Mitigating Popularity Bias in Music Recommender Systems

Effects on Fair Exposure, User Perception, and Motivation for Exploration

Master's Thesis in
Human-Computer-Interaction

Author: **Robin Ungruh**
Student Number: 0982717

Project Supervisor/First Examiner: Hanna Hauptmann
Daily Supervisor: Karlijn Dinnissen
Second Examiner: Anja Volk

Utrecht, October 20th, 2023

Abstract

Recommender Systems (RS) filter an immense array of options to provide us with the most suitable and relevant items. However, certain items are recommended excessively while others receive minimal exposure from the algorithms. Given the profound impact of recommender systems on our lives and decisions, essential questions arise about fairness and the equitable allocation of benefits and resources among all stakeholders. Music recommender systems play a pivotal role in the music industry, introducing the risk of unfair treatment. The ”popularity bias” characterizes the phenomenon where RS tend to disproportionately recommend popular items, thereby augmenting their exposure compared to less popular items. Given the relevance of less popular items to many users, popularity bias causes equity concerns. On one hand, the bias triggers a ”rich-get-richer” effect: already popular artists and providers further amplify their popularity, while lesser-known artists fail to receive exposure despite their equally suitable content. This has tangible consequences, including reduced financial compensation and media attention for these artists. Fairness issues for users arise when the bias is responsible for varying levels of content quality for different user groups. Niche users, for example, might find popular recommendations less satisfying, although such recommendations are well-suited for mainstream-oriented users. Therefore, their satisfaction is directly linked to the popularity bias. Various offline evaluations demonstrate the feasibility of mitigating the adverse effects of the popularity bias for both users (user fairness) and artists (item fairness). However, these evaluations fall short of assessing real users' experiences. In this work, we investigate the prospects of crafting fairer recommendations and probe users' perceptions.
We develop a recommendation algorithm (`RankALS`) and employ popularity bias mitigation techniques. One aims at artist fairness (`FA*IR`), while the second focuses on user fairness (Calibrated Popularity). Through an algorithmic evaluation of the algorithms on the state-of-the-art music dataset (LFM-2b), we observe that the user-centric algorithm performs comparably to the base algorithm based on performance metrics (e.g., accuracy, NDCG) and recommends songs aligned with users' historical listening preferences in terms of popularity. This highlights its high user fairness. However, it retains an over-representation of popular items. Conversely, the artist-focused algorithm increases exposure for underrepresented songs, achieving item fairness, albeit at the expense of matching users' popularity-based listening histories and performances.
Nevertheless, concentrating on suggesting less popular items can yield additional value for users. Particularly in exploration scenarios, lesser-known, unfamiliar songs can facilitate users' discovery of new content, improving their experience. To further investigate this phenomenon, we conducted a user study. Leveraging users' Spotify profiles, we generated personalized recommendations and apply mitigation algorithms. By means of questionnaires, we assess users' perceptions and satisfaction, aiming to conclude how mitigation impacts user experiences and influences future listening behaviours.
The results show no significant differences in satisfaction between the algorithms. While user-centred fairness goals seem to not influence the users' perception, reduced popularity achieved by an item-fairness-focused algorithm was perceived by the users, and the accompanying reduced *Familiarity* with the recommendations can enable *Discovery*, the feeling that the recommendations enrich the user's musical taste. *Discovery* is highly positively associated with satisfaction metrics and behavioural intentions. While no direct impact of the mitigation methods on behavioural intentions could be made, we show that high satisfaction predicts behavioural intentions.
This research advocates for mitigating biases in music recommender systems for the benefit of item providers (artists) and users, avoiding unfair treatment of distinct user groups. Moreover, we demonstrate the effectiveness of mitigation algorithms at an unprecedented scale and investigate real users' perceptions of the outcomes. Lastly, we explore users' behavioural motivations toward engaging with more equitable content. We posit that generating fairer recommendations can achieve a lasting influence on users' consumption behaviours, promoting an overall healthier music consumption pattern.

Table of Contents

Chapter 1

Introduction

Recommender systems (RS) have a profound impact on our decision-making process by helping us navigate through a vast array of options. With so many choices available, RS serve as a valuable tool to filter through them and present us with the most suitable and relevant items based on our specific needs, whether they are explicitly stated or inferred from implicit feedback [18, 24, 31]. The main task of a recommendation system is to the (implicit or explicit) rating of each item for each individual user. The ratings will be used to present the user with the best possible options. However, this task can be challenging due to the potentially large number of users and items, combined with the limited availability of existing ratings.
The impact of recommendations on our lives and decision-making processes raises essential questions regarding fairness and equitable allocation of benefits and resources for all stakeholders affected by these recommendation models. It has been widely acknowledged that social biases are inherent in the data and algorithms used by recommender systems, which in turn are adopted by many users. These biases and inequalities become embedded in the systems, exposing users to biased recommendations that can reinforce existing inequalities and lead to unfair outcomes [24, 31]. In recent years, fairness in recommender systems has garnered significant attention. There is a growing recognition that these systems can inadvertently perpetuate biases, resulting in unfair outcomes for both users and providers [24].
Music recommender systems (MRS) express additional challenges for users since the number of ratings is mostly very sparse and feedback is barely given explicitly, but has to be inferred from interactions with the system [18]. It exists enough music on streaming platforms to satisfy every user, but it is challenging to provide each user with the best suitable items [19]. The need to examine fairness in Music Recommender Systems (MRS) has become increasingly important due to the rising popularity of music consumption through streaming platforms. In fact, streaming platforms contribute a significant 67% to the global recorded music revenue [51]. Unfair outcomes in MRS can impact various stakeholders, including the users, the providers (e.g., artists and labels), and the platforms [9, 26, 31].
Overall, a lack of diversity in recommendations can lead to various problems. Music recommender systems hold the potential to prominently recommend music that is already known by the user or is very similar, therefore reinforcing existing behaviour [33]. This can result in effects like the filter bubble effect [35]. Additionally, algorithms tend to recommend highly popular music more than those of niche artists. This phenomenon is called popularity bias [19, 45, 120]. Current systems recommend more songs of higher popularity in comparison to the organic listening behaviour of the users [6]. Therefore, consumption is shifted towards highly popular items.
One could argue that popularity bias is not inherently problematic in music recommendations. It is a common practice for recommenders to prominently display popular items, and this approach has proven to be effective. Popularity-based recommendations often outperform classical personalised recommendation approaches in terms of accuracy [16]. Nevertheless, many researchers argue that while it might be a sufficient strategy, it might be in contrast with other goals of the recommender [52]. For instance, an RS could also aim at increasing the diversity of the recommendations or at enabling the discovery of novel items. Regarding the goal of fairness, it is argued that systems should not "inordinately favour popular, well-known and possibly well-funded content creators" [31].
Popularity bias can create unfair outcomes for users and artists [39], and many researchers propose that

recommenders should include less popular items since it can lead to strengthened perceived quality and usefulness [18]. Exploring less-known artists can offer numerous advantages, benefiting both the artists themselves and users seeking more diverse content. By deviating from popularity-based recommendations, users have the opportunity to discover unknown or emerging artists. While popularity-based recommendations may be successful in providing items that the user might like, it is important to consider the potential drawbacks. Relying on popularity can lead to unfair recommendations, perpetuating the visibility and success of already popular artists while hindering the visibility of underrepresented or marginalized artists. We argue that boosting new or niche items should be an additional goal of recommenders next to finding fitting items [52]. Creating fairer recommendation lists has the potential to influence and shift the consumption of music in general. This can help to maintain a healthy consumption of music [45].
In conclusion, popularity bias affects artists unfairly by putting less popular artists at a disadvantage solely because they are less known. Additionally, users face negative outcomes when some users receive worse recommendations than others because of popularity bias. This could, for instance, occur when one user has mainstream taste in music while another user prefers niche music and genres. The mainstream user will presumably be satisfied with popular, mainstream recommendations. For the niche user, on the other hand, popular recommendations might be much less suitable and reduce the user's satisfaction. When popularity bias affects users' satisfaction differently solely because of their taste in music, unfairness occurs [5, 7, 64].

Although it has been shown that users are satisfied with recommendations of less popular artists or those that depart from their original taste [45], users currently do not shift to a more diverse listening behaviour by listening to recommendations, but through more organic consumption of music [10], for example, by listening to playlists or exploring albums and songs without relying on musical recommendations. This indicates that current recommender systems do not contribute to achieving fairer listening behaviour of the users, which leads to the question of how recommenders might be able to support more diverse artists. It remains questionable whether recommendations that aim at mitigating the popularity bias can actually perform a change in the user's listening behaviour, and therefore, have a long-lasting impact on music consumption in general. Algorithmic analyses have shown that mitigation of the popularity bias is possible (cf. [7, 57]), but its effects on the users remain unclear. There is a limited number of studies that study the effect of popularity in music recommendations on the user.
Nudging users towards specific choices in the short term has been researched repeatedly, and accomplished successfully [53]. Nevertheless, this does not improve user satisfaction because users are directed towards exploring content away from their current preferences, resulting in less personalisation [73].
While research supports the presumption that fairer recommendations can shift the users listening behaviour [45], it remains unclear whether this has a long-lasting effect on the users. This shows the need for a study that investigates the effects of fair recommendations regarding mitigated popularity bias on the users' attitudes. In this work, we investigate the influence of a recommender system that mitigates the effect of the popularity bias on the satisfaction and motivation for future exploration.
Current research on RS mainly uses offline evaluations for predicting RS performance. The effects of RS on the user cannot be solely assessed by this. Current evaluations lack of analysing the effects of recommendations on real users [52, 65]. Real studies can create insights into the perception of the items. There are only a few user studies that investigate the popularity bias in MRS (e.g., [38, 39, 43, 68]). Those test the perception of different degrees of popularity in recommendations. To our knowledge, there are no user studies that investigate the influence of mitigation strategies on users. Therefore, we propose a two-step approach for evaluating mitigation strategies:

- Firstly, we create a base recommendation algorithm and apply popularity bias mitigation strategies to it. Those will be trained and evaluated on a state-of-the-art music-related dataset [99].

- Furthermore, we leverage these recommendation and mitigation techniques to generate personalised recommendations for participants in a user study. The primary objective of this study is to examine the impact of mitigation strategies on actual users. It is crucial to acknowledge that different situational factors can influence individuals' perception of popularity bias in Music Recommender Systems (MRS).
  To address this, we hypothesize that employing a clear "lean-in" exploration setting [100] would be the most effective approach. During exploration sessions, users actively listen to songs intending to find songs to further listen to in the future or to add to their playlists. In such a setting, less popular items may be positively received, as users engaged in music exploration may value and appreciate unfamiliar and novel recommendations. Therefore, we will explicitly define and communicate this lean-in exploration setting to the participants in our user study, ensuring a clear understanding of the context and promoting an open mindset towards exploring less popular items.

Those steps aim at answering the research questions:

RQ 1: To what extent can mitigation strategies reduce the popularity bias effect in music recommendations, promoting fairer exposure of underrepresented artists?

- **RQ 1.1:** Are the mitigation strategies "Personalised Long-Tail Promotion" (`XQ`), `FA*IR` and "Calibrated Popularity" (`CP`) able to create a similar performance (NDCG, Accuracy, Precision) as the base algorithm (`RankALS`)?
- **RQ 1.2:** To what extent can the mitigation strategies (`XQ`. `FA*IR` and `CP`) reduce popularity bias in terms of item-centred metrics, specifically in promoting equal exposure and long-tail exposure of underrepresented songs, compared to the base algorithm (`RankALS`)?
- **RQ 1.3:** To what extent can the mitigation strategies (`XQ`, `FA*IR` and `CP`) reduce popularity bias in terms of user-centred metrics, specifically popularity lift and user popularity deviation, compared to the base algorithm (`RankALS`)?
- **RQ 1.4:** How do the mitigation strategies (`XQ` and `CP`) compare in terms of user-centred and item-centred metrics, specifically considering their goals of improving item fairness (`XQ` and `FA*IR`) and user fairness (`CP`)?

RQ 2: To what degree does a mitigated popularity bias, created by different recommendation strategies, have an impact on the user perception and satisfaction with music recommendations in an exploration setting?

- **RQ 2.1:** How does the manipulation of recommendation strategy (base algorithm, item-centred mitigation algorithm, user-centred mitigation algorithm) impact user perception of popularity and popularity lift in the recommended music, and how does this influence their satisfaction with the recommendations and choices?
- **RQ 2.2:** To what extent do personal characteristics and previous listening behaviour interact with the recommendation strategy in shaping user perception and satisfaction with the music recommendations?
- **RQ 2.3:** How do mediators, such as perceived popularity, perceived fairness, and perceived familiarity, contribute to the user perception and satisfaction with the music recommendation lists generated by the different recommendation strategies, and how do these factors interact with the recommendation strategy?

RQ3: Does a mitigated popularity bias in an exploration setting lead to increased motivation for exploring long-tail music items and indicate potential changes in user behaviour towards fairer music consumption?

- **RQ 3.1:** To what extent does the mitigated popularity bias in an exploration setting influence users' planned behaviour, as indicated by their openness to receive similar recommendations, intentions to use the recommender system again, and intentions to listen to the recommended songs in the future?
- **RQ 3.2:** To what extent do personal characteristics and previous listening behaviour interact with the users' planned behaviour, as indicated by their openness to receive similar recommendations, intentions to use the recommender system again, and intentions to listen to the recommended songs in the future?
- **RQ 3.3:** Can the presence of mitigated popularity bias in the exploration setting indicate potential changes in user behaviour towards fairer music consumption, based on users' responses regarding their planned behaviour and intentions for future music consumption?

Chapter 2

Related Work

2.1 Literature Review Plan

To review the literature regarding the research question, we follow a snowballing literature review process. We use search engines (mainly Google Scholar[1]) to identify literature. While focusing on ”Music recommender systems” as a key term, exploring the literature on ”recommender systems” in general is also essential. Research in the broader field of recommender systems offers valuable insights and transferable knowledge for developing music recommender systems. Many underlying techniques, algorithms, and principles used in general recommender systems can be adapted to address the specific needs and challenges of music recommendation. Additional terms like ”Fairness”, ”Popularity bias”, as well as ”Nudging” offer valuable insights to answer the research questions. Complementarily, the supervisors of this thesis provided important literature that initialized the literature research.
Based on this initial process, we scan the abstracts of the papers and identified key papers that are central to our research topic and provide a foundation for our understanding of the subject matter.
Following this, we scan the reference lists of the key papers to identify additional articles, books, or other sources (Backwards snowballing). Making use of the ”cited by” feature from Google Scholar, we identify papers that referenced the key papers (Forwards snowballing).
The abstracts of the identified papers were reviewed and relevant papers were selected to identify new key papers. The previous steps are repeated multiple times to develop a thorough understanding of the research area.
The selected papers are read carefully, and we take notes regarding their background, methodology, and results. Additionally, we order them into different categories, like ”Fairness”, ”Popularity bias”, ”Fairness measurements”, and ”Methods”. This offers a clear initial structure for the ”Related Work” chapter. We identify relations to other literature and create additional tags and subcategories. For example, we tagged whether they were related to Information Systems in general, recommender systems, or music recommender systems specifically.

[1]https://scholar.google.de/

2.2 Recommender Systems

Recommender systems are information access systems. Given a set of items, they algorithmically provide the user with the items that satisfy their information need [31, 94]. Other systems that accomplish this task are information retrieval systems or information filtering algorithms. While approaches like filtering use explicit expressions and queries, recommender systems often present the information based on implicit data about the user.
Typical usages of recommender systems include the finding of good items, finding all good items, recommending a sequence of fitting items, browsing through items as entertainment, testing the recommender's credibility for possible future interactions, and more [49]. Providers of recommender systems use those to increase the number of items sold, to sell more diverse items, or increase user satisfaction and fidelity. In general, they aim at understanding the user needs better to create fitting recommendations for their users [94].

2.2.1 Properties

Ekstrand et al. [31] define different steps the system has to accomplish to fulfil its task. Each of those steps has its unique challenges and variations.

- **Understanding the items:** The items in the set or corpus have to be understood and represented properly to connect them to the users' needs.
- **Understanding the user and their needs:** To retrieve fitting items, the system must understand the user and their information needs and represents them accordingly.
- **Retrieval:** Based on the user needs and item representation, the system can retrieve matching items.
- **Presentation:** The retrieved items need to be rendered and presented appropriately.
- **Reflection:** Based on the users' behaviour toward the presented items, the system can understand the user better and update their representation to inform future retrieval and evaluate its performance.

In the following, we will explain those steps and their properties in more detail.

2.2.1.1 Item data

Items in a set can have various sources and different unique properties. The set and its items can be static or dynamic and change over time. Items can be added, changed or removed from the repository.
The items within those repositories can be represented in different ways. Typically, the representation can be divided into three categories: *Content data* is static and refers to the properties that were generated during the creation of the item. *Metadata* expresses more information about the environment, in which an item was created, including its creator, the time of creation, the genre etc. *Usage data* is highly dynamic and refers to historic information on the interaction of users with the items [31].
Algorithmically, this representation is often accomplished by representing the properties in a vector space model. Each item is represented in a high-dimensional space, where each dimension represents a property of the item [76].
Item data is often unstructured (e.g., text) and needs pre-processing to represent it properly in a vector space to facilitate extraction of relevant properties [76].

2.2.1.2 User data

The users who interact with the system have specific information needs. Those might be stated explicitly (e.g., the user wants to listen to songs from an artist), or implicitly (e.g., the user wants to discover music that they will like). Implicit information about the user can be demographic information, but also their previous interactions and information inferred about their preferences. When the user interacts with a system, their interactions with items can be used to infer information about their preferences. By gathering various types of data from the user, a user representation is defined [31, 76].

2.2.1.3 Retrieval

Retrieval is the main task of any recommender. The system must exploit the profiles and preferences of the user to retrieve matching items [18]. Combining the users and items to retrieve the correct items for the user can be done in various ways. Firstly, the user needs can be transformed into queries that can be compared to the item representation. Fitting items can be retrieved based on this information. Another approach to gathering information about the users' attitudes can be done in a matrix format. A rating matrix, for example, represents how each user rated each item previously. This can be retrieved through explicit feedback when the user rates items directly, or through implicit feedback by tracking the users' interactions with items. Those matrices are typically very sparse and incomplete since most users will not have interacted with and rated each item. Nonetheless, the matrix can be used to predict their rating for other items algorithmically [31, 76]. Explicit feedback can be provided on different scales (discrete scale, binary value), but also through more qualitative feedback (e.g., comments). Implicit feedback can be measured in various ways. All kinds of interactions with an item can be monitored and analysed. Nevertheless, implicit feedback cannot easily represent negative feedback [18], and it can easily be misinterpreted [76]. For example, just because a user clicks on an item, it is not clear that they like it or that they will also like it at another point in time.
There are different approaches how to generate recommendations for a user. The 5 most common methods to predict recommendations are [18, 94]:

- **Demographic Filtering** exploits stereotyping by identifying groups of user profiles that tend to like specific items and recommending those items to other users in this group [96].
- **Collaborative Filtering** does not consider the properties of items, but this method only identifies relationships between users and items. The classical approach includes the creation of an item-user matrix where each cell represents the user's (explicit or implicit) rating for the item. By identifying similar items or similar users, the system can predict the user's ratings for unrated items and recommend those with a high rating to the user [62].
- **Content-Based Filtering** makes use of the properties of each item and identifies similar items to those that the user liked before. It does not make use of other users but relies solely on similarities between items [76].
- **Context-Based Filtering** uses contextual information to describe items. Contextual information does not correspond to the content of an item but uses information from the provider and the environment of the creation [8, 18, 58].
- **Hybrid Methods** use multiple approaches combined to mitigate the limitations of each approach.

2.2.1.4 Presentation

The retrieved items can be presented in various ways to the users. The type of presentation is dependent on the interface. In the simplest case, the system presents the item with the highest predicted match to the user's needs. This is suitable for small and limited interfaces. Many popular approaches offer the freedom of selection between highly ranked items.
Ranking approaches are typical and natural for recommender systems since the retrieval often operates by predicting the match of all items to the users' needs and ranking them in decreasing order [31]. Nevertheless, this approach can lead to lost information on relations between those objects. Kagie, van Wezel and Groenen propose a 2-dimensional map approach to visualise the similarity between objects [56].

2.2.1.5 Reflection

The interaction of the user with the interface and their explicit or implicit ratings inform the system about the user's preferences and choices. This can be measured as the utility of each item during user interaction. The utility can be compared to the estimation of the match before presenting the results. This can be used to gather more information about the user and to gain insights into important properties of the items that might have not been considered in the recommendation process previously [31].
There are different approaches for evaluating the performance of a recommender system ranging from offline approaches to user studies. In section 3, we create a recommender system and evaluate it algorithmically. Different evaluation methods are discussed. In section 5, this recommender system is evaluated in a user study.

2.2.2 Challenges

Celma [18] defines the recommendation problem, which has two challenges. Firstly, estimating the likelihood of each item matching the user's needs, and secondly, how to present the relevant items to the user so that they are able to select the best fitting item. There are various factors influencing the effectiveness of a recommendation and each approach has its unique advantages and disadvantages.
Generally, a recommender system should be evaluated considering multiple goals that influence the users' perception. For example, performance, meaning the ability to provide users with the most fitting items over sub-optimal items, is the most obvious goal. But other goals like the novelty of recommendations can additionally add value to a recommender system [17]. Additionally, the system can be evaluated whether it covers all users and items, or whether it is trustful [18].
Recommendations are more effective when a part of them are novel and not the obvious choice and it is transparent why those items were recommended (explainability). Nevertheless, the top items regarding accuracy are not always the best. A certain degree of diversity between the recommended items can improve the chance of recommending a suitable item. For providing proper recommendations, the system must consider changes in the behaviour of the users. Preferences and interests of a user change. Additionally, also the age of an item might influence the quality of the recommendation. Newer items might be preferable over older items in some cases. Therefore, choosing the best options does not always lead to the best item for the user.
An additional challenge is data sparsity and high dimensionality of the data. As explained in section 2.2.1.3, matrices are very sparse; the users won't have interacted with or rated most of the available items. This can cause a challenge in extracting optimal recommendations. Data sparsity also influences the cold start problem. New items are difficult to recommend because there are few ratings for those

items. The cold start problem can also be defined for new users entering the system. Those users have provided no or only a few ratings. Assessing their needs and providing them with proper items is more challenging [18, 94]. There are many more challenges that designers of RSs have to consider. For an overview, see [94].

Some approaches face certain problems more severely than others. For example, recommendations created by content-based filtering tend to suffer from a lack of novelty and can lead to over-specialization [76] since the items are typically very similar. On the other hand, collaborative filtering is challenged by the cold-start problem because of a lack of ratings.

Finally, current research has an increasing interest in creating fair recommendations. Since information access systems take part in critical decision-making and affect many people, the concern of whether benefits and resources are fairly allocated arises. Recommender systems influence multiple stakeholders and can affect all of them unfairly [31]. More details about fairness and how RS can treat users and other stakeholders unfairly are discussed in section 2.4.

