular performing the worst for each user group. Neighbourhood-based algorithms and *Co-Clustering* have the best results. *LastFM* also shows again decreasing values the greater the mainstreaminess, with the baseline algorithm *UserItemAvg* and neighbourhood-based approaches showing the best results and MostPopular again resulting in the least performing way. We can also observe a small spike in value for *NMF* compared to other algorithms. The *Yahoo! Movies* dataset behaves quite differently to the other datasets, with the lowest values found in the LowPop group and the highest in the MedPop group, except for the MostPopular algorithm, which has the opposite behaviour. We measured the best values with the MostPopular algorithm, which is similar to the *BookCrossing* dataset. Additionally, it seems, that neighbourhood-based as well as factorization-based algorithms produce very similar values for this dataset.

**Popularity Lift**

As for the Popularity Lift, it varies greatly on the dataset used. It mostly shows similar to the MAE, lower values for MedPop and HighPop users, with the exception for the LastFM and Yahoo! Movies datasets. This can be allotted to the profile size, as a smaller size leads to smaller values. An additional influence is the deployed algorithm, which has a significant influence on this metric. Depending on the algorithm used, we can see a significant spike, such as MostPopular, which due to its functionality not only propagates, but significantly increases popular bias in recommendations. It appears that all algorithms intensify the popularity bias to varying degrees for the *Anime* dataset, with the MostPopular algorithm having the greatest amplification. Both neighbourhood and factorization-based algorithms perform better than the baseline, with the LowPop user group experiencing the highest degree of amplification and HighPop users the lowest. A very similar scenario can be observed with the *MovieLens* dataset, which, except for having on average slightly lower values than *Anime*, has exactly the same distribution of measurements. Both *BookCrossing* and *Yahoo! Movies* datasets note almost no amplification of popularity bias for all algorithms except MostPopular and Random, for *Yahoo! Movies*, a slight negative increase can be detected. Lastly, the *LastFM* dataset shows very diverse results based on the algorithm which was used. Whereas the worst performing algorithm, which is again the MostPopular algorithm, shows a similar behaviour to other datasets, with the highest value recorded for LowPop users and the lowest for