2.3 Music Recommender Systems

MRS systems play an increasing role in how music is consumed. 67% of the global recorded music revenue in 2022 was created by streaming services. Streaming platforms like Spotify[2], Amazon Music[3] or Apple Music[4] provide unlimited music to their users. This amount of music can be overwhelming; therefore, those platforms make use of recommender systems to limit the number of choices by creating fitting recommendations to their users' preferences [100]. Those recommendations can be albums, artists, tracks, or playlists [100]. The platform Last.fm[5] allows users to store their listening records and provides a recommender system. Additionally, it provides public datasets to facilitate research in music recommendation [98, 99] which is prominently used in research of music recommender systems.
Music streaming services tailor the recommendations and playlists to the user's taste [31] and offer the possibility to explore new music. MRS pose unique challenges and properties in comparison to many other domains [100, 101]. While online shopping systems aim to provide users with items that align with their specific needs, with success measured by completed purchases, music consumption differs in nature. In music consumption, individual tracks are typically consumed for a few minutes. Hence, unfitting recommendations do not have an overwhelmingly negative impact on the user's experience in comparison to, for example, a purchase that doesn't fit the user's needs. However, it is important to consider that multiple items can be consumed in a sequence. Tracks are frequently listened to repeatedly and in conjunction with other tracks, creating a unique listening experience. Moreover, music consumption often occurs in a more passive manner compared to other domains like movies, where consumption tends to be more active and engaged.
Given these distinctions, the challenges of providing recommendations in the music domain that fit the users' needs and preferences are different from those encountered in other contexts, like online shopping. The immense catalogue size, the sequential consumption patterns, and the passive nature of music consumption all contribute to the complexity of creating effective music recommender systems.
Schedl et al. [100] describe three different types of music recommendations. Firstly, **Basic Music Recommendation** refers to assistance in browsing the catalogue of items. The system aims at recommending items (artists, albums, tracks) that match the general consumption of the user in the form of lists, or by providing personalised lists that serve as playlists tailored to the user's taste. **Lean-in Exploration** emphasizes an active and engaging consumption of the recommended music. In those scenarios, the user listens actively to the recommended music. The user might use those to explore new music or create new playlists. This exploration can be improved by recommending fitting items to the user. Finally, **Lean-Back Listening** refers to a contrasting form of music consumption, where the user does not directly interact with the UI, but listens to the music passively, and receives recommendations in sequence without seeing the complete list. All of those approaches have their unique properties and challenges. For example, it might be easier to gain feedback during Lean-in Exploration (e.g., saving songs in own playlist), while gathering feedback might be more difficult during Lean-Back Listening sessions, where the user barely interacts with the system. On the other hand, Lean-in Exploration might require retrieving many novel items, while known songs can be suitable for Lean-Back Listening.
There is an immense amount of music available to satisfy each user but presenting the right items to the right users, which have unique tastes, is not trivial [19]. This is mainly accomplished by presenting the user with a sequence of songs. Cunningham, Brainbridge and Falconer [23] divide music constructs into playlists and mixes. They define that playlists for personal use are mainly used as background for other activities, often with a specific occasion, mood or emotional state in mind. They are repeatedly listened

[2]https://open.spotify.com/
[3]https://www.amazon.com/music
[4]https://www.apple.com/apple-music/
[5]https://www.last.fm/

to on similar occasions and mainly listened to in shuffle mode. This is typically fitting since the songs are mostly compatible and won't lead to disruptive mood switches. On the other hand, formal mixes are defined by a central theme and an organizing principle. The order of the songs is important since they are played at certain events or activities as background music (e.g., a party or gathering) or to tell a story. Preferences for an order might change between users, while some users want songs that fit each other, some prefer switches between genres, tempo and mood.

2.3.1 Properties

Music recommendation follows the same steps as a classical recommendation model (see section 2.2.1). Nevertheless, each step of the recommendation process is unique and their challenges have to be considered and faced. We will discuss the properties of music recommendation in this section and highlight their challenges.

2.3.1.1 Item data

Firstly, musical items possess diverse features and dimensions that can be leveraged in recommendation algorithms. These features can be represented at various levels of complexity and abstraction. The primary elements to be considered in music recommendations are artists and songs. In addition, it is also common to consider whole albums or playlists as distinct items in the recommendation process.
Input media types for abstractions of musical data are audio features, texts, and images (and videos). In a low-level abstraction, they describe the physical features of audio, text or image data. Regarding audio data of a song, this can be the frequency at every time point, or the duration and the timbre of sound events. Those features are called **signal features**. On a mid-level abstraction, the data can be described as concepts and relations derived from the physical data (e.g., rhythm, harmony, genre). Those features are called **content objects**. The highest level of abstraction describes **human knowledge**, which is derived by human reasoning about the content objects. Those features can explain emotions, expectations, or similarities between songs. An overview of the types of metadata can be found in [18] and [85].
Each song has much data that can be analysed, and different levels of abstraction can be used for generating fitting recommendations.

2.3.1.2 User data

Users can be described in various ways: they can be classified based on their degree of interest in music, their demographic data and their preferences [18]. Analysing user data can give recommender systems various insights into what items are suitable for which users.
Nevertheless, estimating preferences is a challenging task since preferences change over time and users differ in their preferences to explore new types of music. Typical approaches of retrieving similar items based on extensively heard music seem to not be sufficient since user preferences are not static but evolve [88]. Phases in which users explore new music and genres heavily occur repeatedly but not continuously. There are weeks when users explore less music followed by weeks when much new music is explored [80]. Over time, users generally shift away from previously heavily streamed genres; therefore, short-term preferences do not necessarily match long-term preferences [73]. Accounting for those shifts can improve recommendations [88], but personal differences influence those shifts immensely. For example, listeners from different age groups consume and explore music differently. When analysing how users explore music,

it was found that younger listeners exploit known content more and explore less new music in comparison to older listeners. In contrast, younger listeners listen to more diverse content [80].
A similar trend can be found when investigating the influence of musical expertise on listening behaviour. Users that have higher expertise listen to more diverse content (artists and genres), but their preferences are more static and consistent. They might be less open to exploring new content that differs from their current tastes. Overall, users tend to explore genres that are already close to their current preferences [73].
Nevertheless, musical preferences can also change based on the situation. Users tend to prefer different music in different contexts, and recommender systems can be more personalised when they consider different contexts and adapt their recommendations to those [110]. Different contexts can include the physical state (e.g., location), as well as the time (e.g., season, daytime) and the emotional and personal state of the user [8]. For example, the music played at a party might differ from the music played while studying. Therefore, the recommendations in those different settings should also match the situation. Additionally, the goal of the user can change and affect the type of music recommendation (general, lean-in, or lean-back) that is needed.
Accounting for individual differences and changes in preferences can influence the effectiveness of recommendations heavily. The goal of the user has to be investigated and situational factors have to be considered to create fitting music recommendations.

The presentation of the user profile does not differ much in comparison to standard recommender systems. Since users barely give explicit feedback to music recommendations, the system often has to rely on implicit feedback and the user's listening history. The user's listening history consisting of all songs they listened to will be addressed as **user profile**.

2.3.1.3 Retrieval

The classical approaches presented in section 2.2.1.3 can also be used for MRSs. Nevertheless, they reflect special challenges.
For instance, collaborative filtering approaches rely on feedback from the user towards items and recommendations. Since there is barely any explicit feedback on typical music streaming platforms, those have to rely heavily on implicit feedback gathered from users' listening habits. A straightforward way of gathering feedback is analysing the number of plays of a song [18]. Additionally, interactions like skips can be used to identify the effectiveness of a recommendation [10, 45]. Implicit feedback poses some difficulties. While it can indicate user preferences, it does not necessarily reflect direct preferences towards the music. It can depend on various factors like the user's activity, the context, mood or engagement [100]. Although it is not the optimal indication for creating recommendations, it is often the only available feedback from the user. The number of plays of a song can be expanded to analyse the user's rating for an artist to identify which artists should be recommended based on the frequency of plays [18].
Collaborative filtering approaches are the most common form of an algorithm for music recommendation [100], but they have to be handled carefully since they are very prone to various types of biases.
Context-based filtering makes use of additional information, like web-mining techniques, collaborative tags and crow annotations and information. This data can be used to calculate the similarity between liked songs and new items. There are different approaches to creating context-based similarity. They can be based on text retrieval, co-occurrence (e.g., in playlists), or based on listening habits from different users [58].
Content-based filtering approaches do not rely on data from other users, but they rank items based on their similarity to each other. The similarity computation can be accomplished on each level of abstraction of the songs [18]. Nevertheless, many recommender systems suffer from presenting recommendations

that are close to previously liked items, and therefore, reinforce current behaviour [33]. Content-based approaches tend to show more of the same and do not facilitate the exploration of novel items [88].
Overall, many of those approaches have disadvantages and advantages in the music field. Those can be mitigated by using hybrid methods, including deep learning approaches, which use all types of data for creating recommendations.

2.3.1.4 Presentation

There are different ways of representing recommendations to a user. Following Cunningham's approach [23], those can often be classified into playlists and mixes. A recommender system can create a whole playlist for the user, expand playlists or play music following up on music previously listened to. Additionally, they have to be tailored to the user's setting and goal (general, lean-in, or lean-back) [100].

2.3.1.5 Reflection

MRS can use different options to reflect on the effect of their recommendations. Firstly, a recommendation seems successful when users interact positively with an item, for instance by saving the song or playing it repeatedly again. A recommendation can also be classified as unsuccessful when the song was skipped or removed from a playlist [45]. Nevertheless, explicit ratings of items are not common for music recommendations which makes their evaluation difficult.
Streaming platforms make use of different measurements to classify a recommendation algorithm as successful or not. Those include direct measures like song streams and song skips or by analysing whether users consider a subscription after being exposed to a certain algorithm [10].

2.3.2 Challenges of Music Recommendation

A significant hurdle faced by numerous music recommendation methods is the high level of data sparsity which occurs due to the high number of available items. This refers to the fact that the majority of entries in the user-item matrix are either empty or missing, hence no (implicit or explicit) rating for them exists. To illustrate, the LFM-2b dataset [99] demonstrates an extreme level of sparsity of 99.81%. On average, the users have listened to 0.19% of the available songs in the dataset. To illustrate the severity in music recommenders, we observe a lower sparsity of 95.53% in the movie-related MovieLens-1M dataset [67].
Users have to provide ratings for collaborative filtering approaches, and contextual data has to be generated by users to use context-based approaches. Data sparsity is especially severe for artists from the "long tail". There is much more data for highly popular artists, which offers more options for many similarity approaches that are context-based. Additionally, collaborative filtering approaches perform worse for those songs because there are fewer ratings for less popular songs [58].
Since music recommendation typically involves sequential consumption, the RS typically should be sequence-aware to create a natural sequence of songs [100]. [23] argue that sequence is less important for playlists, but mood or tempo changes can still feel disruptive.
Some additional decisions and trade-offs have to be made when designing a recommender system. For example, whether novel items or familiar songs should be preferred, or to what degree songs should be similar to the user's taste or diverse and broaden the user's musical knowledge. Music recommender systems as well as many other recommender systems in different areas tend to create less diverse recommendations as the system is used; therefore, there is a "natural" narrowing effect. When recommenders, for instance by using content-based recommenders, recommend highly similar items [88], this effect can

be increased, creating a filter bubble effect [35, 82].

While the main goal of many recommenders and the evaluation of recommenders is focused on creating accurate recommendations for a user, there might be many other factors that should be considered when evaluating a recommender system. The question arises of whether a system respects the users' values, and interests [33] and how effects like the filter bubble effect can raise issues of unfair presentation of items [52]. Various societal and algorithmic biases affect the fairness of the recommendations. What kind of recommendations users receive has a direct influence impact on users and providers. If a system does not reflect each user's preferences equally well, the system might have systematic errors towards certain user groups. Recommender systems need to consider whether each user receives a fair quality of service, but they also need to consider whether all artists and music providers receive fair distribution of their music, or whether some artist groups are systematically over or under-represented. Those effects have a direct influence on the artist [31]. By being aware of unfair treatment, designers of those systems can directly influence and shift the consumption of content. Consequently, they can contribute to creating a healthier and fairer consumption [45]. In the following, we will analyse how recommender systems can create unfair recommendations, and how those recommendations can influence stakeholders that are directly or indirectly influenced by the systems.

2.4 Fairness

We have shown that simply creating a recommender system that only has the goal in mind to recommend the items that have the highest chance to be liked is not sufficient for evaluating its success. There are many other factors like diversity, novelty or serendipity of which the influence can and should be evaluated as well [30]. Additionally, research focuses increasingly on how fair the recommendations an RS provides are. Are the recommendations fairly distributed across all users? Are equally fitting items recommended equally? Since technology and recommendations become increasingly important factors in our decision-making, they can immensely impact the actors affected by those decisions. Therefore, the question of whether all benefits and resources are equally allocated has to be considered and evaluated [31]. Answering this question is complex, and we have to understand what unfairness means and how it can harm different actors. Additionally, we have to clarify how unfairness differs from other terms like biases. Therefore, we will explain those terms in the following.

2.4.1 Bias

Bias refers to deviance in the system or data but does not imply a normative judgment. It describes a statistical concept instead of the implications and effects on a group of people [31]. Mitchell et al. [79] divide biases into statistical biases and societal biases. **Statistical biases** are described as systematic mismatches between the respective data and the real world or target group that the data should represent. Those can occur by not sampling the observed target group properly, or by having systematic measurement errors. **Societal biases** on the other hand refer to differences in the data that are not created by sampling or measurement errors but are influenced by real-life inequalities. For example, data could represent worse outcomes for some groups of people who are already treated unequally by society. While this might represent real-world outcomes correctly, these biases derive from societal inequalities and can explain discrepancies between the real world and the arguably ideal world. Those biases can influence measurements and lead to statistical biases.
Biases, in general, might not necessarily be unfair but explain a discrepancy within the given data. Ekstrand et al. [31] conclude that biases can be unfair, or they can lead to unfairness. Nevertheless, biases also give insights into the structure of data and can be used to address fairness problems by actively creating biases to make up for other biases.

2.4.2 Fairness

Explaining the concept of fairness poses a greater level of difficulty. Generally, fairness describes a normative judgement of the treatment of individuals or groups based on an ethical, moral or legal standard. There are various definitions of fairness and how to measure those [9]. Different stages exist in an algorithm when unfairness can arise (e.g., in the data, algorithm, or the evaluation), it can target different user groups (individual vs group), and it can be measured and evaluated by various criteria [9, 31]. Wang et al. [112] give an overview of different dimensions of fairness definitions. Subsequently, we will provide a concise explanation of these factors and their connection to recommender systems.
Firstly, the focus of a definition can be the process (whether the information used in the process is fair) or the outcome (whether the outcome of a process is fair). Taking into account the outcomes, the question of for whom the outcome is fair has to be considered. Fairness definitions can be divided into group fairness and individual fairness. **Group fairness** considers subgroups divided by some fairness-related attributes (gender, age, race) and considers whether those groups are treated fairly. **Individual fairness** considers

fairness on an individual level. Dividing groups might be more challenging since grouping individuals is not always clearly defined and individuals might belong to multiple groups. Lastly, it must be determined the type of outcome that is considered fair.
There are various concepts (for an overview see [112]) that provide fairness definitions. The most common ones are *Consistent fairness* and *calibrated fairness.* **Consistent fairness** defines that groups or individuals that have similar attributes should receive similar outcomes concerning a particular task. This definition can be derived from the concept of *statistical parity. Statistical parity* defines that every group should receive the same positive or negative classifications as the population as a whole. For example, a minority should have the same chances to receive bank loans as the popularity on average [29]. The goal is to equalize outcomes across protected and non-protected groups. Dwork et al. [29] argue that statistical parity in the form of group fairness is insufficient because it can lead to unfair outcomes for individuals (reduced utility, self-fulfilling prophecy, subset targeting). Therefore, they define fairness definition on an individual level in classification using the Lipschitz condition as a loss function. Dwork et al. state that their metric cannot capture real fairness but will be ”society's current best approximation to the truth”.
Another fairness definition is **calibrated fairness**. It describes that the value of an outcome should be proportional to the individual or groups merit of each individual or group. For example, in a hiring process, a candidate with the most relevant skills, experience, and qualifications should be selected for the job, regardless of gender, race, or any other protected characteristic.

Those definitions give us a broad overview of definitions for fairness in information systems. When evaluating fairness in recommender systems, many factors influence how fairness can be measured. Some approaches, scenarios and systems might have limitations that don't allow some fairness metrics. Amigó et al. [9] investigate and compare various fairness metrics. They come up with 5 fairness dimensions that can be used to categorize criteria. Those categories focus mainly on outcome fairness and were compiled from different works, including [31, 70] and [112]. In the following, we will explain the five dimensions and how groups and individuals can be treated negatively regarding those dimensions.

Benefit

This dimension describes to what extent items are recommended to the user in a fairly distributed way (”Exposure”) or how effective those items are (”Effectiveness”). Exposure relates to statistical parity [29] in the sense that it considers equal distributions of the probability for positive outcomes for each group. Metrics that consider effectiveness focus on the equal quality of recommendations for users across groups. This can mean equal usefulness for each user, but it can also investigate whether items from different providers have equal opportunities to be exposed [9].

Stakeholder

In general, most recommendations influence multiple stakeholders. Those are typically users and the items or item providers. Many fairness measures consider fair treatment for one of those groups, but multi-sided recommender systems have to consider both at the same time [9, 15]. Multiple stakeholders will have fairness concerns, and research has to consider and account for those groups [31]. Unfairness for consumers can influence their lives immensely, for example, if the job offers are worse for some group of users. For providers, unfair recommendations might mean less or worse exposure to their items, which can lead to fewer sales.

Partition Granularity

Fairness can influence not only whole user groups but also single individuals. [9] divide this dimension into ”Two Groups”, ”Many Groups” and ”Individuals”. Often, fairness is considered for certain groups of people. For example, privileged and unprivileged groups. But [29] has shown that group fairness can still lead to unfair outcomes for individuals. Therefore, some metrics consider fairness for each individual.

Exposure Scheme

Presentation in recommender systems varies; therefore, different approaches need different forms of evaluation metrics. Those include fair distribution of ”Ratings”, and exposure in ”Sets” or ”Ranking” fairness.

Fairness Criterion

This dimension describes different assumptions about how the benefits should be fairly distributed among recommendations. ”Parity” assumes that each group should be recommended equally frequently, independently from the group size. ”Size Proportionality” assumes that the exposure should be proportional to the group size, while ”Utility Proportionality” implies exposure proportional to the relevance of the group. This assumes that uniform exposure distributions are not always fair. If a provider provides many useful items, they should be recommended more frequently [9].

In this work, we will generally follow the approach of statistical parity [29] as the main fairness definition. Similar users should receive similar outcomes and similar items should be recommended equally. There are many different fairness definitions, and fairness has to be considered in connection to the use case, the system, the considered stakeholders, and the focus of the research. In the following, we will specify our view towards fairness and create a more detailed definition of fairness concerning our research questions.

2.4.3 The Need for Fairness

Overall, users and providers can be harmed by being unfairly underrepresented or by receiving unfair distributions of items. Often, those unfairnesses are due to biases in the data that is used to train recommender models. While biases do not imply a negative judgement itself, some of them are fairness problems or they cause them. As explained in section 2.4.1, there are different sources for those biases. They can occur because of measurement errors or because of social inequalities that are represented in the data.
If those biases are further exposed or reinforced in algorithms, they can systematically under-represent user groups and harm them. For users, a system can be considered ”unfair if its output is discriminatory in the content provided to different groups” [31]. For providers, unfairness mostly arises when their items are systematically less exposed to users of an RS.
Ekstrand et al. [31] define two different types of harm that can be created by unfairness. Firstly, *distributional harm* is based on classical welfare economics and describes that resources are not equally distributed for different groups of users and providers. This can be compared to Amigó’s [9] definition of ”Exposure fairness”, which also describes the fair distribution of benefits among stakeholders. Additionally, items and users might receive harm because of misrepresentation of their properties and information, leading to *representational harms*. This can be harmful because, for example, the item is not represented correctly and cannot be discovered, but it can also lead to unfair distribution of their data.

In conclusion, we argue that computer systems and especially recommender systems hold the potential to adapt unfairnesses and biases from data and reinforce negative effects. While holding this potential, it should also be considered how those unfairnesses can be overcome and the effects of biases can be mitigated. The main goal of many recommenders is to recommend the most fitting items, those recommenders are aiming at high accuracy in their outcomes, meaning that they retrieve the most fitting items based on their measurements (e.g., the rating of the user). Nevertheless, there might be other goals regarding fairness and mitigation of biases that should be considered when designing recommenders [52].

2.4.4 Fairness in MRS

Music Recommender Systems have unique challenges and are faced with various biases and inequalities that should be considered.
When considering stakeholders, three main stakeholders are influenced by the recommendations of a system. Those are the users, the providers, and the platform [26]. The main goal of the platforms that created the recommenders, is to fulfil a business goal. Regarding MRS, this is mostly to reach high performance for their user in recommending fitting items to keep them engaged with the platform to gather and retain many users and subscribers [15]. Those platforms are typically not at risk of unfair treatment [2]. Fairness-aware systems consider mostly users and providers.
Users of an MRS are typically people who use music platforms to receive personalised recommendations. MRSs can be unfair to users in various ways. For example, it can be considered a fairness problem when the platform provides less well-fitting recommendations for users with uncommon or niche tastes [2, 41]. But recommendations can also be worse for some demographic user groups. [32] has shown different degrees of satisfaction with recommendations for users from different age and gender groups. Overall, different user groups are not treated equally by some algorithms [2]. As mentioned before, fairness requires a normative judgment. When considering that some user groups receive worse recommendations just based on their group membership, this can be considered a fairness problem.
There are various providers of music, including artists, labels, and producers. In the following, we will mainly focus on artists as direct providers of music to the users. Those can be treated unfairly when their items (songs, albums) are not recommended equally. Providers can be grouped by various attributes like gender [34, 36], genre, locality, contemporaneity [84], and popularity [5].
Dinnissen and Bauer [26] conclude that multi-stakeholder approaches that consider multiple stakeholders at the same time are scarce. Since many platforms need to attract and satisfy both, users and providers, multi-stakeholder algorithms, designs and evaluation become more important [2]. In the music domain, platforms need to satisfy their users to gain engagement with the platform, as well as the providers that offer the music on their platform.

There are various biases and unfair outcomes in the music domain regarding recommender systems analysed and evaluated. Approaches that aim at fairer recommendations often focus on more diverse recommendations instead of perfectly accurate recommendations [17, 84]. Diverse recommendations can lead to more novel and interesting recommendations for the user, but they can also contribute to fairness by considering more diverse items; those that deviate from the typical taste. Nevertheless, diversity is not clearly defined. Generally, it relates to internal differences within parts of an experience [17]. Concerning music recommendations, recommendations would be perceived as diverse if they include songs of different styles. But not only the genre is part of diversity. Oliveira et al. [84] include 4 aspects in their diversity definition: Contemporaneity, Locality, Gender, and Music Genre. Even if research on improving diversity does not aim directly at creating fairer recommendations, it is clear that even distributions within those groups can contribute to fairer recommendations by reducing systematic under-representation of some groups. A different definition of diversity only refers to the similarity between songs in an embedded vector space [10]. Overall, low diversity can lead to some items being recommended more frequently and therefore lead to unfair exposure of other items [17]. [45] assumes that increased diversity can create more fair recommendations and, therefore, healthier consumption.

Considering Music Recommender Systems, two overarching fairness problems are analysed and tried to mitigate in current research. Those problems arise from gender imbalance and popularity bias [26].
The **Gender bias** is studied from a user view as well as the artist's view. Regarding providers, Smith et al. [105] analysed that across 1,000 popular songs between 2012 and 2021, only 21.8 of the songs were by a female artist. This effect is even more severe when observing the portion of female producers. [34] find this effect in current music streams, where only 21.75% of streams were from a female artist in non-programmed streams and 23.55% in recommended streams. The authors showed a relationship between organically consumed and recommended songs. This shows a general effect that female artists are under-represented on music streaming platforms and that the proportion of streamed songs by female artists in organically consumed streaming is correlated with their proportion in recommended songs. If a user listens to fewer songs by female artists, they get fewer recommendations from female artists. This leads to the question, of whether music platforms hold the potential to overcome this unequal representation of gender in the music industry. An effect of unbalanced recommendations is also found by [36], but they can increase the number of female artists in recommendations gradually without severe negative effects on the performance of the recommender.
With a focus on fairness towards users, the presence of gender bias can have adverse effects on them as well. Melchiorre et al. [77] found that many collaborative filtering algorithms provide worse recommendations for female users in terms of coverage, recall and ranking quality. Additionally, more accurate algorithms provided less fair results in terms of gender exposure.
We are aware that a binary gender definition is oversimplified but current research focuses mainly on a two-gender perspective. There is a general need for more research on gender differences and how those affect fairness in MRS. Nevertheless, studying gender representation is challenging, and can lead to errors when labelling is done without self-identification [34].
Popularity bias is mainly considered a fairness problem for providers [26], but it can also influence users directly [39, 67]. Users that have less interest in popular items might receive worse recommendations because recommendations are biased towards popular items [2]. Users are differently affected by the popularity bias [69], and different interest in popular songs leads to different levels of satisfaction given popularity bias-mitigated recommendations [64, 67].
Nevertheless, research shows that current MRS favour popular items [64]. This leads to important implications for artists. While the popularity bias implies negative impacts on user satisfaction, it also impacts artists' financial and career prospects [31]. Their financial success depends on streams on the platforms and less exposure can lead to lower public visibility. In the following sections, we will explain the popularity bias and the long tail effect, as well as its implications for users and artists. Following this, we will discuss methods to mitigate the popularity bias, and how those can influence users' music perception.