(a) Anime

(b) BookCrossing

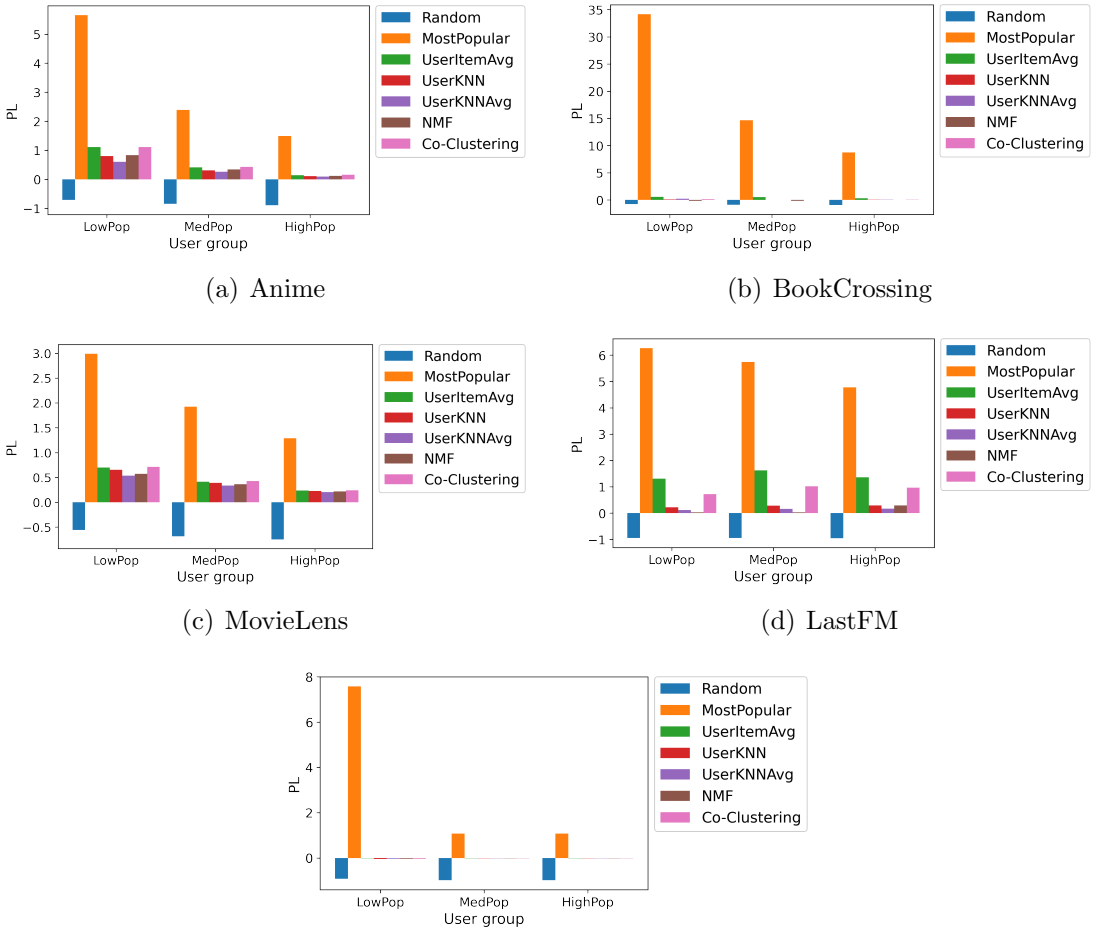
(c) MovieLens

(d) LastFM

(e) Yahoo! Movies

Figure 9: Miscalibration of each dataset, algorithm, and user group. The bars are grouped by algorithms and display the values for each user group: *LowPop*, *MedPop* and *HighPop*.

HighPop users, the baseline and *Co-Clustering* perform the worst with the MedPop user group. The best performing algorithm is *NMF*, with contrary to the results of other datasets, performs best for LowPop users. Overall, we can observe neighbourhood and factorization-based algorithms performing better than baseline options.