2.5 Popularity bias

The popularity bias was early formulated as the *Matthew Effect* in Science [78]. It was described as a rich-get-richer effect. Popular scientists get disproportionately credit for their contributions and work, while less-known scientists get little credit for comparable work. If people recognize a popular name on a paper that is probably the person they pay the most attention to even though they might not be the main contributor to the paper. Generally, the popularity bias describes a minority of popular items that are overly exposed to users while a majority of items remain unrecognised [74]. Consequentially, the popularity bias is not limited to information systems but represents a societal bias.
This effect can be seen in recommender systems as well. Very popular items tend to be recommended more and therefore gain more recognition. Data is naturally imbalanced [46], and in section 2.4.1, we described how biased data is generated. In the data that is used to model and train recommender systems, unbalanced data towards popular items can be found [19, 87, 121].
This imbalance is defined as the long tail. Data is typically divided into the head, a small number of very popular items that have many ratings and lots of interactions, and the long tail, which consists of many items that are unpopular and have few interactions [11, 18]. Music data is divided into *Hits* and *Niches*. There is a very small number of hits that account for most sales, while the niches will mostly stay unrecognized and don't sell well [11].
Typical definitions [7, 67, 69] define *head* items as those items that receive the top 20% of interactions with the users. Those are typically very few very popular items. *Niche* items receive the least interactions and take up 20% of all interactions. These are typically many items with few interactions. The rest of the items belong to the *mid* section. The definition of head and tail equals a qualitative description of the phenomenon. A more quantitative definition describes the popularity of items as a heavy-tailed distribution [18].
This phenomenon can be seen as a fairness problem because a system should not systematically favour items that are already popular or well-known when there are items with similar or even more fitting qualities that stay unrecognized [31]. Additionally, a system that prefers popular items does not necessarily provide better options in comparison to when the system is manipulated towards providing fairer items [19]. Recommending items from the long-tail can be highly valuable to users [87].

2.5.1 Reasons for Popularity Bias

There are various reasons why recommender systems reinforce the popularity bias. Intuitively, barely rated items are harder to recommend. When fewer ratings exist for an item, the accuracy is reduced significantly [87]. Accuracy refers to the number of recommended items in the test set that are known to be liked by the user. Additionally, unpopular or new items create a specific challenge for recommender systems. Fewer data exist for those to base their recommendations on [13, 102]. The lack of data for new users is explained as the cold-start problem. Collaborative filtering approaches cannot recommend those items since no ratings for those items exist when they are new. Similarly, algorithms that rely on metadata about the providers and items (context-based filtering) will have less accurate and less complete metadata for unpopular providers [34].
Considering new users, Yalcin and Bilge [116] found that newer users are faced with many highly popular recommendations. The longer a user interacts with a system, the more they experience long-tail items.

2.5.2 Popularity Bias in Recommender Systems

When observing the data for recommendations, it is a common phenomenon that many ratings exist for a few popular items. Having some items that are very popular and well-known to the population is very common. Those items are consequentially rated more frequently explicitly, but also implicitly because they receive many interactions. This is due to biases in society, where some famous items are simply more known, but also because of the feedback loop of a recommender [1]. Popular items have a high chance of being recommended; therefore, they receive many ratings from users again. Consequentially, recommenders intensify the popularity bias even more strongly [121]. This phenomenon of reinforcing the popularity bias can be called algorithmic **popularity lift** [6].
That recommenders inherit the popularity bias is well known. Analysing common recommendation algorithms, Abdollahpouri [1] found that the popularity bias was identifiable in the data (3% of the items take up more than 20% of the ratings), which was even intensified by the algorithms where 3% of the items of the dataset were shown in 60% up to 100% of the recommendations. A similar result was found by Jannach et al. [52]. They tested different algorithms on different datasets and found a reinforcing characteristic of most of those algorithms. Many typical recommender systems make use of collaborative filtering approaches. Those are very prone to popularity bias [122]. Overall, most state-of-the-art recommender algorithms show popularity bias and popularity lift [1, 22, 52, 69, 115]. Unsurprisingly, exceptions are not personalised approaches like `Random` and `ItemAVG` [115].
Generally, popularity can be defined as a confounding factor that affects the probability of an item being recommended [119]. Zhu et al. define this problem as the popularity-opportunity problem. If two items are equally fitting, the more popular one will have a higher chance of being recommended [122]. Cremonesi, Koren and Turrin [22] found that few of the top popular items can skew the performance of classical recommender systems that rely on minimizing the RMSE. A test set that includes very popular items leads to biased results. Those top items can be seen as trivial choices. Testing on non-trivial items leads to a more accurate evaluation of an algorithm.
Nevertheless, the popularity bias does not always create problems for the effectiveness of an algorithm. Although popularity does have a confounding property that should be eliminated, it can be argued that popularity indicates the high quality of an item (and avoids low-quality items [19]), and it can be useful to a certain degree [119]. It has been shown that algorithms that solely rely on recommending popular items can achieve similar accuracy as some personalised approaches [22]. Some argue that recommending by popularity is not necessarily a sub-optimal approach. Cãnamares and Castells [16] discuss that recommendation by popularity is a common approach (e.g., top charts, best-selling lists), and that especially for new users, popular items can be appropriate recommendations. Their study finds a strong bias towards popular items in tested recommender systems. This bias can, depending on the measurement lead to a positive effect.
While popular items might be a good baseline as a recommendation strategy, this approach might stand in contrast to the goal of a system [52]. The popularity bias can lead to decreased diversity, which might be confounding for recommenders who aim at facilitating diversity. Defining additional goals besides accuracy, like reduced popularity or increased diversity, can help improve recommendations for users and providers.
Generally, it can be seen that there are unfair chances for items from the long tail, which are barely recommended to users. For providers, this leads to unfair treatment if they provide items that are currently part of the long tail. For users, this can lead to worse recommendations if they prefer niche items.

2.5.2.1 Popularity Bias in Music Recommendation

In the music industry, popularity can be defined as the total play count of a song or artist. Music consumption is strongly biased towards a small number of highly popular artists, songs and albums [18]. Popular music generally receives more ratings, which makes it easier for collaborative filtering algorithms to recommend them. This promotes popular artists, while non-superstars receive unequal exposure.
It has been shown that music recommender algorithms are similar to other recommenders (see section 2.5.2) and favour popular items strongly [64, 69]. Satisfaction-centric rankers are biased towards recommending highly popular items [45].
The effects of popularity bias in the music domain are similar to those found in algorithms in other domains. MRS have many other challenges. For example, recommending the same song multiple times in separate sessions is often desired. Simple accuracy measurements might not be sufficient in MRS. Therefore, popularity bias can also have various effects on the perception of a playlist, independently from accuracy. For example, popular songs can also evoke familiarity or engagement and receive positive evaluations from users. Therefore, it is important to directly investigate users and their satisfaction with recommendations. We will discuss the results of multiple studies in section 2.5.4.2.

2.5.3 The Unfairness of the Popularity Bias

We mentioned the unfair aspects of popularity bias recommendations previously. Unequal exposure of providers although items are equally fitting raises fairness concerns. Unfairness affects artists and other providers directly. They receive revenue based on the number of plays of their songs, and the number of recommendations influences how well-known they are in the public perception. This "rich-get-richer" phenomenon can lead to high-quality songs remaining unrecognized because they cannot gain recognition by recommendations alone [21].
The direct influence of the popularity bias on users might not be as intuitive. In general, biased recommendations can influence the quality of recommendations. We argued before that those recommendations that are not diverse but are focused on highly popular items, will not provide optimal results. Especially for exploration, uncommon and novel items can be highly favourable. Additionally, popularity bias can create unfairness when different users are influenced by popularity bias to different degrees.
When investigating the personal tastes of users, one can observe certain users prefer niche items more in comparison to popular, mainstream items. Classical algorithms are influenced by popularity bias and will deviate more from optimal results for niche-interested users than for other user groups [2]. Using a simple categorization of niche, diverse, and blockbuster-focused user groups one can investigate how much the recommendations deviate from their expectations. Abdollahpouri et al. [5] assume that if a user has previously consumed, for example, 30% unpopular items and 70% popular items, it would be reasonable to provide recommendations that are similarly distributed [107]. When a recommender over-represents some item group, it leads to deviation from the user's expectation and can lead to dissatisfaction. This phenomenon is called **miscalibration**. A deviation from user expectations is not necessarily considered a fairness problem, but it becomes one as soon as some groups are affected more heavily by it [6]. Miscalibration is heavily influenced by popularity bias and higher algorithmic popularity bias increases miscalibration.
While the interest in popular items is not clearly defined by groups but by a distribution, Abdollahpouri et al. divide this spectrum into three user groups. In their definition, *niche users* make up 20% of all users, and more than half of their consumption history consists of long-tail items. *Blockbuster-focused users* are the top 20% of users, mainly focused on highly popular items (about 85% head items in profile). The rest of the users are defined as *diverse users* who are neither specifically focused on highly popular nor niche items. When algorithms similar to those described earlier produce recommendations that consist

of almost 100% of head items, this can lead to unfair recommendations for diverse, and especially niche users. For niche users, the deviation from their original taste is more severe than for blockbuster-focused users. Ghazanfar et al. [41] observe a similar effect, where niche users (”grey-sheep users”) receive less accurate recommendations because they do not have many comparable other users. Additionally, they can affect the recommendations of the rest of the community negatively. Kowald et al. [63] observe that in three different domains (Music, Movie, Anime), users with niche tastes receive the least accurate recommendations. Less accurate recommendations are aligned with miscalibration and popularity lift, which indicates the negative consequences of these effects on the recommendations for certain users.
Similarly, some user groups are more affected by popularity bias because of their personality traits [116] or because of their gender [69]. Female users are affected more by popularity bias. Yalcin et al. [115] show that various users with certain traits, e.g. *selective users* with low profile mean and high entropies, or *hard-to-predict users* with high entropies and deviations receive worse and unfair recommendations.

Those aspects of unfairness can also be explored in the music domain.
For users that start exploring new music by listening to popular artists initially, it is more difficult to reach relevant artists from the long tail in comparison to users that start exploration from listening to long-tail artists [19]. This, again, shows unfair chances for users who want to explore uncommon items. Kowald et al. [64] reproduced the study investigating the popularity of users who have different interests in niche and mainstream items by [5]. They focused on the music main and found similar results of increased popularity. They also reported significantly worse results for users with low interest in mainstream music. Those results were further confirmed by Kowald in a recent work [63].

2.5.4 The Need for Mitigation

The long tail offers the opportunity for exploration [18], and niche items should not be discarded or ignored, but they can be seen as a chance for creating more engaging and novel recommendations [87].
Mitigating the popularity bias by manipulating the algorithms or the results of the systems towards promoting unrepresented items leads to fairer recommendations for the artists due to more equal chances. A reduction of the popularity bias also improves fairness for users. Because there is a connection between different degrees of miscalibration between user groups and the popularity bias, mitigation of this bias can facilitate the reduction of miscalibration for the different user groups [6]. This leads to fairer recommendations for users too. Authors, like Melchiorre, propose a need for debiasing [77], which is currently not adopted in MRS. Additionally, the outcomes of mitigating attempts should be validated for different user groups.
Overall, there is still no thorough evaluation of whether popularity bias should be mitigated, in which use case mitigation is useful, and to what degree it is wanted by users. In section 2.2.1.3, we highlighted the importance of defining a specific user goal for the recommendation process. In music recommendation, different settings might require different levels of mitigation. For example, in a lean-back session, the user might want to listen to tracks they know and popular, well-known songs might be appreciated. Contrarily, in a lean-in exploration session, the user might want to receive more unknown songs. Personal differences between users can have an immense impact. From an artist’s point of view, there might always be a need for debiasing. If artists are unfairly underrepresented, the system should ordinarily promote unpopular artists; independently from the current use case. Most of the literature studying popularity bias focuses on offline methods to analyse the effect of popularity and how to mitigate it while retaining high accuracy. Offline experiments typically train models on a train set and evaluate the results using a test set. They test whether the system is able to recommend the items from the test set and use performance metrics to evaluate their success. Those analyses might not represent real-world evaluation through users [24].

Conducting experiments with users in experimental settings can lead to more insights into the actual behaviour and perception of users.
To get more insights into when mitigation of the popularity bias is favourable, we analyse literature that focuses on the artists' and the users' views towards popularity bias.

2.5.4.1 Artists' View

Problems like popularity bias and resulting unfair outcomes affect artists directly. To gain insights into how artists perceive those problems and what they propose to do to mitigate popularity bias and other related fairness issues, we analyse three works.
First, Ferraro [36] interviewed artists regarding gender bias in music recommendation. The artists acknowledge that gender bias is a problem in music recommendation, which should generally be addressed. They address that streaming platforms have the opportunity to influence the users' listening behaviour and that fairer content should be supported. They also address that an immediate change is not feasible, but gradually increasing the proportion of underrepresented genders is more appropriate to avoid reactance.
Ferraro et al. additionally interviewed Spanish artists about various other problems of streaming platforms [37]. Artists explain problems to reach a larger audience as a new artist on a platform. This refers to the cold-start problem and is closely related to the popularity bias (see section 2.5.1). Additionally, it is not clear to artists why their music does not get recommended. Regarding popularity bias, all artists acknowledged the popularity bias and agreed to a need to recommend less popular music more intensively. They presume that a strong bias towards popular artists can harm the music culture; therefore, promoting long-tail items is desirable.
Dinnissen and Bauer [27] followed the interview protocol from [37] and conducted interviews with Dutch artists. All artists expressed a need for the promotion of diverse music to broaden the users' horizons. Artists attribute responsibility to streaming platforms, and many wish for more support for new songs and artists. Generally, the artists expressed that the popularity bias affected them, and they consider it problematic regarding the rich-get-richer effect. Presenting more long-tail items is also a desired goal. Contrarily to [37], artists expressed divergent views towards promoting underrepresented genders in music recommendations.
In conclusion, artists generally perceive a need to promote fairer content on music streaming platforms. Although there are different opinions towards promoting content from under-represented genders, interviewed artists generally agree that users should be exposed to more diverse and unpopular content.

2.5.4.2 Users' View

Interestingly, the evaluation of satisfaction given different levels of popularity differs when conducting studies in experimental settings with users. While offline studies indicate improved fairness outcomes for users if the popularity bias is reduced (e.g., [5, 6, 24, 31, 57, 64]), the perception of real users is mostly not considered. To gain further insights into the actual perception of users, we investigate studies that evaluate the users' perception in experimental settings.
Lesota et al. [68] investigate how well offline metrics that measure the miscalibration, namely the Jensen-Shannon Divergence (see section 3.1.2), reflect the human perception of the popularity bias. While they find that users can perceive different levels of popularity between playlists which correlates to the Jensen-Shannon Divergence, users did not show preference towards certain recommendation lists with different levels of popularity.
Graus et al.'s findings [43] reveal that users tend to perceive playlists that are perceived as popular to

be less satisfying. Nevertheless, it can be argued that popularity can still affect satisfaction positively. Users who show high levels of musical engagement are satisfied with more popular playlists, and popular recommendations lead to increased familiarity, which consequentially influences satisfaction positively. The authors argue that creating a good playlist relies on achieving a good trade-off between familiar items and novel items that lead to discovery. The study followed an approach that is similar to a lean-in session. The participants were asked to actively rate a playlist that was created for them by listening to song previews. The positive effect of familiarity resembles the properties we would assume for a lean-back session. Given the properties of a lean-in exploration session, familiar items should not be beneficial to users, but novel and new items should contribute to user satisfaction. The experiment was performed online. An observation of the users and more qualitative insight would be interesting to gain more insights into the users' perception of the playlist and their goals when investigating the playlist.
On the other hand, Fewerda et al. [39] conducted a similar study, which investigated the influence of popularity bias on perceived familiarity, satisfaction and fairness. They could not identify any relationship between the degree of popularity and those factors. In contrast to [68], they did not find that users were able to perceive different levels of popularity. Nevertheless, participants classified some playlists as fairer. If they were perceived as fairer, they were rated more satisfactory. The conducted study tested different recommendation algorithms but did not attempt to mitigate the effect of popularity bias. Therefore, these works leave room for future exploration and lead to the question of whether users will perceive recommendations with mitigated popularity bias as fairer and, therefore, more satisfactory.

Interviews conducted by Sonboli et al. [106] give insights into users' perceptions of fairness and bias mitigation. The interviews express that the system should be very transparent about its fairness objective, and should explain its reasons for having a fairness objective. Fairness is often perceived as an attempt at manipulation. The authors assume a gap in the interviewee's knowledge about recommender systems. Nevertheless, users were open to using recommenders who are provider fairness aware. It is questionable how well those conclusions can be adapted towards popularity bias mitigation recommender systems. Nevertheless, they show that users are open to fairness-aware systems, but designers have to clarify that they are not tricked into selecting fairer options. Furthermore, popularity bias impacts users as well as providers. The conducted study mainly focuses on provider fairness-aware systems. Arguably, the results would differ if users were aware that they are also impacted by biases or unfairness.
Another study by Porcaro et al. [89] focuses on diversity in music recommendation. Firstly, it shows that assessing diversity is difficult for most participants. The participants expressed that diverse recommendations are desirable. Nevertheless, some users want to stick to familiar music and perceive deviations from their taste as challenging. Overall, the authors conclude that a general willingness to explore more diverse music is important.
Familiarity seems to play an important role when investigating the satisfaction with music recommendations. A study by Ferwerda et al. [38] shows that familiarity increases the attractiveness of playlists and has a positive impact on perception. Nevertheless, more diverse playlists (in terms of distance between songs in a latent vector space) are perceived positively if they enrich the users' taste. This factor is often referred to as discovery. The attractiveness of a playlist is increased if either familiarity is improved or if diversity enriches the user's taste.

2.5.5 Fairness Definition

In the previous section, we clarified the impact of popularity bias and how it can create unfair recommendations for users and artists. In this one, we will clarify our fairness definition for mitigating the popularity bias in music recommendations to ground our mitigation approaches. While user fairness is a crucial evaluation criterion, and this work will also evaluate the created algorithms based on user-centred metrics, we will focus on an item-centred fairness view. For creating a fairness definition based on popularity bias metrics and how mitigation strategies aim to create fairer recommendations. Our fairness definition is divided into five dimensions by Amigó et al. [9].

- **Benefit**
  Our main emphasis will be on achieving exposure fairness. Specifically, within the recommendation lists, our objective is to strive for equal visibility of items with different levels of popularity and therefore promotion of tail items.
- **Stakeholder**
  As previously mentioned, this work will primarily concentrate on ensuring fairness towards artists.
- **Partition Granularity**
  Drawing from previous works and mitigation algorithms (e.g., [3, 7]), we classify the items into three distinct groups: Head, Tail, and Mid items (see section 5.3.4)
- **Exposure Scheme**
  While recommendation algorithms typically create rankings, we focus on the exposure of the items in the top-N recommendations. Therefore, we evaluate exposure in sets.
- **Fairness Criterion**
  The fairness criterion is more difficult to create. While parity aims for even exposure between groups, this might not be suitable for the definition of the groups. The group of tail items is significantly larger than the group of head items (see section 2.5.2). The popularity bias on a group level would be completely eliminated if "size proportionality" was achieved. Items from each group would be recommended equally often on average. Assuming fairness strategies create worse performance [57], complete size proportionality will probably lead to unsatisfactory results. Nevertheless, we strive for this type of fairness and will evaluate varying strengths of mitigation when developing the algorithm.

A user-centred fairness definition focuses on equal performance. A system is fairer for users if every user is equally influenced by the system's effects. Regarding popularity bias, a system is fair if every user group (blockbuster-focused, diverse, niche) perceives the same level of miscalibration. Therefore, user-centred mitigation algorithms and metrics have the goal of reducing the miscalibration for each user group by matching the popularity distribution in the recommendations to the popularity distribution in the natural consumption of the user [7].

2.5.6 Research Directions

In conclusion, some works claim that popularity bias is not a new problem, but is also apparent in "traditional" human-expert recommendations [37], it reflects trends in the data, and can lead to satisfactory results [16, 22]. Therefore, it can be argued that mitigation is not necessary to create effective recommendations. Nevertheless, algorithms often link artists or songs because of popularity levels. Human recommendations link recommendations more easily to less-known artists because of some similarities between the artists. Recommenders often create recommendations because of a mainstream tendency [18]. The popularity bias in algorithmic recommendations reinforces a societal bias and creates unfair chances for unpopular artists based on their level of popularity. Additionally, it influences different user groups to a varying extent and, therefore, creates unfair performance for those groups. The literature proposes

that mitigation of the popularity bias can reduce the miscalibration and differences between user groups [6].
Overall, the results of the observed studies lead to divergent conclusions. While [43] and [68] indicate that different degrees of popularity can be perceived, [39] does not find any correlation. The effects on satisfaction are also not entirely clear. On the one hand, offline studies observe that popularity bias can be mitigated while still creating highly accurate recommendations (e.g., [7, 57, 67]). Therefore, it can be assumed that mitigating biases would improve satisfaction because of increased other factors like novelty [17]. On the other hand, Ferwerda et al. [39] and Lesota et al. [68] found no clear indication of an effect of popularity bias on perceived fairness and satisfaction. Assessing whether mitigated popularity bias can affect the satisfaction of the user positively remains challenging. Popularity can have a positive impact in terms of a feeling of familiarity, which influences the users' satisfaction with a recommendation. Nonetheless, perceived popularity seems to harm satisfaction [43]. Additionally, [39] found that fairness affected satisfaction positively. It remains unclear whether popularity bias-aware recommendations improve the users' satisfaction and fairness perception of recommendations. Nevertheless, users show openness to explore more diverse music, but challenges like openness to exploration and perception of manipulation have to be overcome [106].
Based on these results we presume that mitigating the popularity bias can positively influence satisfaction. It remains questionable whether users perceive less popular recommendations as fairer. Current research does not find clear tendencies of which factors influence the perception and experience of popularity in music recommendations. Multiple factors and how they influence each other have to be investigated. It has to be considered that Ferwerda et al. and Lesota et al. did not test the actual mitigation of the popularity bias. They compared different recommendation approaches that differ in their proneness to popularity bias. Based on Lesota et al.'s analysis [69], all of them increased the effect of popularity bias. None of them reduced the level of popularity in comparison to the data that was used to train the model. And the societal bias on a platform level is not actively reduced through recommendation. The data remains unbalanced and biased towards popular songs. We presume positive outcomes from actual mitigation. To our knowledge, there are no studies that investigate the effect of popularity bias mitigation algorithms on users in a user study.
Nevertheless, we have to consider important factors in which cases reduced popularity bias can lead to higher satisfaction. In basic music recommendations or lean-back sessions [100], the popularity bias can have a positive influence because of increased familiarity [43]. Nevertheless, in lean-in exploration sessions, fewer popular songs could lead to improved exploration, because novel items might be appreciated even more. Graus et al. did not define a clear setting for the user. Therefore, the goals of the user were not clear and dependent on the user's expectations and other situational factors. Setting a clear goal can reduce situational effects by creating clear expectations for the user. Additionally, it creates a higher willingness of the user to explore novel music, which was explained to be crucial by Porcaro et al. [89]. Ferwerda et al. [38] have shown that the attractiveness of a playlist is increased if either familiarity is improved or if diversity enriches the user's taste. We presume that familiarity is especially important in lean-back sessions. When defining the lean-in scenario, the effect of discovery might play a more important role for the users and further increase their satisfaction.
Overall, many factors might influence the perception of music recommendations and popularity bias in the recommendation lists. We aim for a thorough analysis of various factors, and we will include qualitative results to understand the experience of the users fully.
In conclusion like many other authors, we propose that the popularity bias needs to be mitigated. The popularity bias leads to various problems for users and artists. While the impact of reducing the popularity bias for artists is very clear, the impact on the users' experience remains unclear and dependent on the situation. Therefore, we have to investigate the impact fairer recommendations can have on users.