(a) Anime

(b) BookCrossing

(c) MovieLens

(d) LastFM

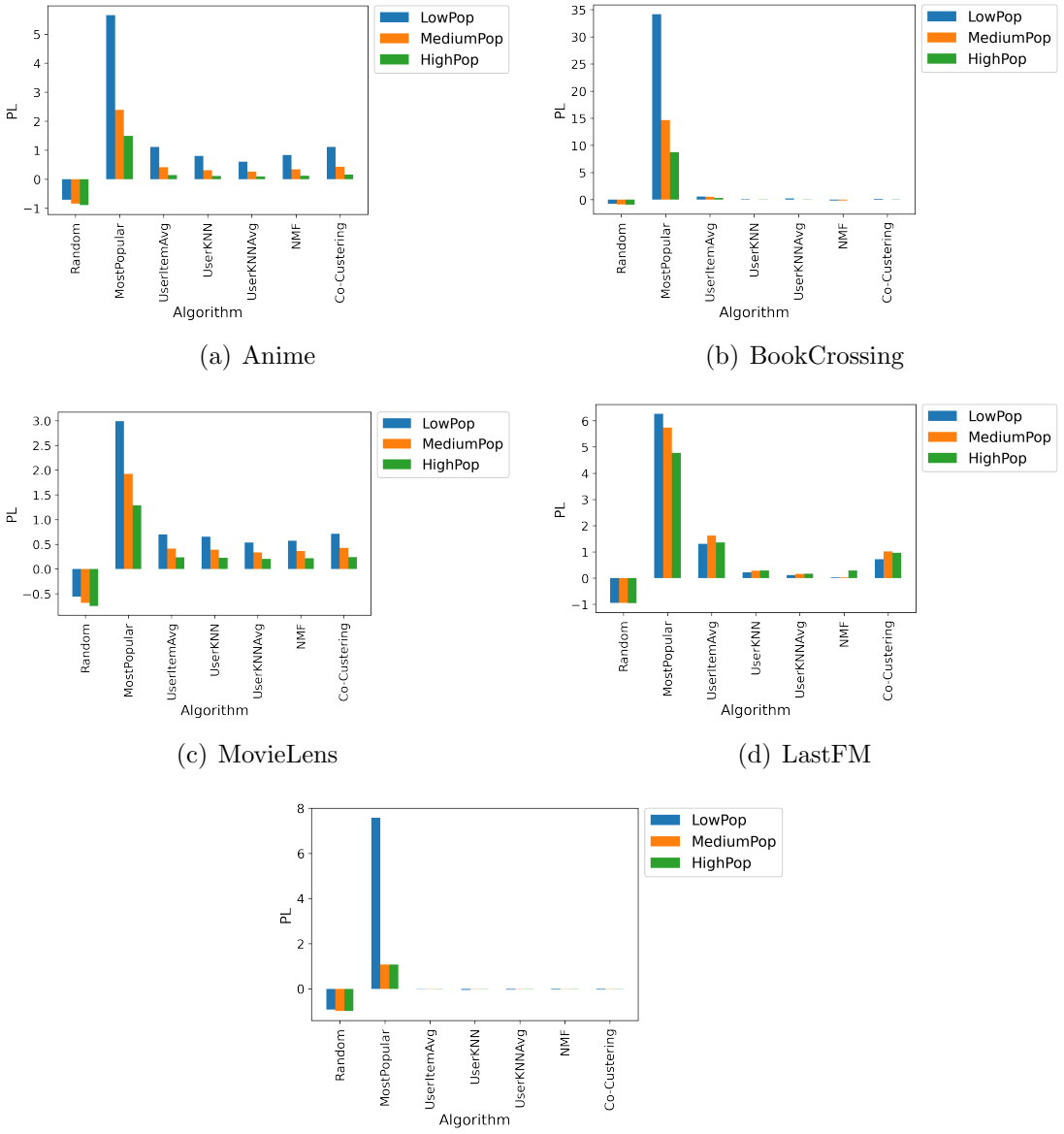
(e) Yahoo! Movies

Figure 10: Popularity Lift of each dataset, algorithm, and user group. The bars are grouped by user group and display the values for the algorithms *UserItemAvg*, *UserKNN*, *UserKNNAvg*, *NMF* and *Co-clustering*.

The biggest influence on Popularity Lift is, as already mentioned, the algorithm, but we can also see the data itself having a big impact on results. This is due to multiple factors, such as genre distribution, average profile size and system bias.

## Discussion

To answer the question, if particular user groups are affected differently on the basis of miscalibration (RQ1), we can compare the average values of each user group for each algorithm and dataset. As already observed, we can see a clear differential be-




(a) Anime

(b) BookCrossing

(c) MovieLens

(d) LastFM

(e) Yahoo! Movies

Figure 11: Popularity Lift of each dataset, algorithm, and user group. The bars are grouped by algorithms and display the values for each user group: *LowPop*, *MedPop* and *HighPop*.

tween user groups, which seems to confirm this assumption. With the exception for Random and MostPopular, we additionally see very similar values of Miscalibration over our selected algorithms, which states an important point, namely, the deployed algorithms are not calibrated in any way, therefore depending on the dataset, the

miscalibration can be quite high, *i.e.*, BookCrossing and Yahoo! Movies. From the collected measurements we can infer an almost clear distinction between user groups, as we measured the highest values of Miscalibration with the LowPop users and the lowest values with the HighPop users for all datasets, except Yahoo! Movies. This by far is the best indication for an unfair behaviour between the three metrics which were used.

Whereas the Miscalibration metric seems impartial to the utilized algorithm, the MAE on the other hand is influenced. This is due to the MAE measuring the accuracy of the ranking of items, which is done by an algorithm. To answer, if there is a connection between MAE and miscalibration (RQ2), we can look, if their values correlate with each other. Using Pearson correlation over the results for each dataset over each algorithm, except Random and MostPopular, we receive the following coefficients:

$$r_{Anime} = 0.02,\ r_{BookCrossing} = 0.668,\ r_{MovieLens} = 0.955,$$
$$r_{LastFM} = -0.351,\ r_{Yahoo!Movies} = -0.476.$$

From these values, we can infer that there are indeed cases, in which a connection between MAE and miscalibration exists, but this heavily depends on the dataset. For example, *BookCrossing* and *MovieLens* show a clear positive linear relationship, whereas *LastFM* and *Yahoo! Movies* have slightly negative linear relationships. Lastly, the *Anime* dataset seems to have no relationship between MAE and miscalibration. Overall, we can conclude, that it is not possible to find a conclusive connection between MAE and miscalibration in every dataset, negating this assumption.

Lastly, we want to infer, if the popularity lift has an influence on miscalibration (RQ3). Using Pearson correlation again, we check the linear relationship between popularity lift and miscalibration:

$$r_{Anime} = -0.141,\ r_{BookCrossing} = -0.038,\ r_{MovieLens} = -0.035,$$
$$r_{LastFM} = 0.127,\ r_{Yahoo!Movies} = -0.177.$$

Such results clearly deny this question, as values close to zero show no particular linear relationship at all. This time the results are even more clear, with all datasets sharing this behaviour, meaning, the popularity lift has no particular influence on miscalibration.

# Conclusions and Future Work

Fairness in recommender systems is a loose concept with multiple interpretations. In this thesis, we defined it as an inconsistency of metric measurements between different user groups. We split users based on the mainstreaminess of their user profile into 3 respective groups: *LowPop*, *MedPop* and *HighPop*. Additionally, we used 5 different datasets to measure the performance of several recommendation algorithms using evaluation metrics to make clear, if the metrics behave similar or have different or even opposing behaviours between datasets. As for the metrics, we used the measure for recommendation accuracy, MAE, the metric of miscalibration and a measurement of popularity bias, the popularity lift.
We observed a clear division of treatment based on the miscalibration metric between the different user groups in all datasets and for all algorithms except *Random*, which indicates an unfair behaviour in the recommendation system (RQ1). Additionally, we wanted to confirm, if a connection between MAE and miscalibration exists (RQ2), which we have to reject, as we have found no conclusive evidence which confirms this hypothesis. We found, that such a connection heavily depends on the dataset and can either be negative, positive, or none at all. Lastly, based on the previous question, we checked, if the popularity lift has an influence on miscalibration (RQ3). Unfortunately, this hypothesis has to be rejected as well, as all datasets gave clear indicators, that no such relationship exists.

These results of course have to be taken with a grain of salt, as certain limitations apply. For one, we used the standard hyperparameters, which results in an average higher MAE value than in an optimized solution. Additionally, the number of algorithms we measured were limited, as we didn't test *SVD* or other factorization-based methods except *NMF* due to computational limitations and availability. Even though the datasets received basic preprocessing, the data can be further cleaned and outlier or inherent biases could be removed, which may influence the measure-

ments. But such steps can be different on each dataset, which certainly defeats the purpose of this research, where we needed a uniform processing of the data. Finally, but also most importantly, we focused these experiments only on collaborative filtering recommenders, specifically, CF using rating predictions. Nowadays, a multitude of different systems exist, such as the ones we talked about in the related work section, and can lead to very different results which are unfortunately out of scope for this thesis.

As for the direction of future work, a comparison between different systems is clearly a good start, as this will give insight if the measured unfairness in our experiments is a unique problem of collaborative filtering, or similar problems occur in other recommender methods too. We saw (Figure 8, 9), how miscalibration is a good representation of unfairness across datasets and therefore could be used to improve recommendations using different methods, such as implementing weights or using an adversarial training method. Additionally, it is imperative to test other metrics and algorithms, which may lead to better measurements and fairer results. This ultimately leads to the development of new and the improvement of old algorithms, may they be a combination of already existing methods or completely new.

# Bibliography

[Abdollahpouri, 2019] Abdollahpouri, H. (2019). Popularity bias in ranking and recommendation. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 529–530.

[Abdollahpouri et al., 2019] Abdollahpouri, H., Mansoury, M., Burke, R., and Mobasher, B. (2019). The unfairness of popularity bias in recommendation. *arXiv preprint arXiv:1907.13286*.