2.6 Impact of fair recommendations

In the previous sections, we argued for mitigating the popularity bias in music recommender systems. We presume that it has a positive effect on artists, whose music is underrepresented because they are new artists or are currently unpopular. Additionally, we argue that it can have a positive effect on users because of increased novelty, especially in lean-in exploration sessions. Additionally, it reduces the unfair treatment of users with different tastes in music.
Nevertheless, a majority of music is consumed naturally, without recommendations. We want to investigate whether fairer recommendations can have an impact on the users' behaviour. If the recommendations that the users receive are fairer, do the users change their behaviour and consume music more fairly? If this is the case, we can argue that fair recommendations can have an impact on reducing societal biases. In this section, we will explore the opportunities of a recommender system having an impact on the users' attitudes and behaviour.

2.6.1 The Effect of Recommenders on User Behaviour

Recommenders have the natural ability to direct our behaviour. Recommender systems provide the user with a limited number of choices and, therefore, direct the user to buy or consume those items. Users search for suggestions from a wide range of items [95]. Naturally, this influences which items are consumed by the users. The recommender system aims at providing choices that they will probably like but there could be additional goals from third parties that specific items are chosen [53].
Nevertheless, recommender systems hold the potential to influence the users' behaviour even further. This can, for example, be investigated by the filter bubble phenomenon. By recommending increasingly similar items, a feedback loop can occur [104] that leads users to consume less diverse, one-sided content. While recommending similar items can be seen as an expected feature [35], this can lead to a filter bubble effect [86]. Considering diversity, a study by Nguyen et al. [82] shows that recommended items become less diverse over time and that the content diversity of the consumed items decreases over time as well. Nevertheless, the researcher also found a "natural" narrowing effect, even if users did not follow recommendations. Interestingly, this effect was even weakened by using collaborative filtering methods in the movie domain.
Generally, a relationship between organic consumed music and recommended music can be seen. For example, Epps-Darling et al. [34] found a positive relationship between the proportion of female artists of songs that were streamed organically and those that were consumed through recommendations. This relationship leads to the question of whether influencing the recommendations could also influence the organic consumption of users.

2.6.2 Nudging

We ask the question of whether it is possible to motivate users to consume fairer music. One approach to accomplish this is called nudging. Nudging is defined by choosing an architecture that is supposed to change users' behaviour to make better decisions [109]. Jesse and Jannach define recommendations by recommender systems as digital nudges [53]. They aim at changing or influencing the users' behaviour towards a goal that is typically providing the best options. Additional goals might be apparent. For example in this work, we defined mitigated popularity bias as an additional goal for the users. Jesse and Jannach identified nudging mechanisms from various works and classified them into four categories:

- **Decision Information** methods change the appearance based on given options. For example, items that fulfil an additional goal are highlighted. The actual recommended items are not changed.
- **Decision Structure** refers to methods that change the structure of a decision. For example, those methods change the order of the items or divide them into categories.
- **Decision Assistance** give the user additional direction towards desired goals, for example, by reminding them of their goals.
- **Social Decision Appeal** expand *decision information* methods by additionally altering the information that is presented to users. Additionally, it aims at the emotional and social implications of the recommendations, for example, by referring to social reference points.

Those methods can have an immediate effect on the users' behaviour. Choices like highlighting or reordering have to be considered when presenting the user with recommendations regarding a specific goal. Nevertheless, a long-term assessment of nudging methods has not been researched sufficiently.

Regarding music recommendation, researchers explored whether users can be nudged towards listening to more diverse music. Liang and Willemsen explored the options for creating personalised nudges that motivate the user to explore genres that deviate from current preferences [73]. They make use of changing the *decision structure* [53] of the recommendation. They change the order of recommendations to promote more distant genres first. They find that users can generally be nudged to explore more distant genres when the order of genres is changed. This process did not improve perceived helpfulness directly because the recommendations were perceived as less personalised, but it provides insights into the possibilities of nudging. When using the system for a longer period [72], they have found positive effects on presenting playlists focusing more on exploration of distant genres improve the helpfulness of the system. Additionally, they felt that the personalisation over time increased. Furthermore, they found an effect that indicates that users who had less personalised recommendations began to explore a bit further away from their previous preferences. Users who started with less personalised recommendation lists continued to purposely select even less personalised items and changed their listening behaviour after the sessions.
A mitigating algorithm can also be seen as a nudge. By promoting unpopular items, for example, by re-ranking, users are directed towards consuming those items.
To investigate whether those nudging mechanisms can create fairer listening behaviour, Hansen et al. perform a study that investigates the potential for creating healthier music consumption [45]. They aim at supporting diverse content, defined as less popular music which deviates from the typically consumed content of the user. By promoting this content they want to reduce the filter bubble as well as the popularity bias effects. They identify a general potential for more diverse consumption since their data analysis shows that users can generally enjoy music that departs from their typical taste and that is less popular. Additionally, they provide methods that can accomplish shifting recommendations towards less popular content.
However, the effectiveness of recommendations depends heavily on the recommender itself. In a survey [66], it was found that there are multiple reasons to not listen to recommendations. Those include the aesthetics of the recommender and external factors, but also a suboptimal recommendation strategy. Poor suggestions can lead to retention. When creating a recommender that purposely deviates from user preferences, it has to be considered that those recommendations can be perceived as suboptimal. This would harm the usage of the recommender. A good design and explanations about the recommendation process can be used to avoid this issue. Nonetheless, the degree to which users become aware of these phenomena depends highly on the system. In the longitudinal study by Liang and Willemsen [72], users did not perceive differences in personalisation although the less personalised playlists with high deviations

from the users' typical taste increased the helpfulness and satisfaction.
Overall, this leads to the suggestion that shifting the users' consumption must be carefully attempted, but nudging the users to explore seems feasible. It remains questionable whether users will still consume the items they were nudged to (e.g., fairer items) in the future. For example, if a user was nudged to consume more diverse genres, they are shortly more likely to explore those genres. It is not clear whether they will further explore those genres in their natural music consumption after the nudging interaction. The theory of planned behaviour [40] explains that for a user to behave in a certain way, they have to form a "behavioural intention" which will lead to the behaviour.

2.6.3 Current Effects

While we presume that it is possible to shift the users' perception through recommendation, the question arises whether current music streaming services can accomplish this goal and what effects their recommendations have on their users. An analysis of two papers on consumption behaviour on Spotify does not show a positive influence on diverse exploration behaviour.
Mok et al. investigate how users explore music on Spotify [80]. They find differences between age groups. For instance, older users are more likely to explore new music, while younger users listen to more diverse types of music. Additionally, exploration occurs in recurring phases. When discovering older music, young users are less likely to find older (before 2014) music.
Additionally, they are less likely to find this music organically. Therefore, older music is discovered less over time. Newer releases are discovered constantly by following releases or receiving recommendations for new music. To analyse how this consumption affects diverse content, Anderson et al. identified the effects of the platform's recommendation on diversity consumption [10]. They identify diversity by analysing the similarity between songs in an embedded vector space. The popularity bias indicates that users are likely to receive recommendations that are biased towards highly popular content. A similar effect can be seen when analysing the diversity of recommendations. The research shows that algorithmic consumption is less diverse than organic consumption. Consequentially, the analysis finds that users who become more diverse in their music consumption do so by shifting towards organic consumption and not listening to recommendations.

2.6.4 Indications for Recommendation Algorithm Design

We have shown that methods exist to nudge users towards fairer music consumption. Nevertheless, current music platforms do not seem to incorporate these possibilities. [10] shows that Spotify's algorithms lead to the opposite effect. However, we have shown that current recommender systems are prone to reinforcing biases. We propose that mitigating those biases can be used as a nudge to achieve fairer music consumption.
In conclusion, Hansen et al.'s work [45] gives much insight into the possibilities of creating fairer ranking methods. Their data analysis shows that users are typically satisfied with less popular recommendations. This offers the chance to provide users with less popular content and shift their consumption towards fairer consumption. Nevertheless, these findings are completely based on offline metrics. A validation with users in a user study is required to assess whether it is possible to motivate the users to consume more diverse content.
Nonetheless, a suboptimal recommendation strategy can lead to retention. An appropriate recommendation strategy has to be discovered carefully. Finally, current research indicates that nudging can change short-term behaviour. A shift in music consumption requires users to change on a long-term basis. Achieving long-term evaluations is time and cost-intensive. Nevertheless, short-term analysis can provide us with

indications about users' behaviour in the future. By nudging the users in the short term, they can form behavioural intentions that will lead to a long-lasting change towards fairer listening behaviour in natural consumption.

In section 2.5, we argued for the need to mitigate the popularity bias. We showed that mitigation to create fairer results for artists, as well as users, is possible. Those can be highly effective, especially in lean-in exploration sessions. In this section, we argued that fairer recommendations can have a positive impact on the users' organic listening behaviour. Therefore, we hypothesize that a recommender system that mitigates the popularity bias can have a positive impact on the user's behaviour. This hypothesis needs to be validated by a study that investigates actual users. In the following, we want to investigate the potential of popularity bias mitigated recommendations to foster fairer listening behaviour towards less popular artists and create highly effective recommendations. When faced with fairer recommendations, do users show motivation towards exploring those contents after interacting with the recommender? To test this, we need to create an appropriate music recommender system that creates personalised, effective recommendations. Moreover, we need to evaluate mitigation strategies and apply those to the recommender system.

CHAPTER 3

Developing the Recommender System

For creating a recommender system that is able to achieve fair recommendations based on our definition in section 2.5.5, we need to investigate different recommendation and mitigation strategies. We define two main goals for the creation of the recommender system. Firstly, we want to create highly accurate and relevant recommendations for the user. The user is not supposed to receive significantly worse recommendations because of the mitigation strategy. Presumably, they will even receive more effective recommendations because they are positively influenced by the reduced effect of the bias and more novel recommendations.
Secondly, we want to reduce the effect of the bias. We explore, and compare, to what extent this will be done. A trade-off between performance and fairness is probably necessary. We will conduct an algorithmic evaluation of the algorithms to identify the appropriate weights of the mitigation strategy. Popularity bias raises fairness problems for users and for artists. For the purpose of this work, we will evaluate methods to reduce unfairness caused by popularity bias for artists and users. In terms of artist fairness, we aim to create a more balanced distribution of recommendations between popular artists and artists from the long tail. In terms of user fairness, we evaluate methods to reduce the miscalibration for individuals to reach the same performance for each user group. As described before, we investigate the users' satisfaction, and we will analyse whether the given mitigation approaches have a positive influence on artist and user fairness as measured by various metrics.

3.1 Metrics and definitions

For a clear understanding of the following recommendation and mitigation algorithms, some definitions and metrics for evaluating those are needed. In the following, we will give an overview of metrics that will be discussed in the following sections. Additionally, we will explain various definitions that are needed for understanding those metrics.

3.1.1 Definitions

User Groups G

Based on [5], we define three user groups:

- **Niche Users** (N) (low-mainstream users) are the bottom 20% of the users in terms of the ratio of popular items in their profile. More than half of their profile consists of non-popular items.
- **Blockbuster-focused Users** (B) (high-mainstream users) are the top 20% of the users in terms of the ratio of popular items in their profile. These users, on average, have more than 85% popular items in their profile.
- **Diverse Users** (D) (medium-mainstream users) are the rest of the users that do not fall in the Niche or Blockbuster-focused category.

Although the term "Blockbuster-focused" refers to the movie domain, we keep this term in the future since it is also clear in the music domain.

Artists Groups A

As described in section 2.5, we define three different groups of item popularity. We adopt those, similarly to [7] for artist popularity.

- **Head Songs** (H) are the songs that receive most interactions. These highly popular songs collectively accumulate approximately 20% of the total interactions or user-song engagements.
- **Tail Songs** (T) refer to the extensive collection of the least popular songs within a dataset. Despite their individual unpopularity, these songs collectively account for approximately 20% of the total interactions or user-song engagements.
- **Mid Songs** (M) include the rest of the songs.

Similar definitions can be created regarding artists. For instance, *head artists* are the top artists that receive 20% of all interactions.

3.1.1.1 Additional definitions

To consistently describe the discussed metrics, we make use of the definitions by [7], and adapt other metrics accordingly.

- L is the combined list of all recommendations lists given to different users. An item **i** can occur multiple times in L if it occurs in multiple recommendation lists.
- L_u is the recommended list of items for a user u or the list of items in the user profile. If a distinction has to be made, this is done by the subscripts r and p, which refer to the recommendation given to the user, or the user profile (listening history), respectively. When the distinction has to be made to define a metric, $L_{u,r}$ refers to the recommendation list given to user u.
  In general, all metrics can be applied to either the list of all recommendations or to the list of all user profiles. This is always shown through the subscript, and can typically be done by replacing L_u respectively. For example, for $GAP_p(g)$ replace L_u by $L_{u,p}$ in the formula; for $GAP_r(g)$ replace L_u by $L_{u,r}$.
- I is the set of all items in the catalogue.
- U is the set of all users.
- $\Phi(i)$ refers to the popularity of an item i. This is typically the number of ratings that an item received. Regarding music recommendation, it can be the number of plays (cf. [69]) of a specific song or the popularity metric provided by the Spotify API. Since we only consider binary feedback, we will refer to $\Phi(i)$ as the sum of users that listened at least twice to the track i. This is done in line with [69] to avoid spurious interactions with songs.
- Γ is the set of items which are tail songs T.

3.1.2 Metrics

3.1.2.1 Performance-based Metrics

In algorithmic analyses, the training set is typically divided into 80% train set and 20% test set. The models are created and trained using the train set. Performance-based metrics measure how well the model performs at predicting the items from the test set that have been originally in the user profile.

- **Accuracy, Precision**
  Metrics like Accuracy and Precision are standard metrics to investigate the performance of a recommendation algorithm. Accuracy is a standard performance metric that gauges the overall correctness of a recommendation algorithm by measuring the proportion of correctly predicted items, both relevant and irrelevant, in the test set. Precision, on the other hand, focuses on the ratio of correctly predicted relevant items to the total number of items predicted as relevant, providing insights into the algorithm's ability to avoid false positives in recommendations.
- **Normalized Discounted Cumulative Gain ($NDCG$)** [4, 67, 113, 115]
  $NDCG$ is a measurement for evaluating the ranking performance of an algorithm. It considers the actual ratings by users as well as the items' positions in the recommended list.

- **Mean Average Precsision (*MAP*)** [103] and [63]
  MAP considers the precision as well as the ranking performance by calculating the precision for each relevant in the recommendation set with the size that corresponds to the relevant item. The MAP is the average of all those precisions. Therefore, it considers additionally to the precision of the system whether relevant items are ranked at the start of the list.

3.1.2.2 Popularity bias focused metrics

Metrics directed at investigating the impact of popularity bias can be divided into item-centred and user-centred metrics [7, 57]. Item-centred metrics focus on distributions of the items in the recommendations. They analyse how items are treated differently by an algorithm. These metrics can further be divided into descriptive metrics (M), metrics that observe equal exposure between item groups ($Agg - Div$, $Gini$), and metrics that specifically observe the exposure of long-tail items (ARP, GAP, $APLT$, $ACLT$).
User-centred metrics focus on how user groups are treated differently by algorithms. These metrics that aim at measuring effects like miscalibration make use of the difference between the metric in the users' previous listening behaviour and in the recommendations.

Item-centred Metrics

- **Descriptive Metrics (*M*)** [69]
  M describes a metric for defining the distribution of a recommendation list for a certain user u. It can be one of the following *Mean, Median, Variance, Skew, Kurtosis.* For example, *Mean(L_u)* is defined as:
  $$Mean(L_u) = \frac{\sum_{i \in L_u} \Phi(i)}{|L_u|} \tag{3.1}$$

- **Average Recommendation Popularity (*ARP*)** [117] and [4, 7, 57]
  ARP calculates the average popularity of the recommended item in each list. It is equal to *Mean(L_u)* on a population level. It can be biased towards low values if some extremely unpopular items were recommended to each user.
  $$ARP = \frac{1}{|U|} \sum_{u \in U} \frac{\sum_{i \in L_u} \Phi(i)}{|L_u|} \tag{3.2}$$

- **Group Average Popularity (*$GAP(g)$*)** [5, 6, 64]
  GAP is similar to ARP and measures the average popularity of the items in the user profiles or recommendation lists of each user u in the group $g \in G$. The average popularity of the items in the recommendations lists is defined as $GAP_r(g)$; the average popularity of the items in the user profiles is defined as $GAP_p(g)$.
  $$GAP(g) = \frac{1}{|g|} \sum_{u \in g} \frac{\sum_{i \in L_u} \Phi(i)}{|L_u|} \tag{3.3}$$
  The GAP metric might not be suitable for music-related datasets since there is a high number of available items [64].

- **Aggregate Diversity (*$Agg - Div$*)** [7, 57]
  $Agg - Div$ describes the ratio of unique items across all users. This refers to the coverage of the items. It aims at compensating for the drawback of ARP. The higher the value for this metric is, the

more items are recommended or in the user profiles. Nevertheless, this metric does not consider how often items are recommended. The distribution of item recommendations can still be very unequal although many items are recommended. The later discussed *Gini*-index considers fair distributions.

$$Agg - Div = \frac{|\bigcup_{u \in U} L_u|}{|I|} \tag{3.4}$$

- **Gini Index ($Gini$)** [7, 57]
  This metric measures the inequality across the frequency distribution of the recommended items. A value of zero would, therefore, indicate an entirely equal distribution in which each item is equally often recommended. The formula was adapted from [28]

$$Gini(L) = 1 - \frac{2}{||p(i|L)||_1} \sum_{k=1}^{|I|} \frac{(|I| - k + 1)}{|I|} p(i_k|L)^{\uparrow} \tag{3.5}$$

  where $p(i|L)$ is the ratio of occurrence of item i in L. The values of $p(i|L)$ are ordered by increasing ratio. The k denotes the position of the item in the ordered list.

- **Average Percentage of Long Tail Items ($APLT$)** [4, 57]
  This metric directly investigates the average percentage of items in users' recommendations that belong to the long tail.

$$APTL = \frac{1}{|U|} \sum_{u \in U} \frac{|\{i, i \in (L_u \cap \Gamma)\}|}{|L_u|} \tag{3.6}$$

- **Average Coverage of Long Tail Items ($ACTL$)** [4, 57]
  This metric explains how much coverage long-tail items get in the entire recommendation. This solves a similar issue in regard to $APTL$ as $Agg - Div$ does in regard to ARP. $APTL$ could receive very high values even though only some long-tail items are recommended, but not a wide range of them. Nevertheless, this metric only investigates long-tail items and not all available items. For clarity reasons, the formula presented in [4] was adapted consistently to the $Agg - Div$ formula. Similarly to the *Tail_Coverage@K* metric in [75], we divide by $|\Gamma|$ to retrieve the ratio of long-tail items in all recommendations.

$$ACTL = \frac{|\{i, i \in (\Gamma \cap (\bigcup_{u \in U} L_u))\}|}{|\Gamma|} \tag{3.7}$$

User-centred Metrics

- **Algorithmic popularity lift ($\Delta GAP(g)$ or $PL(g)$)** [5, 6, 63, 64, 67]
  ΔGAP relies on GAP to identify whether the degree of popularity is lifted by the recommendations in comparison to their natural listening behaviour (user profile). Positive values indicate an amplification of the average popularity level, negative values indicate a reduction of the average popularity level. Algorithmic popularity lift can only account for average changes in the degree of popularity, but not for differences between the distributions.

$$PL(g) = \frac{GAP_r(g) - GAP_p(G)}{GAP_p(g)} \tag{3.8}$$

- **User Popularity Deviation (*UPD*)** [7, 57]
  Abdollahpouri et al. [7] propose this metric to account for deviations of the recommendations' popularity distributions to those of the user's listening history. This metric can be interpreted as the degree of miscalibration for different user groups. Therefore, a perfect value would indicate that the recommendations perfectly reflect the users' interests in each user group with respect to popularity. This assumes that the previous listening history clearly reflects the user's preferences. Therefore, a low value does not indicate a low level of popularity, but a level that matches the user's listening history.
  The distribution is described by the ratio of items belonging to the previously described artist groups. The Jensen-Shannon divergence is used [42] to measure miscalibration, the distance between the two probability distributions ($JSD(P(L_{u,p}), Q(L_{u,r}))$). $P(L_{u,p})$ describes the distribution of the items in the user profile, and $Q(L_u, r)$ describes the distribution of items in the recommendation list for user u.

$$UPD = \frac{\sum_{g \in G} UPD(g)}{|G|} \tag{3.9}$$

  with

$$UPD(g) = \frac{\sum_{u \in g} JSD(P(L_{u,p}), Q(L_{u,r}))}{|g|} \tag{3.10}$$

  with

$$\begin{aligned} JSD(P(L_{u,p}), Q(L_{u,r})) = \frac{1}{2}(&\sum_c P(L_{u,p})(c) log_2 \frac{2P(L_{u,p})(c)}{P(L_{u,p})(c) + Q(L_{u,r})(c)} \\ &+ \sum_c Q(L_{u,r})(c) log_2 \frac{2Q(L_{u,r})(c)}{P(L_{u,p})(c) + Q(L_{u,r})(c)}) \end{aligned} \tag{3.11}$$

  where $P(L_{u,p})(c)$ is the proportion of items which belong to the popularity category $c \in A$ in the consumption history of user u.

- **%Δ-metrics** [69]
  The metrics currently described as item-based metrics can only be used to describe the popularity levels or distributions in the recommendation lists or user profiles. Nevertheless, we can use them by comparing the metrics of the user profiles to the recommendation lists. This enables insights into the effects, the recommendation algorithm has on the popularity bias. This is similarly done by ΔGAP to explain the average lift of popularity for a specific user group. Lesota et al. [69] define a formula to compare descriptive metrics M for individual users. Naturally, %Δ metrics can also be defined on a popularity level (e.g., for ARP). Those would then give insights into the item-centred effects of the popularity bias (similar to popularity lift).
  For two distributions, the recommendation distribution and the user profile distribution, any %Δ metric for a user u is defined as:

$$\%\Delta M(u) = \frac{M(L_{u,r}) - M(L_{u,p})}{M(L_{u,p})} * 100 \tag{3.12}$$

  Positive $\%\Delta Mean$ and $\%\Delta Median$ indicate overall more popular tracks recommended to the user. Positive $\%\Delta Variance$ means that the list of recommended items is more diverse in terms of different popularity values than the user's history. Positive $\%\Delta Skew$ indicates that the tail of the distribution is heavier than in the user's profile, indicating that more items of lower popularity have been recommended. Positive $\%\Delta Kurtosis$ indicates that the distribution of the recommendations is heavier than that of the user's profile. Therefore, it indicates that the distribution is closer to a uniform distribution.

Generally, it is important to understand the differences between the categories. While performance-based metrics generally describe how well a system performs in regard to predicting preferences, popularity-focused metrics aim at explaining the effect of popularity bias. Item-centred metrics explain the distribution of popular items of all user profiles or recommendation lists. For instance, we can identify how evenly items are recommended ($Gini$) and compare values between recommendation lists and user profiles. Therefore, we conclude that item-based metrics give insights into artist fairness. They investigate item distribution on a platform level, and if items are more evenly recommended, this indicates that items are less affected by popularity bias. On the other hand, user-centred metrics analyse to what extent users are affected differently by the popularity bias. For example, different values for algorithmic popularity lift for different user groups indicate that some user groups are unfairly treated. Nevertheless, those can be used to analyse effects like popularity lift for the entire population. In this case, it shows the effects of the algorithm in general.
In general, it is often approached to reach a trade-off between performance and mitigation techniques. For example, [67] defines two goals: Maximizing $NDCG$ and user fairness, defined as $JSF = (1 - JSD)$ for each user group.
Overall, we make use of both metrics to analyse how the algorithms in the next sections perform for promoting fairer recommendations regarding the popularity bias for artists as well as users.