[Abdollahpouri et al., 2020] Abdollahpouri, H., Mansoury, M., Burke, R., and Mobasher, B. (2020). The connection between popularity bias, calibration, and fairness in recommendation. In *Fourteenth ACM Conference on Recommender Systems*, pages 726–731.

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749.

[Aggarwal et al., 2016] Aggarwal, C. C. et al. (2016). *Recommender systems*, volume 1. Springer.

[Almazro et al., 2010] Almazro, D., Shahatah, G., Albdulkarim, L., Kherees, M., Martinez, R., and Nzoukou, W. (2010). A survey paper on recommender systems. *arXiv preprint arXiv:1006.5278*.

[Breese et al., 2013] Breese, J. S., Heckerman, D., and Kadie, C. (2013). Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*.

[Brozovsky and Petricek, 2007] Brozovsky, L. and Petricek, V. (2007). Recommender system for online dating service. *arXiv preprint cs/0703042*.

[Brynjolfsson et al., 2006] Brynjolfsson, E., Hu, Y. J., and Smith, M. D. (2006). From niches to riches: Anatomy of the long tail. *Sloan management review*, 47(4):67–71.

[Burke, 2000] Burke, R. (2000). Knowledge-based recommender systems. *Encyclopedia of library and information systems*, 69(Supplement 32):175–186.

[Burke, 2007] Burke, R. (2007). Hybrid web recommender systems. *The adaptive web*, pages 377–408.

[Chen et al., 2013] Chen, L., De Gemmis, M., Felfernig, A., Lops, P., Ricci, F., and Semeraro, G. (2013). Human decision making and recommender systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 3(3):1–7.

[Chen and Pu, 2012] Chen, L. and Pu, P. (2012). Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, 22(1):125–150.

[Chen et al., 2015] Chen, M.-H., Teng, C.-H., and Chang, P.-C. (2015). Applying artificial immune systems to collaborative filtering for movie recommendation. *Advanced Engineering Informatics*, 29(4):830–839.

[Cheng et al., 2016] Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10.

[Cramer et al., 2008] Cramer, H., Evers, V., Ramlal, S., Van Someren, M., Rutledge, L., Stash, N., Aroyo, L., and Wielinga, B. (2008). The effects of transparency on trust in and acceptance of a content-based art recommender. *User Modeling and User-adapted interaction*, 18(5):455–496.

[Ekstrand et al., 2018] Ekstrand, M. D., Tian, M., Azpiazu, I. M., Ekstrand, J. D., Anuyah, O., McNeill, D., and Pera, M. S. (2018). All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and

effectiveness. In Friedler, S. A. and Wilson, C., editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 172–186. PMLR.

[Elahi et al., 2021] Elahi, M., Kholgh, D. K., Kiarostami, M. S., Saghari, S., Rad, S. P., and Tkalčič, M. (2021). Investigating the impact of recommender systems on user-based and item-based popularity bias. *Information Processing & Management*, 58(5):102655.

[Felfernig et al., 2018] Felfernig, A., Boratto, L., Stettinger, M., and Tkalčič, M. (2018). *Group recommender systems: An introduction*. Springer.

[Felfernig and Burke, 2008] Felfernig, A. and Burke, R. (2008). Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce*, pages 1–10.

[Felfernig et al., 2007] Felfernig, A., Isak, K., Szabo, K., and Zachar, P. (2007). The vita financial services sales support environment. In *Proceedings of the national conference on artificial intelligence*, volume 22, page 1692. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Fix and Hodges, 1989] Fix, E. and Hodges, J. L. (1989). Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57(3):238–247.

[Friedman and Nissenbaum, 1996] Friedman, B. and Nissenbaum, H. (1996). Bias in computer systems. *ACM Transactions on Information Systems (TOIS)*, 14(3):330–347.

[George and Merugu, 2005] George, T. and Merugu, S. (2005). A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4 pp.–.

[Gomez-Uribe and Hunt, 2015] Gomez-Uribe, C. A. and Hunt, N. (2015). The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19.

[Harper and Konstan, 2015] Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5:19:1–19:19.

[Hendrickx et al., 2021] Hendrickx, J., Smets, A., and Ballon, P. (2021). News recommender systems and news diversity, two of a kind? a case study from a small media market. *Journalism and Media*, 2(3):515–528.

[Jain et al., 2015] Jain, S., Grover, A., Thakur, P. S., and Choudhary, S. K. (2015). Trends, problems and solutions of recommender system. In *International conference on computing, communication & automation*, pages 955–958. IEEE.

[Jameson et al., 2015] Jameson, A., Willemsen, M. C., Felfernig, A., Gemmis, M. d., Lops, P., Semeraro, G., and Chen, L. (2015). Human decision making and recommender systems. In *Recommender systems handbook*, pages 611–648. Springer.

[Jannach et al., 2015] Jannach, D., Lerche, L., Kamehkhosh, I., and Jugovac, M. (2015). What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction*, 25(5):427–491.

[Jannach et al., 2010] Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press.

[Kaminskas et al., 2013] Kaminskas, M., Ricci, F., and Schedl, M. (2013). Location-aware music recommendation using auto-tagging and hybrid matching. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 17–24.

[Kamishima et al., 2012] Kamishima, T., Akaho, S., Asoh, H., and Sakuma, J. (2012). Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer.

[Koren et al., 2009] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[Kowald et al., 2020] Kowald, D., Schedl, M., and Lex, E. (2020). The unfairness of popularity bias in music recommendation: a reproducibility study. *Advances in Information Retrieval*, 12036:35.

[Kunaver and Požrl, 2017] Kunaver, M. and Požrl, T. (2017). Diversity in recommender systems–a survey. *Knowledge-based systems*, 123:154–162.

[Lee and Hosanagar, 2014] Lee, D. and Hosanagar, K. (2014). Impact of recommender systems on sales volume and diversity.

[Lee et al., 2004] Lee, S., Yang, J., and Park, S.-Y. (2004). Discovery of hidden similarity on collaborative filtering to overcome sparsity problem. In *International Conference on Discovery Science*, pages 396–402. Springer.

[Lin et al., 2020] Lin, K., Sonboli, N., Mobasher, B., and Burke, R. (2020). Calibration in collaborative filtering recommender systems: A user-centered analysis. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, HT '20, page 197–206, New York, NY, USA. Association for Computing Machinery.

[Lops et al., 2011] Lops, P., Gemmis, M. d., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pages 73–105.

[Luo et al., 2014] Luo, X., Zhou, M., Xia, Y., and Zhu, Q. (2014). An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, 10(2):1273–1284.

[Mandl and Felfernig, 2012] Mandl, M. and Felfernig, A. (2012). Improving the performance of unit critiquing. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 176–187. Springer.

[Mansoury et al., 2020a] Mansoury, M., Abdollahpouri, H., Pechenizkiy, M., Mobasher, B., and Burke, R. (2020a). Feedback loop and bias amplification in recommender systems. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 2145–2148.

[Mansoury et al., 2020b] Mansoury, M., Abdollahpouri, H., Smith, J., Dehpanah, A., Pechenizkiy, M., and Mobasher, B. (2020b). Investigating potential factors associated with gender discrimination in collaborative recommender systems. In *The Thirty-Third International Flairs Conference*.

[Masthoff, 2011] Masthoff, J. (2011). Group recommender systems: Combining individual models. In *Recommender systems handbook*, pages 677–702. Springer.

[McCarthy et al., 2010] McCarthy, K., Salem, Y., and Smyth, B. (2010). Experience-based critiquing: Reusing critiquing experiences to improve conversational recommendation. In *International Conference on Case-Based Reasoning*, pages 480–494. Springer.

[Melville and Sindhwani, 2010] Melville, P. and Sindhwani, V. (2010). Recommender systems. *Encyclopedia of machine learning*, 1:829–838.

[Möller et al., 2018] Möller, J., Trilling, D., Helberger, N., and van Es, B. (2018). Do not blame it on the algorithm: an empirical assessment of multiple recommender systems and their impact on content diversity. *Information, Communication & Society*, 21(7):959–977.