3.2 Selection of a Recommender System

For this study, we aim to observe the effects of recommender systems on popularity. Various advanced recommender systems are based on neural approaches or advanced algorithms. In this work, we select a well-known and evaluated algorithm. By this, we can ensure that we use a recommendation algorithm that is proven to create appropriate recommendations. The focus of this work should be the evaluation and effects of this recommendation approach and the mitigation of its biases.
Schedl et al. discuss different recommendation approaches [100]. They discuss data-driven psychologically inspired models that mimic the human's memory processes or can account for the user's psychological states. Another approach is context-aware approaches. Context can refer to the item's context, the user's context, or the context related to interactions between items and users. Thus, context-based algorithms offer many opportunities towards creating advanced recommender systems [8, 58, 110]. Since we aim to develop a recommender system for a user study, we try to minimize the effect of the user's context. While we acknowledge that the item's context and interaction-related context offer a lot of research potential, we will not consider them in the following.
We follow a similar approach regarding sequence-aware approaches. Sequences are a crucial factor in music recommendation, and a coherent listening sequence improves the experience immensely. Nonetheless, track compositions are more important in lean-back experiences [100]. We later explain that we will investigate the user's behaviour in a lean-in exploration setting. Hence, we do not consider sequence-aware systems for now. Nevertheless, consistency and a good flow of musical pieces remain important.
There are three important and established methods in music recommendations. Content-based filtering approaches consider the item properties to recommend similar items to those a user likes. By analysing the song information, user profiles can be created. Items that match are recommended without considering the information by other users. Collaborative filtering (CF) approaches do not consider the musical attributes but recommend songs based on the ratings of other users. Hybrid methods exploit the advantages of multiple approaches and combine them to overcome their individual disadvantages [100].
Music recommender systems studies mainly focus on CF approaches. Since we want to investigate the popularity bias in commonly used algorithms, we will solely focus on those. Another reason for this is, as explained in section 2.2.2, that CF filtering approaches are especially prone to popularity lift, and they require methods for debiasing [100].

3.2.1 Collaborative-filtering algorithms

Popularity bias reinforcement is found for most algorithms and in most domains. For instance, Jannach et al. [52] show the reinforcement effect for most algorithms in the movie domain. For our analysis, we focus on recommendation algorithms and their effect on popularity in the music domain. We mainly investigate two studies that compare different algorithms towards their proneness to the popularity bias. Initially, Kowald et al. [64] reproduced a study by Abdollahpouri et al. [5]. The original work proposes that the popularity lift becomes unfair if different users are influenced to a different degree by the popularity bias. Therefore, they analyse the increase in the average popularity level between the user's listening history and the recommendations per user group. They analysed different algorithms in the movie domain. Kowald et al. reproduced this study in the music domain. They used the LFM-1B dataset [98]. Abdollahpouri et al. further investigated the popularity lift and miscalibration for different users in the movie domain [6]. Additionally, they analyse the general effect of popularity lift for different user groups.
Secondly, Lesota et al. [69] analyze the effect of the popularity bias on different genders. They compare different metrics to measure the effect of the popularity bias and compare various algorithms. They use a filtered version of the more recent LFM-2B dataset [77]

Another extensive recent analysis of multiple recommendation algorithms was recently done by Yalcin et al. [115]. They evaluated ten different algorithms on four datasets and evaluated them in a user-centred manner.
In the following, we will give an overview of baseline algorithms, as well as popular collaborative filtering algorithms. We will explain how they relate to the popularity bias, and we will select a suitable algorithm for implementing mitigation strategies. As explained earlier, we will mainly focus on evaluations done in the music domain by [64] and [69]. Nevertheless, we will expand them with insights from studies in other domains (e.g., [5, 6, 115]).

Baseline Algorithms

Baseline algorithms are typically not suitable for personalised recommendations. Nonetheless, they give important insights to compare them to personalised algorithms regarding the analysis of the recommendation goals.

- **Random (`RAND`)** [5, 64, 69, 115]
  This algorithm selects a list of random items for the user and avoids already consumed items.
- **Most Popular Items (`POP`)** [5, 6, 64, 69]
  This item always recommends the same set to the user consisting of the top-N most popular items.
- `UserItemAvg` [64]
  This algorithm considers the average listening rating (listening count) for each song as well as typical average deviations in the ratings of the user to predict which items should be recommended to the user.

Naturally, random algorithms should not show any bias towards popularity [115] since the recommended items are randomly selected. The *Most Popular Items* algorithm on the other hand represents the other side of the spectrum since it only recommends the most popular items.

Personalised Collaborative Filtering Algorithms

There are various approaches to recommending items based on collaborative filtering. The most popular ones make use of neighbourhood and matrix factorization approaches, or they use autoencoders.

- `ItemKNN` [25] and [5, 6, 69]
  This algorithm is a neighbourhood-based algorithm, which recommends items based on item-to-item similarity, which is computed through user ratings. It makes use of a predefined similarity metric and selects items that are similar to those previously selected by the user.
- `UserKNN` [93] and [5, 6, 64]
  Similarly to `ItemKNN`, `UserKNN` relies on a neighbourhood approach. In contrast to `ItemKNN`, it uses user-to-user similarity to create recommendations.
- `UserKNNAvg` [64]
  This is a similar user-based collaborative filtering algorithm as `UserKNN`, but it additionally incorporates properties of `UserItemAvg`.

- `SLIM` [83]
  `SLIM` is another item-to-item neighborhood-based algorithm. Nevertheless, the similarity metric is not predefined but learned from the data with a regression model.
- `RankALS` [108] and [69]
  In contrast to the previous algorithms, `RankALS` is a matrix factorization approach. It is specifically designed for creating ranked recommendations using implicit feedback, by creating user and item embeddings.
- `NMF` [48] and [64]
  `NMF` is another matrix factorization approach that uses non-negative matrix factorization but replaces the inner product with a neural architecture.
- `BPR` [91] and [69]
  This is another matrix factorization approach, which learns user and item embeddings. It makes use of an optimization function that aims to rank items based on preference instead of predicting ratings for specific pairs of users and items.
- `MultVAE` [71] and [69]
  `MultVAE` is an autoencoder-based algorithm, which given a user's interaction vector estimates a probability distribution over all items.

There are further algorithms, like $SVD++$ [5] or $LightGCN$ [47, 67] that are common collaborative filtering methods, but are excluded in this work. Most collaborative filtering algorithms show strong effects of the popularity bias. Yalcin et al. [115] find that each of the personalised algorithms has medium to strong correlations between popularity and frequency of recommendation for all datasets. Abdollahpouri et al. [5] show that many items are rarely recommended while some few items are very frequently recommended.
Regarding music recommendations, the results are difficult to compare. The researchers use different datasets, different algorithms, and different evaluation metrics. Kowald et al. [64] find an increase in popularity (based on ΔGAP) for almost every algorithm (in decreasing order: `POP`, `UserItemAvg`, `UserKNN`, `UserKNNAvg`, `NMF`). Overall `NMF` generates the fairest results regarding the popularity lift. Interestingly, NMF also generated the best results regarding performance (MAE).
Lesota et al. [69] find that lower popularity levels do not necessarily indicate worse performance. `SLIM` performs best according to $NDCG$, while increasing the mean popularity level by 49.8%. `BPR` shows less biased results ($\%\Delta Mean = -49$), but also performs less well ($NDCG = 0.129$). On the other hand, `ImplicitALS` [50] which is a predecessor of `RankALS` provides a higher popularity bias ($\%\Delta Mean = 121.8$), and a similar performance ($NDCG = 0.184$).

For selecting a fitting algorithm, we need to consider the requirements for our study. We want to test the effect, a mitigation strategy has on the perception of the users. While some algorithms perform very well regarding accuracy and popularity bias (`BPR` or `NMF`), this might not reflect our goal. Ferwerda et al. [39] found no differences in perception in a user study between the algorithms `SLIM`, `MultVAE`, and `ItemKNN`. Hence, we presume that the initial algorithm does not influence the perception to a great extent, but selecting an algorithm with an already low level of popularity might not be suitable since mitigation might have a barely detectable effect. Therefore, we observe studies that applied mitigation strategies to select a fitting algorithm.
Mitigation evaluations can be done by focusing on user-centred or item-centred metrics (see section 5.3.4). Both types of evaluations for different mitigation algorithms have been done by Klimashevskaia

et al. [57] and Abdollahpouri et al. [7]. Later even accomplished evaluating mitigation on the LFM-1b [98] dataset for music recommendations. Both of these studies use Alternating least squares (`RankALS`) as their base algorithm for applying mitigation algorithms, and they find insightful results regarding mitigation strategies. `RankALS` can be used to create ranked recommendations based on implicit feedback and is effective for ranked recommendation lists [108]. The algorithm's predecessor `ImplicitALS` was used by [38] to create personalised recommendations based on Last.fm profiles. In a recent study, Rendle et al. [92] showed that `ImplicitALS` can, despite its age, achieve competitive results to current benchmark algorithms. They highlight the algorithm's role as a baseline and that it is computationally efficient and scalable. Furthermore, they revisited the algorithm's hyperparameters and training algorithm and were able to improve its performance. This work gives important insights into the algorithm's possibilities as well as implementation. We will investigate how their findings apply to the more advanced `RankALS` algorithm

We select `RankALS` as the baseline for our mitigation strategy. Since it has been also used as baselines in [7] and [57], it enables us to create comparable results. Furthermore, its scalability and possibility to use implicit feedback are beneficial since we use large and extremely sparse music datasets. We did not find a direct comparison between it and comparable recommendation algorithms in terms of popularity bias. Nevertheless, `ImplicitALS` shows a medium effect in popularity lift compared to other algorithms, which is also suitable for our study since we can mitigate this effect and compare the impact of the mitigation strategy on the user. In the following, we will compare mitigation strategies and choose a suitable algorithm for our research.

3.3 Mitigating the Popularity Bias

Mitigating popularity bias within recommendation systems involves various strategies and techniques aimed at reducing the disproportionate influence of popular items in recommendations. Mitigation algorithms address this challenge by modifying recommendation processes to ensure a fairer representation of items, thereby promoting a more balanced distribution of recommended items.
There are various approaches and techniques to accomplish the mitigation of popularity bias. For instance, an early approach to mitigate the effect of the popularity effect includes splitting the item set into head and tail parts and building respective models for the resulting groups [87]. Other early approaches that also focus on using solely collaborative filtering, make use of weighting according to the popularity of an item. This approach can lead to increased accuracy and diversity of the recommendations [121]. There are various other approaches, but generally, one can divide those into three kinds of approaches to mitigate the effects of popularity bias. Pre-processing approaches modify the training data to reduce the bias within the training sets; in-processing methods adapt the training process to create debiased predictions; post-processing approaches modify the results of the recommendation algorithm to re-rank the recommended items to reduce biases [122].
Conventional debiasing methods are more effective for strongly mistreated user groups (e.g., *hard-to-predict* users) in counteracting unfairness and improving beyond-accuracy quality [115]. Nevertheless, they seem to create overall worse accuracy levels than the biased algorithms. Similar results have been found by Klimasheskaia et al.[57]. While the mitigation strategies generally improved metrics regarding reduced popularity bias, the base algorithm (`RankALS`) without applied mitigation strategies performed best regarding accuracy. A comparison regarding performance can not be observed in Abdollahpouri et al.'s [7] study regarding mitigation strategies because they tuned the weight of the mitigation λ to be similar for each approach. Still, generally, they find a decrease in precision when increasing λ.
To gain more insights into the performance of different mitigation strategies, we will explain some of them shortly and discuss their weaknesses and strengths.

3.3.1 Mitigation strategies

Item-centred algorithms

- **ReGularization (`RG`)** [3] and [7]
  ReGularization is an in-processing method that controls the ratio of popular and less popular items via a regularizer. It can easily be added to `RankALS` as an additional objective of the recommendation algorithm. The goal of `RG` is to create fairness between popular and non-popular items.

- **Discrepancy Minimization (`DM`)** [12] and [7]
  This algorithm is also an in-processing method and aims at increasing the number of unique recommended items ($Agg-Div$) using a minimum-cost network flow method. The recommendation model is defined as a supergraph. The mitigating theory searches for subgraphs that favour diversity, as well as rating quality. `DM`'s aim is aggregate diversity.

- **`FA*IR` (`FS`)** [118] and [7, 57]
  `FA*IR` is a post-processing algorithm that divides items into two groups: "protected items" and "unprotected items". "Protected items" are tail items [57] or tail and mid items [7]. The algorithm creates priority queues for protected and non-protected candidates from which it creates a ranked group fairness table. Given this table, it creates a ranked list with a minimum number of protected

candidates. Therefore, it guarantees a certain exposure of protected items in the final recommendation list. The goal of `FA*IR` is also fairness between popular and non-popular items.

- **Personalised Long-tail promotion `XQ`** [4] and [7, 57]
  Another post-processing method is `XQ` which focuses on a balanced distribution of popular and non-popular items in the recommendation lists. It aims at balancing the proportion of head and tail items in the recommendation lists by leveraging the user propensity towards popular items. It also divides the dataset into two parts. Mid items belong to tail items in this case as well. While `FA*IR` and `RG` only aim for fairness in terms of exposure between popular and non-popular item groups, `XQ`'s goal is diversification by incorporating diverse item popularity groups in the recommendation.

User-centred algorithms

- **Calibrated Popularity (`CP`)** [7] and [57, 67]
  In contrast to the item-centred algorithms, `CP`'s goal is to improve calibration, measured by the Jensen-Shannon Divergence. It is another post-processing method that re-ranks the items by considering the user profile and their preferences for popular items. It differentiates between head, middle and tail items. The goal of the re-ranking is to create a similar distribution of popularity categories as observed in the user profile.

All the discussed algorithms can be controlled and have a trade-off between the recommendation goal of relevance/precision and their additional fairness goal. In consistency to [7], we call the trade-off parameter λ. A higher λ indicates more weight for bias mitigation. For `FA*IR`, the parameter p controls the target ratio of protected items in the final recommendation lists. For simplicity reasons, we will refer to p as the weight of the mitigation as well.
In the following, we will analyse the evaluations of the five mitigation algorithms. In the movie domain, `DM` performs best on the item-centred popularity metrics followed by `XQ`, according to [7]. However, `CP` performs best regarding the user-centred metric UPD. Klimashevskaia et al. [57] show a similar effect. While they did not investigate `DM`, they find that `CP` performs best regarding calibration (UPD) while `XQ` outperforms `CP` regarding long tail exposure metrics, which were only partly investigated by Abdollahpouri et al. In contrast to Abdollahpouri et al.'s findings, `CP` performs better than `XQ` on metrics that investigate equal exposure ($Agg - Div$ and $Gini$). In conclusion, `CP` outperforms item-centred mitigation strategies regarding calibration, but item-centred algorithms can outperform it regarding long tail exposure. The performance regarding equal exposure is not consistent between the conducted studies.
Unfortunately, only Abdollahpouri et al. investigate the performance of the mitigation strategies on a music-related dataset as well. The results, in this case, show that `CP` outperforms the item-centred algorithms in each metric when comparing mitigation algorithms weighted to have similar performances. Nevertheless, the only metric that was tested regarding long-tail exposure is ARP, which can be biased if the recommendations include some very unpopular items.

To summarize, while the user-centred mitigation algorithm `CP` seems very promising, its optimal performance would be to match the users' preferences in popularity exactly. It makes sense that `CP` performs best regarding UPD since the mitigation strategy directly aims at reducing miscalibration. While this might benefit users in general as it represents user fairness, it still reflects the popularity bias shown in organic consumption. Item-centred metrics regarding long-tail exposure do not clearly favour `CP` over the item-centred algorithms. Those item-centred algorithms can represent artist fairness and more exposure of unpopular artists more clearly by ensuring exposure of long-tail items. This can be seen since they

outperform `CP` regarding long tail exposure metrics in the movie domain. Klimashevskaia et al. [57] conclude that user-centred mitigation strategies might not be the best choice when the goal is to reduce platform-wide popularity effects. While this effect has not clearly been shown in the music domain, we presume similar results regarding the long tail exposure and, therefore, artist fairness.
Generally, since we generally want to promote long-tail items, we select an item-centred algorithm. However, we additionally implement `CP` for comparison reasons since it also performed very well on some item-centred metrics and it provides insights into the prospects of user-centred mitigation strategies.
Because `CP` is a post-processing method, we select `XQ` which is also a post-processing method. Another advantage of `XQ` is that it was validated by two separate studies. Even though the results are not entirely consistent, we aim to create new insights by implementing the algorithm on a music-related dataset. Nevertheless, `XQ` does not entirely focus on the item-centred metrics. The algorithm considers the user profiles and has the secondary goal of matching the distribution of popularity categories of the user profiles in the recommendation lists. For further comparisons, we additionally implement `FA*IR` which focuses solely on ensuring exposure of long-tail items. Despite performing slightly worse in terms of item-centred and user-centred popularity bias metrics, it still has a decent performance.
In the next section, we explain the implementation as well as the algorithmic evaluation of the recommendation algorithm and the mitigation strategies.

Chapter 4

Recommendation and Mitigation Algorithm Development

The development of the Music Recommender System was accomplished with Python 3.10.9 [1]. Additionally, Python packages were installed that supported the development of the algorithms. Those are:

- `spyder`[2]: An integrated development environment (IDE) for coding and testing.
- `pandas`[3]: A data manipulation library providing efficient data structures and analysis tools.
- `seaborn`[4]: A data visualization library based on Matplotlib, offering a high-level interface for statistical graphics.
- `matplotlib`[5]: A comprehensive plotting library for creating publication-quality figures and visualizations.
- `numpy`[6]: A package for scientific computing with support for large, multi-dimensional arrays and mathematical functions.
- `scikit-learn`[7]: A machine learning library with various supervised and unsupervised learning algorithms and evaluation metrics.
- `implicit`[8]: A collaborative filtering library designed for recommendation systems on implicit feedback datasets.

After developing and testing the code locally, the models were trained and evaluated on the Dutch national supercomputer *Snellius*[9]. Utrecht University provides 10,000 credits, which result in 10,000 CPU core hours that were utilized for training and evaluation.
Generally, the development can be summarized into five steps consisting of multiple tasks[10]. Those tasks were split into separate scripts to enable free and easy access to the scripts' functionalities. The five main steps are:

[1]https://www.python.org/
[2]https://www.spyder-ide.org/
[3]https://pandas.pydata.org/
[4]https://seaborn.pydata.org/
[5]https://matplotlib.org/
[6]https://numpy.org/
[7]https://scikit-learn.org/
[8]https://github.com/benfred/implicit
[9]https://www.surf.nl/en/dutch-national-supercomputer-snellius
[10]The code can be accessed at https://git.science.uu.nl/0982717/mitigatingpopularitybiascode.git

- Pre-Processing
- Data-Analysis
- Algorithm Development
- Training and Recommendation Functionality
- Evaluation

In the following sections, we will further explain the steps and the development.

4.1 Pre-Processing

We inspect the data provided by the LFM-2B dataset[11] [99]. It provides listening records from February 2005 until March 2020. We use the listening counts subset, which includes information about how often each user interacted with the songs they listened to. It keeps track of 519,293,333 user-artist interactions, consisting of 2,014,164,872 listening events, 50,813,373 tracks and 120,322 users. We create a subset in line with Lesota et al. [69] and Melchiorre et al. [77]. They only consider user-track interactions with a play count > 1 to avoid spurious interactions. Furthermore, only tracks listened to by at least 5 different users and only those that listened to at least 5 tracks are considered. In a later step, we further reduce the dataset by filtering out songs that have no matching Spotify URI. The URI is the unique identifier for the song for the Spotify API and is necessary for combining the recommender system with the Spotify API. We validate this reduction by comparing the distributions of the dataset before reduction and afterwards. This shows a similar distribution of interactions and popularities.
The main dataset consists of three columns: `user_id`, `track_id` and `count`. Each row represents the number of interactions of the users with the tracks they listened to. For pre-processing the rows are loaded in batches and interactions with counts that are less than 2 were removed. After this, interactions are removed iteratively if the user has listened to less than 5 tracks or the track was listened to less than 5 times. The new, reduced dataset is saved.
We will only consider binary interactions. Therefore, we only consider whether a user interacted with a track or not. This is done to represent the implicit feedback most accurately for the recommendation algorithm. The reduced dataset is used as the main dataset for further development and analysis.
Additionally, we created datasets that save the popularity category of each track as well as the count of the interactions with this track and a dataset that saves the categorization of the user profiles (see section 3.1.1), and their ratio of songs that are head, mid or tail items in their profile.
For later analysis, we created two complementary datasets that save the popularity category of each track and the user category of each user. We refer to them as `track_popularities` and `user_profiles` respectively. The categories are in line with the definitions from section 3.1.1. To create `track_popularities`, the tracks are ordered by their number of interactions. The most popular tracks that overall received 20% of the total interactions are classified as *head* items. The least popular tracks that received 20% of the interactions are classified as *tail* items. The rest of them are classified as *mid* items. For accessing the data the category is saved next to each `track_id`.
`user_profiles` is created by calculating the ratio of *tail*, *mid* and *head* items for each user and sorting them by the ratio of head items. Based on this order, the user types are assigned according to our definition. The first 20% of the users are *Blockbuster-focused* users, the last 20% are *Niche* users and the rest

[11] http://www.cp.jku.at/datasets/LFM-2b

are *Diverse* users. The data frame assigns the ratios as well as the user type to the user ids.
Furthermore, for the training, a user-item matrix is created, which saves each user's rating for each item. Since we only consider binary ratings, the entries are either 0 or 1. For keeping the computations small, the matrix is saved as a scipy sparse matrix[12]. This type of matrix only stores the indices of each value that is not 0. Since the matrix is very sparse and includes a lot of 0-values, which are not stored, the matrix is significantly smaller. The matrix is created by filling the empty sparse matrix with values from the dataset. Since a sparse matrix only stores indices and not the user and track ids, two dictionaries are created that store the matching user id for each index value.
Finally, a train and a test matrix are created by randomly selecting 20% of the values for testing purposes.

4.2 Data Analysis

The dataset after reduction results in a subset consisting of 211,590,265 song-user interactions, from 117,949 users and 4,825,739 tracks. The matrix has a sparsity of 99.9628%. By filtering for available Spotify URIs, we further reduce the dataset. This results in 171,668,326 song-user interactions, from 117,337 users and 2,238,656 tracks. The user-item matrix has a sparsity of 99.9346%. Our modified dataset is significantly larger than the one by Lesota et al. [67] since they also filtered for listening events in the year 2019 and for users with meta-information regarding age, gender and country. A wide array of tracks is especially important for the study to create appropriate user profiles and recommendations. Lesota's final dataset consists of 10.75 Million user-track interactions. Their dataset has a similar sparsity of 99.81%.
Considering the popularity of the tracks, one can observe a clear popularity bias in the subset. The distribution can be seen in figure 4.1. Only about 0.6% of the items take up 20% of all interactions while about 76% of the items are long-tail items. Tail items are considered items that have 56 or fewer interactions, head items are items that have more than 1390 interactions. Mid items are those items that lie in between. As per definition, 20% of the users are blockbuster-focused users, 60% are diverse users, and 20% are niche users. Filtering shows that blockbuster users are users that have a ratio of more than 43.0726% head items, while Niche users only have less than 13.3394% head items in their profile. Diverse users are in between this interval. One interesting difference to Abdollahpouri et al.'s analysis of the user profiles in the movie domain [5] can be seen when investigating the profile sizes. On the one hand, our analysis aligns with Abdollahpouri et al. and Kowald et al [64] that users that are highly interested in popular items tend to have a small profile size in terms of different consumed tracks (see figure 4.2b). On the other hand, Abdollahpouri finds that the profile size of Niche users is almost twice as large as the profile size of Diverse users. Our analysis shows that the profile size of diverse users is larger than those of diverse users.
Furthermore, figure 4.2a shows that Blockbuster-focused users consume about 50% head items, while this ratio is much smaller for diverse and especially niche users. Diverse and niche users have a similar ratio of mid items, while niche users have much more niche items in their profile.

[12]https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_array.html#scipy.sparse.csr_array

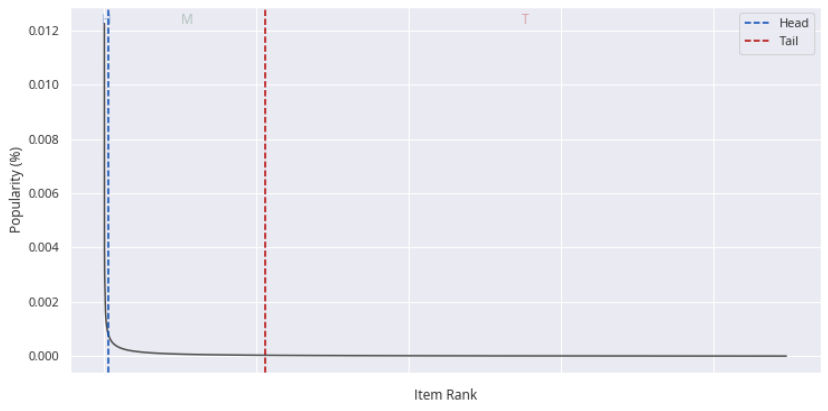
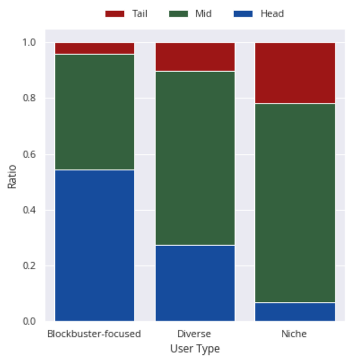
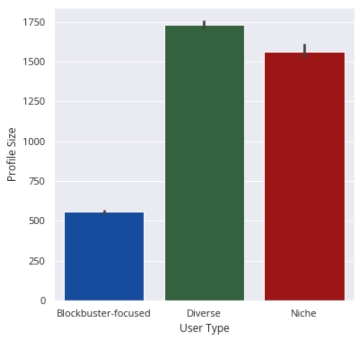


Figure 4.1: Distribution of user-artist Interactions by Item Rank on the Spotify subset
Number of items: Tail: 1,711,292, Mid: 514,543, Head: 12,821
Percentage of items: Tail: 76.4428%, Mid: 22.9845%, Head: 0.5727%



(a) Distribution of popularity groups in user profiles by the user group in the Spotify subset
Blockbuster-focused users ratios:
Tail: 0.0415, Mid: 0.4129, Head: 0.5456
Diverse users ratios:
Tail: 0.1032, Mid items: 0.6221, Head: 0.2747
Niche users ratios:
Tail: 0.2178, Mid: 0.7131, Head: 0.0690



(b) Average user profile size by user group in Spotify subset
Blockbuster-focused:555.2413
Diverse: 1732.0833
Niche: 1563.7192

4.3 Algorithm Development

For consistency reasons, a framework for the models was used. Each model is a Python class with attributes that store values that are important for the recommendation. Those are defined by one of the two main functions `fit()`. This function takes the user-item matrix as input and computes the class attributes. The second function `recommend()` takes the user indices as input as well as the number of items N that should be recommended. It returns a 2-dimensional list consisting of the top N item indices for each user index. Fitted models can easily be saved and retrieved using the `pickle` library.

4.3.1 Baselines

For evaluation and comparison purposes, we develop two baseline algorithms, namely *Random* and *Pop*. Both implementations are straightforward.
Random stores the number of tracks in the user-item matrix and randomly selects N indices in this range while avoiding duplicates.
Pop sorts the track indices by popularity during the fitting. Its recommendation function returns the first N items from the sorted list.

4.3.2 RankALS

The base algorithm `RankALS` is implemented by applying the pseudocode provided in [108]. Since no existing libraries implement `RankALS` in Python, we implement the algorithm ourselves. For development, we make use of the Python `implicit` library[13], which implements the classical `ALS`. Therefore, it provides a base structure for recommender systems. We re-write the `AlternatingLeastSquares` class of the `ALScpu.py` script. This class is a child class of `MatrixFactorizationBase`, which provides the implementation for the `recommend()` function, which computes the matrix factorization and predicts item rankings for users and returns the top N items.
The general approach of a matrix factorization algorithm is to compute item factors for each item and user factors for each user. $P = \mathbb{R}^{|U| \times F}$ is a matrix that represents the user factors for each user $u \in U$. $Q = \mathbb{R}^{|I| \times F}$ represents the item factors for each item $i \in I$. F denotes the number of factors. The vectors in the matrix aim to map items and users into a latent vector space where they can be easily compared. The prediction of the rating matrix $R = \mathbb{R}^{|U| \times |I|}$ which represents the predicted rating for each item for each user is done by computing $R = PQ^T$. The rating for item $i \in I$ for user $u \in U$ can be computed by multiplying the respective factors: $R_{u,i} = P_u * Q_i^T$. This matrix computation aims at a low prediction error as expressed by a cost function. The classical predecessor `ImplicitALS` has a cost function that is able to account for implicit feedback by considering items that were not rated by the user. The vectors are updated to minimize the cost function. Since classical gradient descent is not feasible for huge implicit feedback datasets, the optimization is done by alternating between re-computing user factors with fixed item factors and re-computing item factors with fixed user factors. Each step ensures improvement of the cost function [50]. The user and item factors are initially random.
`RankALS` follows the same approach of alternating between the re-computation of user and item factors. `RankALS` uses a different cost function with a ranking objective. The computation of the user factors is called P-step while the computation of the item factors is called Q-step. Additionally, the algorithm offers

[13]https://pypi.org/project/implicit/

the possibility to provide an importance vector. The importance vector assigns weights to each item to support specific items in the ratings. For example, this could be used to assign higher weights to less popular items to promote their exposure. Since we will only use post-processing mitigation strategies, every item will receive the same weight of 1.
The implementation of the algorithm is quite complex and we refer to the pseudocode of the original paper. In summary, we adapted the loop of the `fit()`-function of the `AlternatingLeastSquares` class. Furthermore, we make use of two Java implementations of the algorithm to ensure the correct development of the algorithm. They are part of the libraries librec[14] [44] and RankSys[15]. The P and Q matrices are iteratively re-computed for i iterations. Furthermore, the function `partial_fit_users()` was created to compute the user factors for one user without having to train the whole model. This is done by accomplishing the P step for the single user.
The model is stored in the main class `RankALS`, which handles the fitting and recommendation. The two main variables that can be adapted in the model are the factor size and the number of iterations. The factor size increases the size of the model but enables a more fine-grained representation in the latent vector space. By increasing the number of iterations, the model ensures a better fit while taking more time.

4.3.3 Mitigation Algorithms

The three mitigation algorithms are functions that receive the initial ranked list $L'_{u,r}$ of length N and the scores predicted by the recommendation algorithm, the popularities of the tracks (`track_popularities`) and the number of items in the re-ranked list. The functions return the re-ranked lists $L_{u,r}$ of length k, which are typically a subset of the initial list with $k << N$. By re-ranking the initial list, underrepresented items that were placed at a later spot in the list get promoted to be within the smaller subset. Personalised long tail promotion (`XQ`) and calibrated popularity (`CP`) additionally take a weight λ as input that represents the weight of the mitigation and the user profiles because they are personalised approaches. All approaches generally follow the same approach. They start with an empty list and iteratively add the items which receive the best score. This score is a trade-off between the score generated by the initial ranking algorithm, and the score that represents the fairness metric.

4.3.3.1 Personalised long tail promotion

Personalised long tail promotion [4] creates the re-ranked list according to the criterion $P(i|u)+\lambda P(i, S'|u)$, where P$(i|u)$ represents the likelihood of user $u \in U$ being interested in item $i \in I$. The likelihood is the score computed by `RankALS`. This term represents the ranking accuracy. The second term promotes diversity between two different categories of items. It denotes the likelihood of user u being interested in an item i as an item not in the currently generated list $L_{u,r}$. This term is the marginal likelihood over the item categories and aims for equal exposure of the categories. The method that aims for equal exposure is called *Smooth xQuAD*. Generally, the second term promotes items from categories with less exposure in the list. Based on this criterion, new items are iteratively added to the list. The full algorithm can be found in the original paper [4].

[14]https://github.com/guoguibing/librec
[15]https://github.com/RankSys/RankSys

4.3.3.2 Calibrated Popularity

Calibrated popularity aims at creating a re-ranked sub-list in which the popularity category distribution matches the distribution in the user's listening history. The distributional differences in the group head, mid and tail are described by the Jensen-Shannon divergence (see section 3.1.2.2). The criterion is described by the maximum marginal relevance $(1-\lambda) * Rel(L_u) + \lambda * JSD(P(L_{u,p}), Q(L_{u,r}))$. The first part describes the impact of the score created by the base algorithm. The second part describes how similar the recommendation list is to the user profile. Items that lead to a more similar distribution receive a higher score. Items are added iteratively. The maximum marginal relevance is computed for the current list with each item appended. The item that increases the criterion the most is added to the re-ranked list. This process is repeated until the desired length is reached.

4.3.3.3 FA*IR

`FA*IR` [118] aims at ensuring a minimum ratio of protected items to ensure fairness for the protected group while maximizing utility in terms of ranking higher-qualified candidates higher than less-qualified ones. The algorithm is grounded in statistical tests. The algorithm proposes a conservative setting of the significance level $\alpha = 0.1$ for the ranked group fairness condition. To avoid false negatives (rejecting fair rankings), the algorithm proposes to create an adjusted significance α_c.
The general algorithm takes the expected size k of the re-ranking, the qualifications q_i for each item i (the score of each item), and the indicator variables q_i, which indicates whether an item is protected. Additionally, p indicates the minimum proportion of protected candidates, and the adjusted significance level α_c. The algorithm returns a re-ranked list that satisfies the group fairness condition. This is achieved by creating two priority queues. One for non-protected candidates, and one for protected candidates. It computes the minimum number of fair candidates at each position in the ranking and constructs a ranking subject to candidate qualifications, and minimum protected elements required. If a protected candidate is demanded, the next best candidate from the protected queue will be added, otherwise the best candidate of either of the two queues.
For our implementation, we made use of the fairsearch[16] library, which offers an implementation for `FA*IR`. Initially, an instance of the `Fair` class of the `fairseachcore.py` script is created, the adjusted alpha is calculated (`Fair.adjust_alpha()`), and a `Fair` instance with the adjusted α_c is created. Finally, the items are classified as protected or not-protected depending on their popularity group. The `Fair` instance is used to re-rank the initial recommendation list.

4.4 Training and Recommendation

Training and Recommendation are processed by the `model_handler.py` module. It incorporates a function that initializes a model and calls the `fit()` function of the model with appropriate parameters. After training, it saves the model using `pickle`. The module additionally offers functionality for loading the models. Finally, it handles predictions and re-ranking with two functions. The function `predict` takes the model, the user ids and the track and user index maps as input, and it transforms the data so that it can be handled by the model. By calling the function `predict()` of the model, it creates a recommendation list of N items. The track indices, the track ids and the prediction scores are, finally, returned.
After creating the recommendations, the output can be fed to the function `rerank()`, which calls one of

[16]https://pypi.org/project/fairsearchcore/

the re-ranking functions with appropriate parameters and returns a list of k items. The function takes the algorithm name and specifications, like the λ as input, and returns the re-ranked track indices, track ids and prediction scores.
Abdollahpouri et al. [7] create recommendations of size $N = 100$ which are transformed into re-ranked sub-lists of $k = 10$ items. Therefore, we initially propose and test to create an initial list that is 10 times as large as the intended re-ranked recommendation list.

4.5 Evaluation

The evaluation of the algorithms is accomplished by two scripts: `evaluation.py` implements the pre-processing of the user profile data and implements the metrics. `full-evaluation.py` calls the metrics and saves them to a csv file. An overview of the different metrics can be found in section 3.1.2. The following metrics are implemented:

- **Performance-based:** Precision, Recall, NDCG, MAP
- **Descriptive metrics:** Mean Popularity, Median Popularity, Variance, Group Average Popularity
- **Item-centred (equal exposure):** Aggregate Diversity, Gini-index
- **Item-centred (long-tail exposure):** Average Percentage of Long-Tail Items, Average Coverage of Long-Tail Items
- **User-centred:** Group Popularity Lift, Popularity Lift, Group User Popularity Deviation, Popularity Deviation
- **Visualizations:** Distribution of popularity categories, Distribution of popularity categories per user group

User-centred metrics require descriptive metrics of the user profile. Since those have to be used various times for different algorithms, we pre-compute those and save them to files, so they can easily be loaded every time they are required. Those include the ratios of popularity groups for each profile, the average popularity of each profile, the track ids for each profile, and the popularity mean, median, and variance of each profile.
Before the metrics are computed the same descriptive metrics are computed for the recommendations for each user. For evaluation, those are used to compute the metrics listed above. The metrics are saved in a data frame and added to a csv file that keeps track of the evaluations. The plots are saved with an appropriate title. In the final evaluation, the metrics are computed for the user profiles, the baseline algorithms (Random and Popularity), the base recommendation algorithm (`RankALS`) with different parameters (# iterations and # factors), and (after selecting one algorithm as the base algorithm) for the three mitigation strategies with varying parameters (λ for `XQ` and `CP`, p for `FA*IR`).

4.5.1 Algorithm Training

In this section, the training process of the Base algorithm will be explained and its results will be discussed. The results of the Base algorithm will be compared to the user profile, and the baseline algorithms *Pop*, and *RAND*. For the evaluation, 80% of the initial user-item matrix was chosen as the train set and 20% as a test set. The train set represents the user profiles, respective metrics are computed on this part of the dataset. Every user profile consists of an average of 1170.43 items. The test set consists of 292.61 items on average. Performance-focused metrics are computed by comparing the recommendations of the system to the test set. Metrics, like popularity lift, compare the user profile (train set) to the recommendation created by the recommendation algorithm. We perform the evaluations on the full set. Therefore, we create personalised recommendations for each user and compute the overall metrics. For each user, the top-25 recommendations are used and evaluated.

The popularity and random algorithm show baselines for simple, unpersonalised algorithms. `RankALS` is initialized with random values; therefore, providing random recommendations. The first evaluation is done after 5 iterations. For simplicity reasons, we omit the evaluation for iteration 0. The results would be similar to the random classifier. The exact values of the evaluation can be seen in the appendix in section A. Optimally, larger models would be trained as well to identify further possible improvements or overfitting, but we have to refrain from doing so. On the one hand, the training for the algorithm takes much time (about 1 hour per iteration for the large model on 32 CPUs) due to the size of the train set. On the other hand, larger models are not feasible for usage in the later study. The large model is 1.4GB large. A larger model would be difficult to handle on a server. The models have been trained and evaluated on the thin model of Snellius (Dutch National supercomputer hosted at SURF).
In the following, we will discuss the different models on performance-related and user and item-centred popularity metrics. Generally, it can be observed that smaller models struggle to fit the dataset well. They perform less well, and they show less popularity bias. This might be due to the model fitting less well to the data and, therefore, creating more randomly selected recommendations. Furthermore, it can be seen that the metrics are stable after 20 to 30 iterations. Further training does not seem to lead to significant changes. In the following, we will relate to the largest `RankALS` model with 128 factors and 30 iterations as the base.

4.5.1.1 Performance-related metrics

The performance-related metrics (see figure 4.3) show that the performance increases with every iteration, and with the factor size. Especially doubling the size from 64 to 128 factors increases the performance a lot. Nevertheless, while all algorithms perform much better than the random ranker, ranking by popularity performs better than all models with less than 128 factors. The base, the `RankALS` model with 128 factors and 30 iterations, only performs slightly better than the popularity ranker.

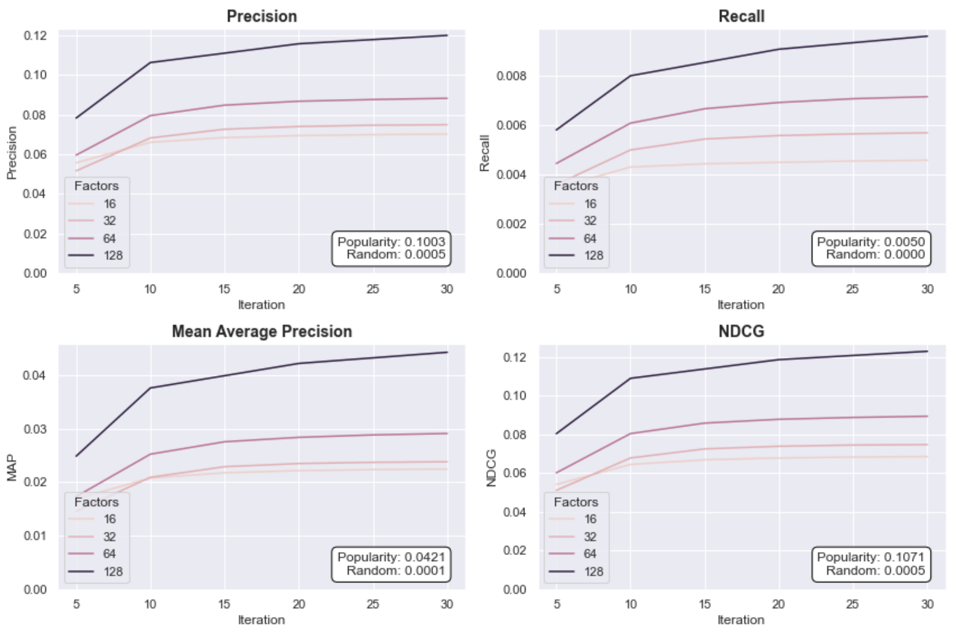


Figure 4.3: Performance across Iterations for Top 25 Recommendations with Different Factor Sizes

4.5.1.2 Item-centred Metrics

An analysis of the item-centred metrics does not show a clear popularity bias at first sight (see figure 4.4). The mean popularity is decreased in comparison to the user profiles, especially for the smaller models. The base model almost reaches the same mean popularity as the user profiles. Nevertheless, the median popularity is almost twice as high in the recommendations in comparison to the user profiles. This shows that the mean is very skewed in the user profiles. The mean is more than twice as large as the median. This indicates that there are some very popular items that skew the popularity. An increased median through the recommendation algorithm shows that the models create more items that have higher popularity than the user profiles.
When investigating the equal-exposure-focused metrics (see figure 4.5), it can be observed that the models create much less diverse recommendations. The recommendations of the base model are only able to achieve a ratio of 0.08 unique items (as defined by Aggregate Diversity). This means that in all recommendations combined, 92% of the items are never recommended to any user. Furthermore, the Gini index of 0.99 for the full model indicates a very unequal frequency distribution across the items. Smaller models show a slightly more equal distribution which can be explained by them not fitting the data that well and, therefore, being influenced by less popularity bias. The cause of the unequal distributions can be explained by the low percentage of long-tail items in the recommendations. While, on average, 18.5% of the songs in the user profiles are tail items, the recommendations consist of only 0.11% long-tail items for the base model. In the smallest model, 1% of the recommendations are still long-tail items. Nevertheless, long-tail items have a very low probability of being recommended. In the base model, only 0.1% of the tail items are ever recommended. This shows a clear popularity bias that tends to not recommend long-tail items. The larger the model becomes and the better the model is able to fit the data, the worse this effect becomes. Figure 4.8 shows that mainly mid items are recommended, and the larger models tend to recommend more head items.

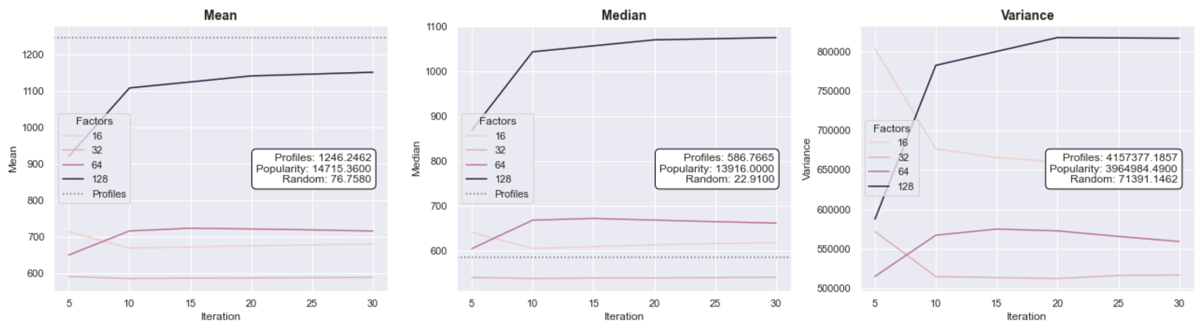


Figure 4.4: Decriptive Metrics across Iterations for Top 25 Recommendations with Different Factor Sizes

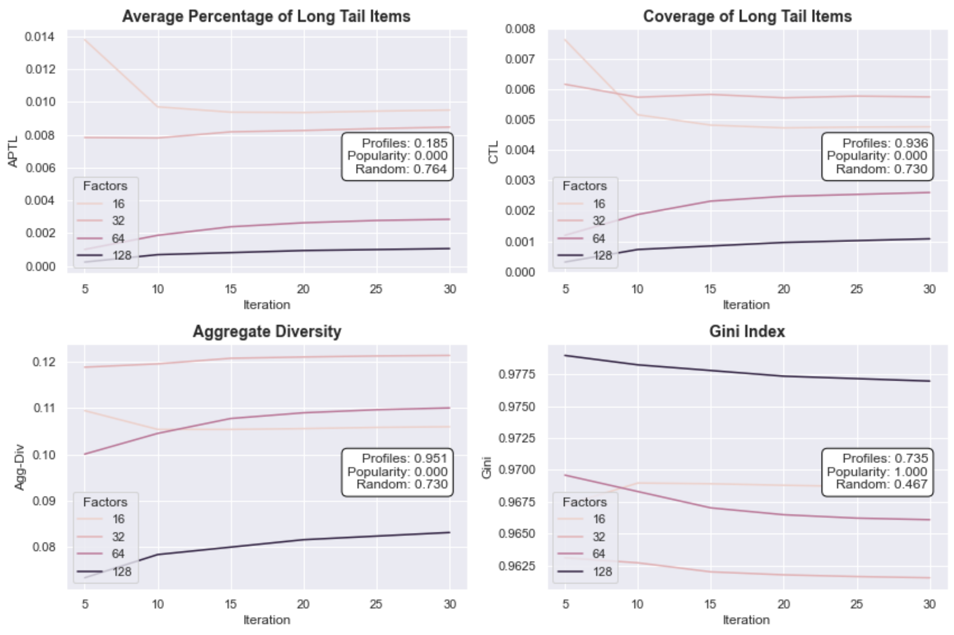


Figure 4.5: Exposure-related Metrics across Iterations for Top 25 Recommendations with Different Factor Sizes

4.5.1.3 User-centred Metrics

The findings from the item-centred evaluation can be further investigated when observing the distribution across different user groups (see figure 4.8). The popularity ranker only provides head items, and the random ranker provides distributions that match the overall distribution of items across the dataset. `RankALS`, on the other hand, mainly recommends mid items. Interestingly, even the smaller models seem to have a tendency to match the user profile. They recommend more head items for blockbuster-focused users and less for niche users. Niche users are the only user group that gets tail items recommended. Nevertheless, the larger the model is, the smaller gets the distribution of tail items, and the ratio of head items increases. The large model still does not recommend as many head items as appearing in the user profile, but it recommends about twice as many as the model with 64 factors. The ratio of head items between the different categories approximately fits the ratio in the user profiles.
Therefore, the user popularity deviation decreases especially for the base model (see figure 4.7). While the larger the model is, the better the overall UPD seems to be, this effect cannot be found for niche users. Since the base model recommends almost no tail items anymore, which make up a significant amount of the niche users' profiles, the user popularity deviation remains high. When comparing the user profiles' mean popularity to the recommendation mean popularity for various groups (see figure 4.6), we can make inferences about the popularity lift (see figure 4.9). Although tail items are almost not present in the recommendations, the overall popularity lift is negative. The mean popularity score is slightly decreased in comparison to the user profiles. Further inspection of the different user groups shows that this might be caused by an over-representation of mid items. In each group, head items and tail items are underrepresented. The ratio of tail items for blockbuster-focused and diverse users is relatively small. Their absence does not influence the mean that much while fewer head items with very high scores will influence the mean much; therefore, the over-representation of mid items leads to a negative popularity lift. Nonetheless, the popularity lift for niche users is high with over 0.5 for the base model. Smaller models show a negative popularity lift, but the base model shows a clear indication of popularity bias since it increases the mean popularity. Head items contribute barely to the user profiles of niche users, but tail items do. Therefore, the absence of tail items and many mid items lead in this case to a positive popularity lift.