[Müller et al., 2011] Müller, M., Goto, M., and Dixon, S. (2011). Multimodal music processing (dagstuhl seminar 11041). In *Dagstuhl Reports*, volume 1. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Musto et al., 2015] Musto, C., Semeraro, G., Lops, P., De Gemmis, M., and Lekkas, G. (2015). Personalized finance advisory through case-based recommender systems and diversification strategies. *Decision Support Systems*, 77:100–111.

[Narayanan, 2018] Narayanan, A. (2018). Translation tutorial: 21 fairness definitions and their politics. In *Proc. Conf. Fairness Accountability Transp., New York, USA*, volume 1170.

[Nematzadeh et al., 2017] Nematzadeh, A., Ciampaglia, G. L., Menczer, F., and Flammini, A. (2017). How algorithmic popularity bias hinders or promotes quality. *arXiv preprint arXiv:1707.00574*.

[Nguyen et al., 2014] Nguyen, T. T., Hui, P.-M., Harper, F. M., Terveen, L., and Konstan, J. A. (2014). Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*, pages 677–686.

[Oard et al., 1998] Oard, D. W., Kim, J., et al. (1998). Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, volume 83, pages 81–83. AAAI.

[Park et al., 2012] Park, D. H., Kim, H. K., Choi, I. Y., and Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert systems with applications*, 39(11):10059–10072.

[Reilly et al., 2004] Reilly, J., McCarthy, K., McGinty, L., and Smyth, B. (2004). Dynamic critiquing. In *European Conference on Case-Based Reasoning*, pages 763–777. Springer.

[Ricci et al., 2010] Ricci, F., Rokach, L., and Shapira, B. (2010). *Recommender Systems Handbook*, volume 1-35, pages 1–35.

[Ricci et al., 2011] Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer.

[Sarwar et al., 2000] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 285–295, New York, NY, USA. Association for Computing Machinery.

[Sarwar et al., 2002] Sarwar, B. M., Karypis, G., Konstan, J., and Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, volume 1, pages 291–324. Citeseer.

[Schafer et al., 2007] Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer.

[Schedl, 2016] Schedl, M. (2016). The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on international conference on multimedia retrieval*, pages 103–110.

[Schedl and Hauger, 2015] Schedl, M. and Hauger, D. (2015). Tailoring music recommendations to users by considering diversity, mainstreaminess, and novelty. In *Proceedings of the 38th international acm sigir conference on research and development in information retrieval*, pages 947–950.

[Singh and Joachims, 2018] Singh, A. and Joachims, T. (2018). Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228.

[Steck, 2011] Steck, H. (2011). Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 125–132.

[Steck, 2018] Steck, H. (2018). Calibrated recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 154–162, New York, NY, USA. Association for Computing Machinery.

[Thomas et al., 2020] Thomas, M., Vacek, T., Shuai, X., Liao, W., Sanchez, G., Sethia, P., Teo, D., Madan, K., and Custis, T. (2020). Quick check: A legal research recommendation system. In *NLLP@ KDD*, pages 57–60.

[Turcotte et al., 2015] Turcotte, J., York, C., Irving, J., Scholl, R. M., and Pingree, R. J. (2015). News recommendations from social media opinion leaders: Effects on media trust and information seeking. *Journal of computer-mediated communication*, 20(5):520–535.

[Wang et al., 2018] Wang, D., Liang, Y., Xu, D., Feng, X., and Guan, R. (2018). A content-based recommender system for computer science publications. *Knowledge-Based Systems*, 157:1–9.

[Wang et al., 2006] Wang, J., De Vries, A. P., and Reinders, M. J. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508.

[Watson et al., 2016] Watson, C. C. A. I. T. et al. (2016). *Recommender Systems: The Textbook*. Buku Digital.

[Yao and Huang, 2017] Yao, S. and Huang, B. (2017). Beyond parity: Fairness objectives for collaborative filtering. *arXiv preprint arXiv:1705.08804*.

[Zafar et al., 2017] Zafar, M. B., Valera, I., Rogriguez, M. G., and Gummadi, K. P. (2017). Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics*, pages 962–970. PMLR.

[Zhang et al., 2019] Zhang, S., Yao, L., Sun, A., and Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38.

[Ziegler et al., 2005] Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32.