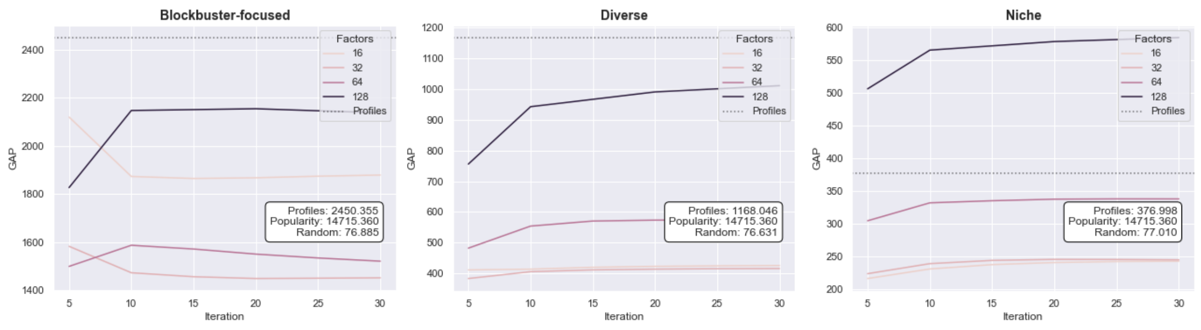


Figure 4.6: Group Average Popularity across Iterations for Top 25 Recommendations with Different Factor Sizes

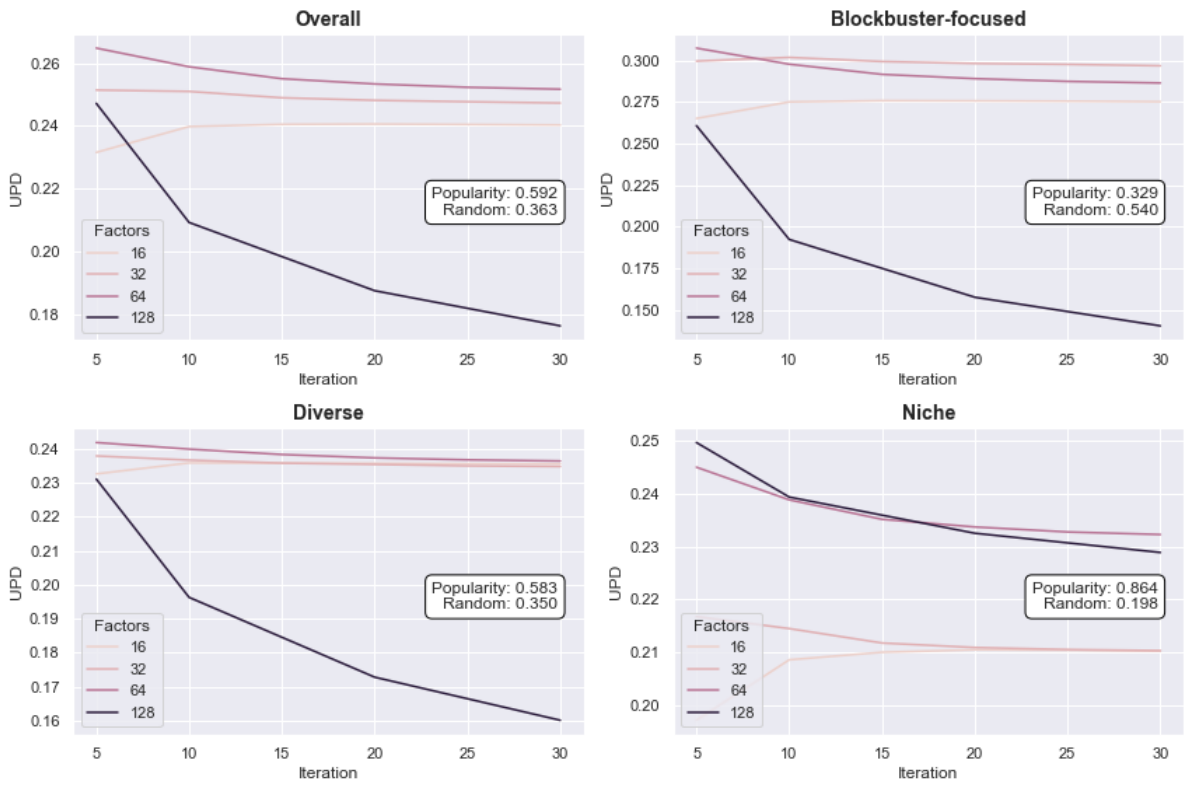
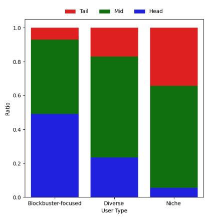
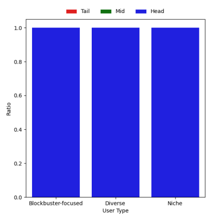


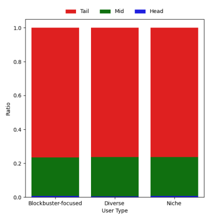
Figure 4.7: User Population Deviation across Iterations for Top 25 Recommendations with Different Factor Sizes



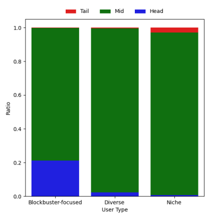
(a) User Profiles



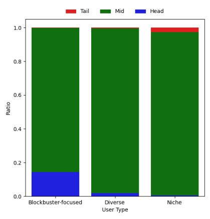
(b) Popularity



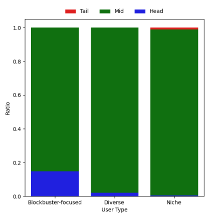
(c) Random



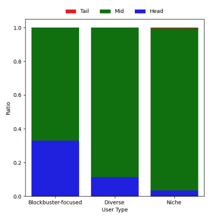
(d) `RankALS`
Factors: 16
Iterations: 30



(e) `RankALS`
Factors: 32
Iterations: 30



(f) `RankALS`
Factors: 64
Iterations: 30



(g) `RankALS`
Factors: 128
Iterations: 30

Figure 4.8: Popularity Distributions for Top 25 Recommendations with Different Factor Sizes

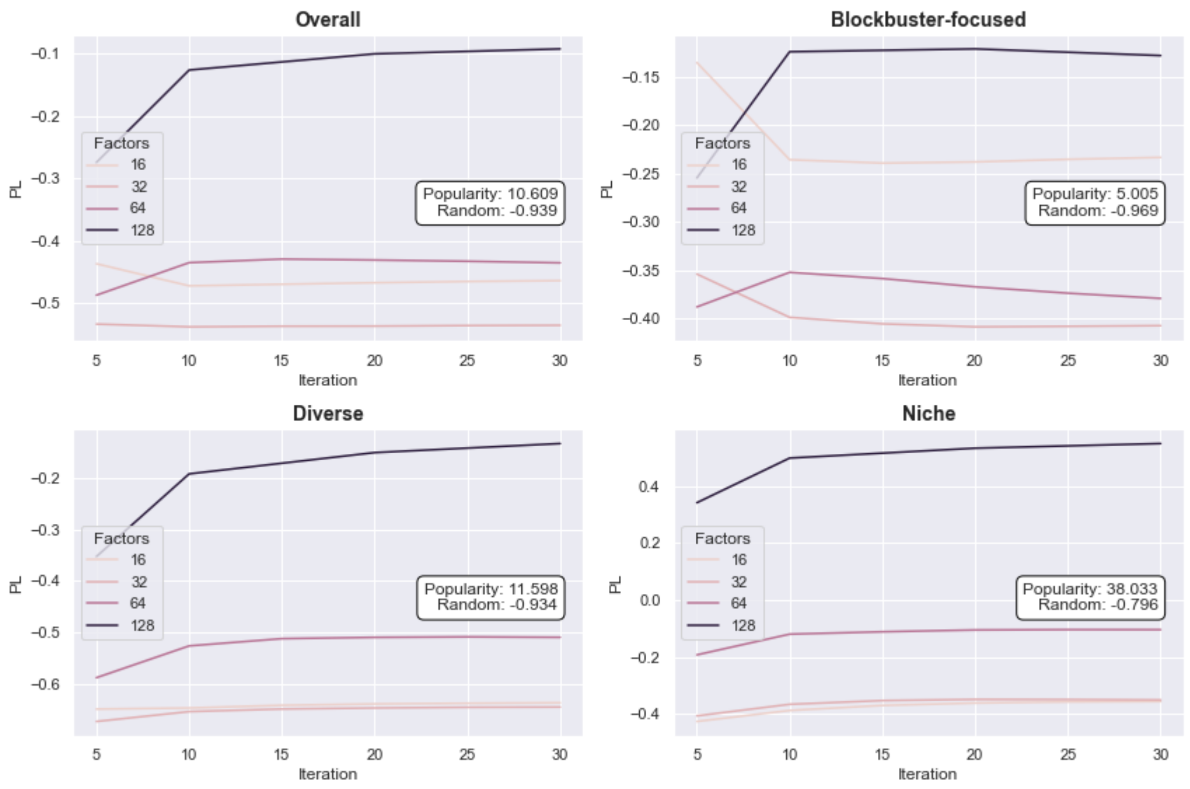


Figure 4.9: Popularity Lift across Iterations for Top 25 Recommendations with Different Factor Sizes

4.5.2 Training Discussion

In conclusion, the evaluation shows some unexpected effects, like a low overall performance, and a negative popularity lift for blockbuster-focused and diverse users. Nevertheless, we argue that the model achieves personalised recommendations and reflects a clearly identifiable popularity bias.
Smaller models that have been trained less show a lower performance as well as lower popularity. This leads to the presumption that smaller models do not fit the data well. The dataset and the number of tracks and users are immense. The number of factors determines the dimensionality of the vector space for the item and user representation. Less than 128 factors might struggle to represent the data appropriately. There might be more of a randomness factor in smaller models, which leads to more diverse and less popular items being recommended. Consequentially, this leads to worse performance and a reduced popularity lift because fewer of the highly popular items are recommended. It has been shown that recommending by popularity can outperform various collaborative filtering models [22]. Popular items have a high chance of being fitting and those non-personalised approaches can be highly effective [16]. On average 20% of the user profiles are head items which only make up 0.6% of all songs. Therefore, it is very likely that selecting a song from this category matches the user profile. The larger models fit the data better, and they are able to learn the advantages of popular items since those have a high success rate and are fitting for many users.
Presumably, a larger model would fit the data even better. Due to time and training resource limitations, it was not possible to test a larger model. The performance increased much when comparing the model with a size of 64 to the base model. We argue, that this is not the best possible performance, but a larger model could outperform the base model. Comparable studies that tested different collaborative filtering algorithms on smaller subsets showed similar performances. Lesota et al.'s [69] show NDCG scores for the top 10 items between 0.1 and 0.2 for most algorithms, while SLIM and ItemKNN outperform them with an NDCG score of over 0.3. They used a subset of the LFM-2b dataset consisting of 23,000 users and 100,000 tracks. While the papers focus on the mitigation, we can infer the initial performance of the `RankALS` algorithm used by Klimashevskaia [57] in the movie domain and by Abdollahpouri et al. [7] in the movie and music domain. While the precision in the movie domain was high in both studies (0.73 in [57] and 0.3 in [7]), the performance on the LFM-2b subset was much lower with 0.075 [7] and is comparable to the performance of our base algorithm. The selected dataset consists of 2693 users and 274,707 ratings and is much smaller than the dataset selected in our study. Generally, recommenders seem to perform worse on the larger music-related datasets than on the smaller movie-related datasets.
Additionally, the low performance can be traced to the structure of the dataset and created recommendations. The dataset we use has more than 2 million songs, about 292 are in the test set for each user on average. For the evaluation, we only create 25 recommendations. The model has to perform very well to select exactly 25 songs that are in the user profile. Therefore, high precision is very hard to reach for a rather small model. That we still create slightly better recommendations than the popularity ranker and much better performance than the random ranker shows that the given approach reaches personalised recommendations that seem to capture the structure of the data. This can also be seen in the distribution of the popularity categories. Not each user gets the same distribution, but blockbuster users receive more popular and head items than diverse users. Niche users receive a low number of head items. We can only partially conclude that we achieve content calibration (matching the content preferences of the user)[7], but some degree of popularity calibration can be concluded, which indicates personalisation. To infer that the recommendations also actually fit the content preferences, further analyses would be necessary. The performance metrics can give us some indication, but further analysis including genre analysis would create more robust insights.
Nevertheless, the evaluation approach cannot capture the whole item space appropriately. Measures like aggregate diversity or Coverage of long-tail items consider the coverage of the items. In this evaluation, only 25 items are recommended per user. A larger number of recommended items would lead to better

values since the model is able to cover a large number of items. Additionally, the high number of items in the data set leads to less coverage. For instance, a much smaller dataset in [57] performs much better in terms of coverage, and [7] reaches Gini scores of 0.92 (in comparison to 0.977 in our study) and Aggregate Diversity of 0.14 (in comparison to 0.08). Nonetheless, we can identify and evaluate different models and compare their performance.

Overall, we argue that the model shows a strong popularity bias. Less than 0.6% of the items in the dataset are head items, but the model recommends almost 35% head items to Blockbuster-focused users. Head items receive much exposure while tail items, which make up most of the items are almost never recommended. The model seems to struggle to recommend those items to the users. Although the distribution of the popularity categories and the median indicate a popularity bias, the model still creates an overall negative popularity lift. We attribute this to the suboptimal fitting of the model. We have shown that larger models focus more heavily on head items and very popular items. The base model shows a positive popularity lift for niche users and a small negative popularity lift for diverse and blockbuster-focused users. Although there is no positive popularity lift for those user groups, the base model still focuses much on recommending head items. Larger models would probably increase the exposure of head items and might increase the ratio of head items in the recommendations in comparison to the user profiles. An interesting observation was made by Kowald et al. [64]. They argue that the metric "popularity lift" might not be suitable for music-related datasets because of the large number of items. In the related studies [5, 6], increased algorithmic popularity lift for users that tend to listen to less popular items or niche users was found. Kowald et al. are not able to replicate this on a subset of LFM with 3000 users and 352,805 artists. This partly is confirmed by our study. On the one hand, there is no clear difference between blockbuster-focused and diverse users in terms of popularity lift, but we clearly find an increased popularity lift for niche users. This highlights the effect of the popularity bias on niche users. We further observe popularity lift in section 4.5.5 and argue that the mean popularity and popularity lift might not be suitable to represent popularity lift accurately.
Comparing the results of the popularity bias metrics to other works is challenging because they use other data sets, subsets of the used data set or different models. The two previously mentioned studies that use `RankALS` [7, 57] unfortunately do not mention or discuss the model size and training iterations. Nevertheless, some comparisons and inferences can be made. First of all, other studies found a more severe popularity bias. For example, [69] found an increase in mean popularity of 121.8% and in median popularity of 316.6%. In our study, a decrease in mean popularity was apparent, which could be related to sub-optimal fit. Additionally, [7] found a much higher UPD (0.466 in comparison to 0.18 in this study). This higher UPD can be related to the number of head items in the recommendations. Abdollahpouri et al. show a high number of head items in the recommendations (about 0.75 for all user groups). Our base algorithm recommends many more mid items, which are also the majority in the user profiles. Therefore, the miscalibration is clearly reduced by focusing on mid items.

In conclusion, we argue that the model creates personalised recommendations while showing a clear popularity bias. From an item-centred view, it becomes clear that tail items are very underrepresented while head items (while not exceeding the ratio in the user profile) are overexposed in comparison to their overall ratio. Many of the recommended items are mid items. Interest in niche items is not reflected by the recommendations. From a user-centred view, it is shown that niche users are affected more severely by the popularity bias. In the base model, they have a larger user popularity deviation, and their high interest in tail items is almost not reflected in the recommendations. Niche users are also the only user group that experiences a positive popularity lift. The user popularity deviation for Blockbuster-focused seems to stem mainly from a lack of head items. The UPD for diverse users is mainly due to a lack of head and tail items. In both cases, increased popularity bias might decrease the user popularity deviation initially even more.

4.5.3 Mitigation

The goal of the mitigation strategies is to mitigate the popularity bias. We use item-centred approaches (`FA*IR` and `XQ`), as well as a user-centred approach (`CP`). We will evaluate and compare the different strategies. Following this, we will select appropriate strategies for the user study.
We do an initial analysis, where we create re-rankings and observe the results. For this, we create initial recommendation lists of various sizes N and inspect the categories of the corresponding recommendations. Furthermore, we apply the re-ranking methods and create sublists of $k = 25$ items. Finally, we investigate the differences between the recommendation lists.
One early observation is that a majority of the first 25 recommendations consist of head and mid items. Originally, `FA*IR` classifies mid and tail items as the protected items which are supposed to be promoted in the re-ranking, and `QX` aims for equal exposure between the group consisting of head items and the combined group of mid and tail items. Since the initial recommendation lists do not consist of many tail items and the item-centred algorithms do not aim to support their exposure specifically but in combination with mid items, tail items also do not occur in the re-ranked lists. As explained in the previous evaluation of the base algorithm, mid items are overrepresented in the top 25 lists. Classifying those as protected items does not change the recommendation lists for a majority of the users. Because of this, we decided to only select tail items as protected items for `FA*IR` and to combine head and mid items for `XQ`. Therefore, the definition of head items for those two mitigation algorithms includes head and mid items while tail items remain the same as the definition in section 3.1.2. This is in line with how protected items were defined by [57]
Nonetheless, the initial analysis shows that re-ranking with $N = 250$ items does not seem to include many tail items in the recommendations despite high re-ranking weights. Observing the initial re-ranking lists shows that even in the top 250 recommendations barely any tail items are recommended. Therefore, the algorithms have no items available to promote in the re-ranking. Therefore, we choose a higher number of initial recommendations and find that tail items in the initial lists can only be assured with about 5000 items. Creating re-rankings on lists that are this big has the disadvantage that the re-ranking takes much more time because the algorithms iterate over the entire initial list. Nonetheless, to create proper re-ranking, we decide to choose those parameters.
In conclusion, we will create initial recommendation lists of size $N = 5000$. The three algorithms will be used to create re-rankings of size $k = 25$. `FA*IR` and `XQ` will aim to promote tail items, while `CP` aims at matching the user profile with respect to all three categories. To thoroughly explore the algorithmic performance, the weights Λ will be incrementally tested from 0 to 1 ($p = 0.02$, to 0.98 for `FA*IR`), at intervals of 0.1. For `FA*IR` we will select the standard value $\alpha = 0.1$. This systematic evaluation will cover a wide range of weight values, allowing for a comprehensive analysis of their impact on the algorithms.
Creating a re-ranking takes much time. A single re-ranking takes up to 2 minutes for `CP`. Therefore, an evaluation of the full dataset is not feasible. To achieve comparable results, we created a subset of 500 users. This has some disadvantages because it might not reflect the actual overall performance for all users and the results deviate from the baseline. Additionally, approaches that observe the whole set of recommendations (CTL, Agg-Div and Gini) will perform worse since a smaller number of items is created, and coverage is harder to achieve. Nonetheless, this approach is sufficient since it allows us to gather insights into the different performances and effects of the algorithms. In the graphs, we annotate the metric values of the baseline algorithms (such as Profiles, Base, Random, and Popularity) based on the evaluation in the previous step, not on the subset.

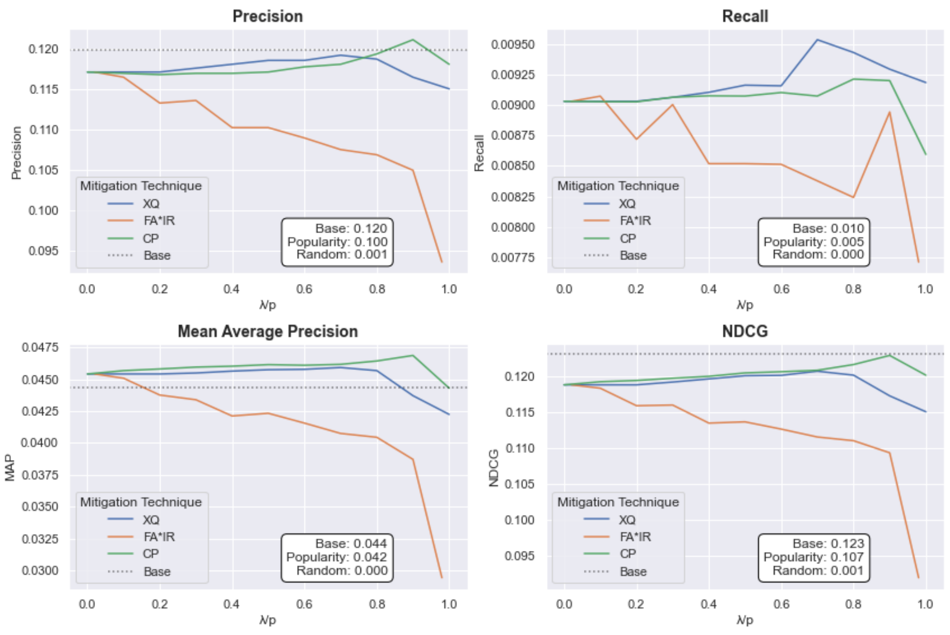


Figure 4.10: Performance across Mitigation Weight for Top 25 Recommendations ($N = 5000$) for Mitigation Algorithms

4.5.3.1 Performance-related metrics

The performance (see figure 4.10) of `XQ` and `CP` is very static over different mitigation weights. There is a slow increase in performance until $\lambda = 0.8$ for `XQ` and $\lambda = 0.9$ for `CP` and a decrease after that. `FA*IR` on the other hand experiences a constant decrease in performance for increasing weight p. Similar to the other two algorithms, an additional more severe decrease in performance is experienced for maximum mitigation ($p = 0.98$).

4.5.3.2 Item-centred Metrics

The descriptive metrics (see figure 4.11) mean and median show a decrease in popularity for increasing mitigation weight. `FA*IR` expresses the biggest decrease, while `CP` shows minimal change until full mitigation. Similarly, `XQ`'s mean does not change much, but a steady decrease in mean and median popularity can be seen. A similar effect can be seen when investigating the exposure-related metrics (see figure 4.12). While `FA*IR` increases the diversity (measured by aggregate diversity and Gini) much with increasing weight, `XQ` achieves the same, but with a slightly less strong effect. `CP` on the other hand decreases the diversity. This can be compared to the coverage of long-tail items where `XQ` increases the exposure slightly, while `FA*IR` increases the long-tail item exposure a lot. With a weight of $p = 0.9$, the average percentage of long-tail items is over 25%, with full mitigation weight, it almost reaches 50%. `XQ` on the other hand, achieves a percentage of about 8% for $\lambda = 0.9$, and slightly above 10% for full mitigation. The percentage and coverage of long-tail items for `CP` are always below the scores of `XQ`. Only with full mitigation weight, it almost achieves the same percentage and coverage of long-tail items.

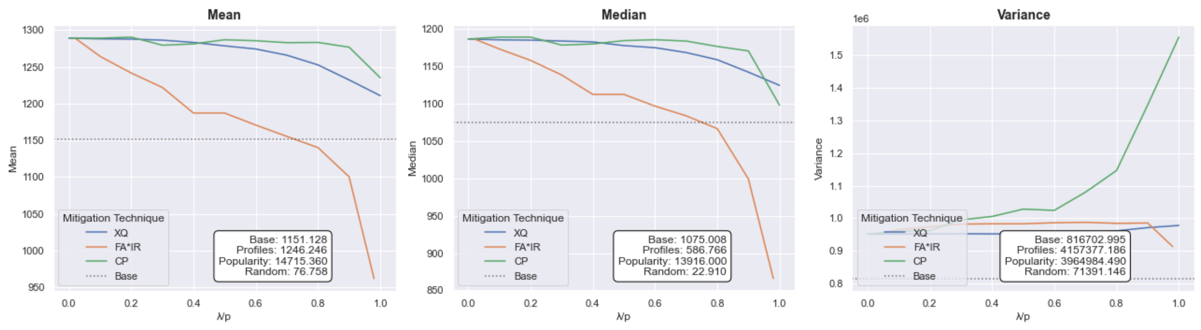


Figure 4.11: Decriptive Metrics across Mitigation Weight for Top 25 Recommendations ($N = 5000$) for Mitigation Algorithms

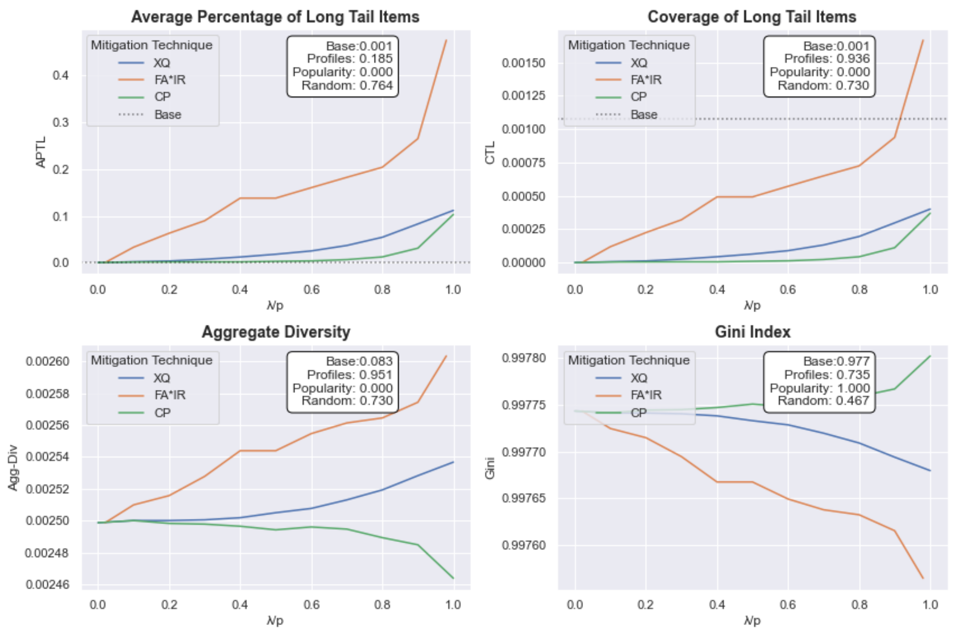


Figure 4.12: Exposure-related Metrics across Mitigation Weight for Top 25 Recommendations ($N = 5000$) for Mitigation Algorithms

4.5.3.3 User-centred Metrics

When investigating the user-centred metrics, the user-centred and item-centred re-rankers both achieve better scores in user popularity deviation by promoting different factors (see figure 4.14). For each user group, `FA*IR` achieves an immediate improvement in user popularity deviation with low weights. The distribution of the categories (see figure 4.16) indicates that `FA*IR` increases the exposure of tail items in a similar ratio as the user profiles. The user popularity deviation is on a similar level for the weights between 0.4 to 0.8. For $p = 0.9$, the user deviation increases slightly and increases a lot for $p = 1.0$. This indicates an overexposure of tail items. The percentage of tail items increases a lot for those values and is higher than in the user profiles. For `XQ` and `CP`, the UPD decreases continuously with increasing weight. The reason for this seems to vary between those two. `XQ` increases the number of tail items. For higher weight, the ratio of tail items fits the user distribution increasingly better. `CP`, on the other hand, increases the number of head items as well. Therefore, the distribution of head, mid and tail items is very similar to the distribution in the user profiles. This explains, why `FA*IR` and `XQ` perform relatively poorly for blockbuster-focused users, and `CP` leads to much improvement for this user group. The item-centred algorithms will only add long-tail items. Blockbuster-focused users do not lack long-tail items, but the base algorithm creates fewer head items than in the user profile. Therefore, `CP` can improve the distribution by promoting head items. Consequently, `FA*IR` and `XQ` can create similar good results in terms of UPD for niche users since their recommendations mainly lack tail items. Nonetheless, it can be seen that `XQ` recommends much fewer tail items than `FA*IR`, therefore avoiding overexposure.
This effect is reflected by the popularity lift metrics (see figure 4.14 and 4.13). `CP` does not change the mean popularity much since it adds head and tail items to the recommendation list to match the user profile. Therefore, the popularity lift stays similar to the initial recommendations. `XQ` reduces the mean popularity by adding tail items, but much less than `FA*IR`.

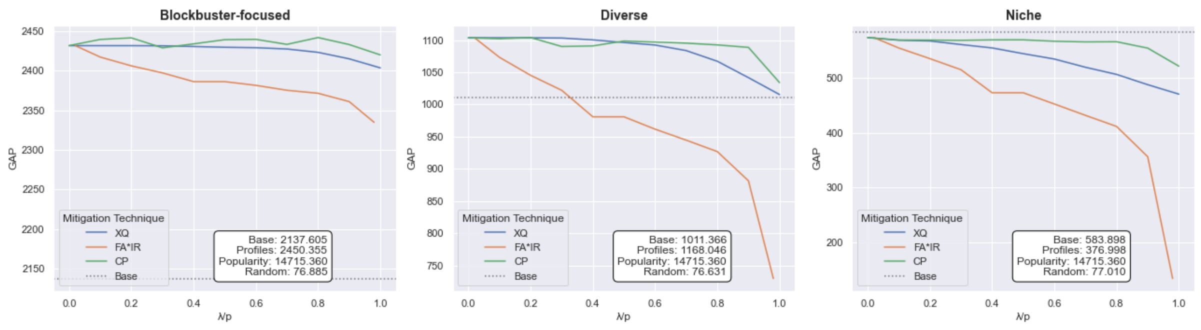


Figure 4.13: Group Average Popularity across Mitigation Weight for Top 25 Recommendations ($N = 5000$) for Mitigation Algorithms

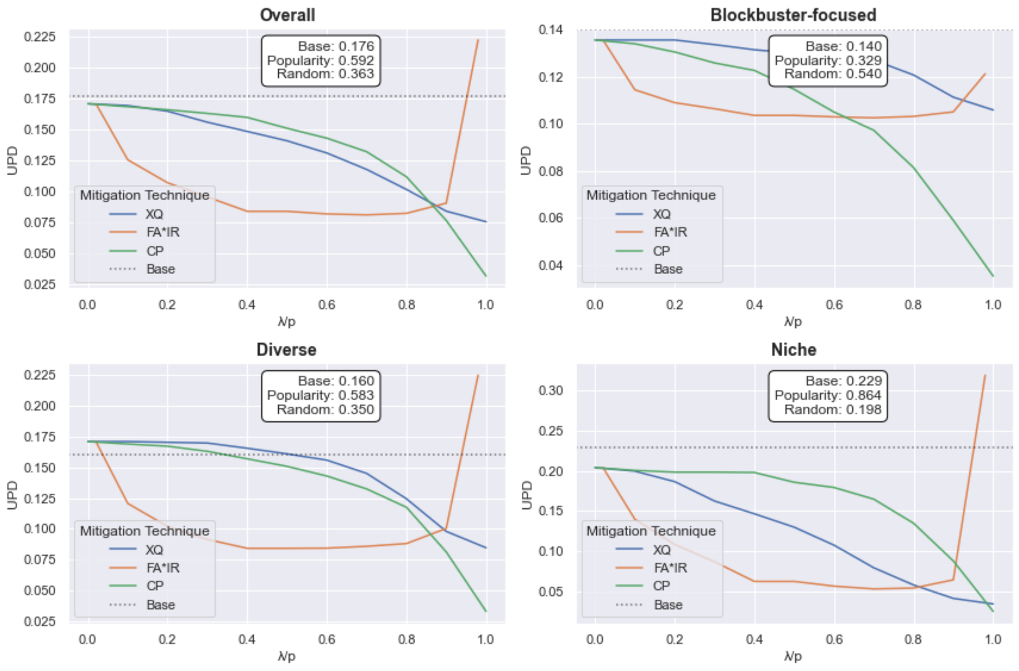


Figure 4.14: User Population Deviation across Mitigation Weight for Top 25 Recommendations ($N = 5000$) for Mitigation Algorithms

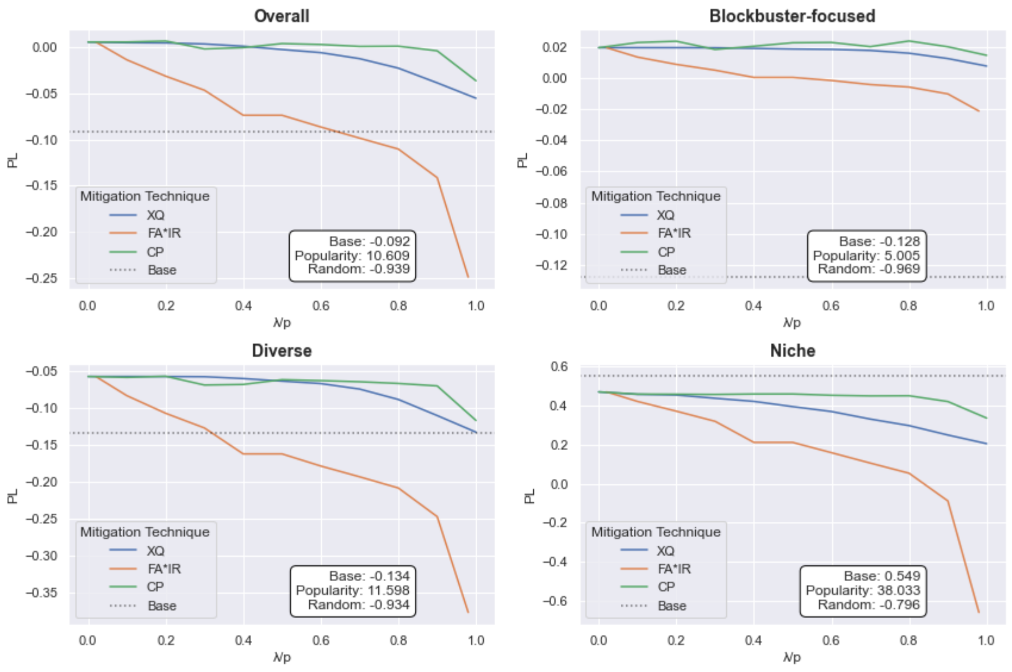
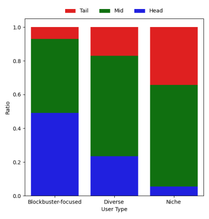
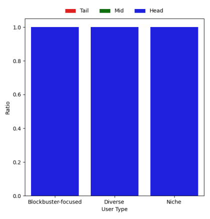


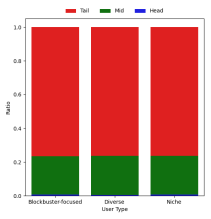
Figure 4.15: Popularity Lift across Mitigation Weight for Top 25 Recommendations ($N = 5000$) for Mitigation Algorithms



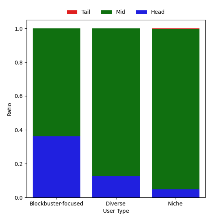
(a) User Profiles



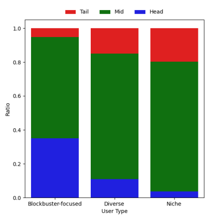
(b) Popularity



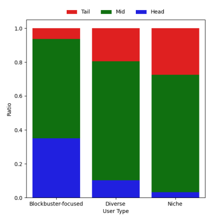
(c) Random



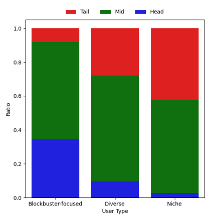
(d) FA*IR
p: 0.02



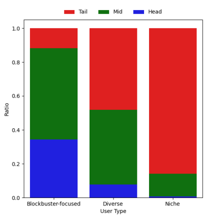
(e) FA*IR
p: 0.40



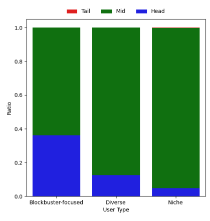
(f) FA*IR
p: 0.70



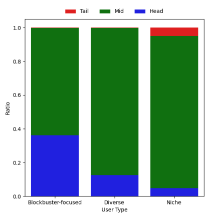
(g) FA*IR
p: 0.90



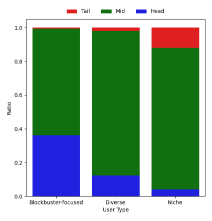
(h) FA*IR
p: 0.98



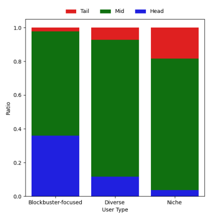
(i) XQ
Λ: 0.00



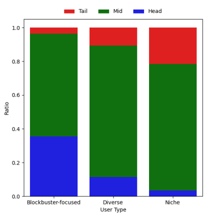
(j) XQ
Λ: 0.40



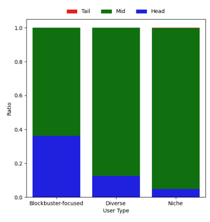
(k) XQ
Λ: 0.70



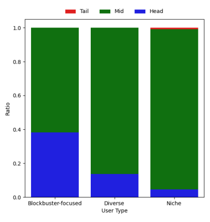
(l) XQ
Λ: 0.90



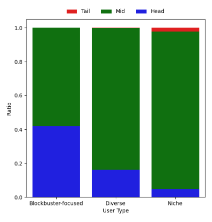
(m) XQ
Λ: 1.00



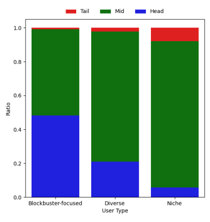
(n) CP
Λ: 0.00



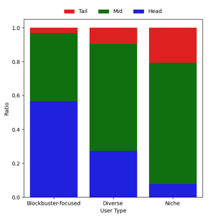
(o) CP
Λ: 0.40



(p) CP
Λ: 0.70



(q) CP
Λ: 0.90



(r) CP
Λ: 1.00

Figure 4.16: Popularity Distributions with Different Mitigation Weights for Top 25 Recommendations ($N = 5000$) for Mitigation Algorithms

4.5.4 Mitigation Discussion

There are various factors that have to be considered when analysing the three different algorithms. On the one hand, we confirm that the algorithms have the potential to reduce the effects of popularity bias. We confirm the presumption that item-centred algorithms improve fairness for artists by improving the exposure of long-tail items. From a user-centred view, this primarily improves the user popularity deviation for niche and diverse users. UPD is heavily improved for blockbuster-focused users by `CP` since it aims at matching the distribution and, therefore, also considers adding head items, which are underrepresented for blockbuster-focused users in the original recommendations. Overall, we conclude that item-centred mitigation algorithms improve item-centred fairness and the user-centred mitigation algorithm improves user-centred fairness.
Considering the performance of the algorithms, almost no or even a positive effect can be seen for `XQ` and `CP`. The performance for `FA*IR` decreases with increasing weight. This is partly in line with related studies. In [57], the precision is reduced by each mitigation algorithm. A similar effect can be seen in [7] on the LFM-2b subset. While `FA*IR` performs the worst in terms of precision, `CP` also experiences a decrease in precision with increasing weights. We explain this contradictory effect with the initial structure of the base recommendations. A majority of the recommendations in [7] are head items. Therefore, `CP` adds more mid and tail items to match the user profiles while reducing the number of head items. Our base algorithm recommends more mid items and has to add head items to match the user profiles. Head items have a high chance of being in the user profile since those are a few items with many interactions. Our approach is able to improve the performance by adding those items; in [7], those items are removed.
When comparing the item-centred metrics, it becomes clear that `FA*IR` adds a much higher number of tail items in the re-ranking. This might be partly due to the fact that `XQ` is classified as an item-centred technique but has an additional personalisation factor, which tracks the match between the user profile and the recommendations. This seems to avoid further promotion of long-tail items beyond the user ratio match. Nonetheless, it does not create more head items than existing in the original list, because only long-tail items get more exposure. `FA*IR` on the other hand does not consider personalisation but focuses mainly on item exposure. Therefore, `FA*IR` seems to create fairer recommendations from an item-centred view. Those recommendations can deviate more from the user profile through overexposure of tail items. This is avoided by `XQ`. Interestingly, the number of tail items varies between the user groups. Blockbuster users receive much fewer tail items even after re-ranking by an item-centred algorithm. This is due to the structure of the recommendations. The personalised recommendations show way less exposure to tail items for users who mainly consume head items. Therefore, they are also included less in the final recommendations. Klimashevskaia et al. [57] propose that item-centred mitigation strategies can be more successful than user-centred strategies if the goal is item fairness. Abdollahpouri et al.'s [7] observations on a music-related dataset do not confirm this theory. In their analysis, `CP` outperforms all item-centred strategies on the exposure-related metrics. Similar to our observations of the differences in the algorithms' performance, we argue that this contradiction occurs due to the initial structure of the recommendations. If there are already many head items in the initial recommendations, `CP` can improve the recommendations clearly by matching the user profile. Item-centred algorithms can achieve exposure of tail items beyond their ratio in the user profile.
`CP`, on the other hand, clearly shows improved user popularity deviation for high weights, which is to be expected as the algorithm directly aims at improving this metric. Since the base algorithm creates fewer head items than in the user profiles, head and tail items seem to be promoted accordingly to the user profile. While `FA*IR` might receive equally good recommendations in terms of UPD for niche users because the algorithm promotes underrepresented tail items for them, `CP` performs much better in terms of UPD for blockbuster-focused users by additionally promoting head items. `CP` avoids miscalibration for different user groups more effectively and consistently. This observation is confirmed by Abdollahpouri et al. [7]. `FA*IR` is able to create lower UPD for low weights by initially adding tail items, but for high

weights, CP is able to match the user profiles more accurately.
When observing the weights, it has to be considered that for many metrics, only high weights of the mitigation make a difference. For example, CP improves the percentage of long-tail items almost not at all until $\lambda = 0.7$. FA*IR for example continuously improves the UPD for $p <= 0.9$ but performs way worse for $p = 98$. Most metrics seem to follow a clear trend which gets an additional positive or negative effect for maximum weight. This also relates to the performance metrics. XQ has stable performance for $\lambda <= 0.8$, CP increases the performance for $\lambda <= 0.9$. Higher weights reduce performance. FA*IR always leads to worse performance, probably due to exposure of long-tail items that have a low chance of matching the user's profile due to their low number of interactions in the dataset. For $p <= 0.9$, this trend seems relatively linear. For $p = 0.98$, the performance is reduced strongly. Those results show that full mitigation seems not useful for typical recommendations, and might create worse recommendations. Lower weights seem to influence the mitigation relatively weak, while strong mitigation of around 0.8 to 0.9 can have strong effects on the item and user-centred metrics while having a weak negative to even slightly positive effect on performance.
In conclusion, we confirm Klimaschevskaia et al's [57] presumption that item-centred mitigation strategies are suitable for achieving item fairness and reducing the effect of the popularity bias on a platform level. For achieving user fairness, algorithms that are specifically designed to achieve user fairness tend to outperform item-centred approaches. Nonetheless, this effect seems to be impacted by the initial structure of the recommendations. If the initial recommendations include mainly head items, user-centred algorithms can be able to perform better than item-centred approaches in terms of item fairness with similar levels of accuracy as shown in [7]. Those presumptions have to be validated further and mitigation strategies have to be tested on different algorithms and various datasets with different user and item sizes to fully grasp the effects of the algorithms. Additionally, we observed that the number of initial recommendations can influence the results massively. Nevertheless, the analysis of the current works focusing on mitigating RankALS' recommendation indicates an effect of dataset size and possibly model fit.
Furthermore, it is important to consider the changes that we applied to the algorithms. In an initial analysis, the item-centred algorithms impacted the re-ranking minimally. Only by selecting tail items separately as protected items, we were able to achieve recognizable changes in the re-ranking.

4.5.5 Limitations

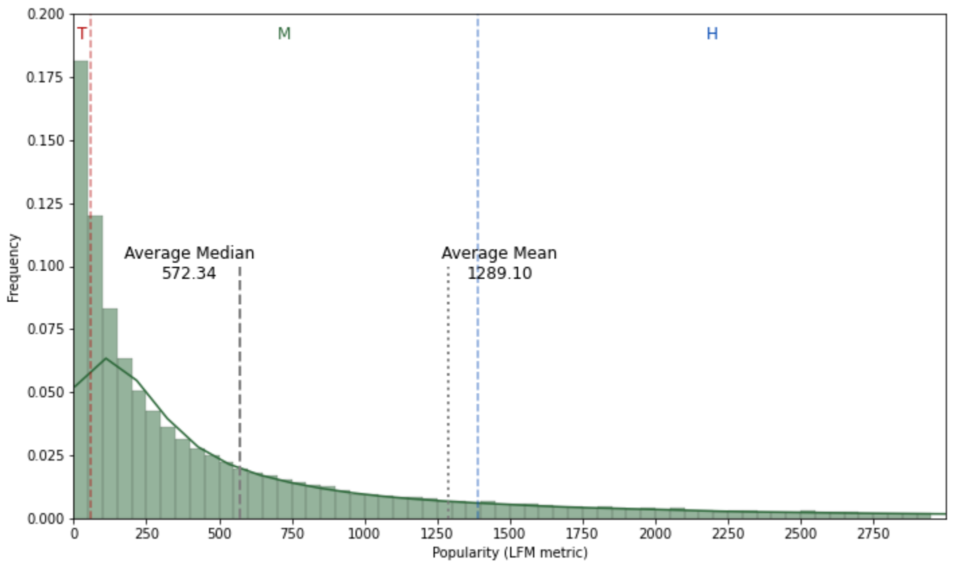
In the previous sections, we already mentioned some changes that we applied to the algorithms and the evaluations that have to be considered when discussing the results. In this section, we will further discuss some limitations and their prospects for future work.
First of all, we can observe some limitations of the used popularity measurement. Popularity is measured as the number of users who interacted with the song in the past. The median value deviates much from the mean popularity in the user profiles. This indicates that the distribution is very skewed. Highly popular items can have much impact on the mean because they often have very high values. If, for instance, a recommendation list consists of 4 items with a popularity of 10 and one item with a popularity of 5000, the mean popularity would still be about 1000 although the majority of the recommendations consist of tail items. This is not necessarily a problem for the descriptive metrics since we can also observe the median, but other metrics rely on the mean values and might, therefore, present unreliable, skewed results. For example, if the user has some highly popular songs in their profile, the mean popularity will be skewed upwards. If the algorithm predicts no head items, but many mid items, there could still be some kind of popularity bias. This could be because most songs have a higher popularity than those in the user profile, but the user profile has a skewed mean value because of some highly popular songs. We presume that this is also responsible for the negative popularity lift found in the base recommendations.

If we compare the median values, an increased popularity can be observed. The median is not skewed by highly popular items. It reflects the middle value. Half of the items are more popular than this one, and the other half is less popular. We have shown that the algorithm does not recommend many head items. Head items in the user profile skew the mean popularity and can lead to an inaccurate representation of the popularity lift. However, an increased median shows that more items have high popularity in the recommendations in comparison to the user profiles, indicating a popularity bias. This indicates that the popularity lift might not be the most suitable metric for measuring popularity bias.
We highlight this effect in figure 4.17. The figures show histograms of the popularity scores of the songs in the recommendations and user profiles. For computing those, the subset created for the mitigation evaluation was used (see section 4.5.3). The graph shows that the distribution of the user profile songs is very skewed to the left (tail). Half of the items in the user profiles have a popularity score of less than 572.34 as indicated by the median. 20% of the items are tail items and have 56 or fewer interactions. Nonetheless, the mean popularity is much higher (1289.10), which is much closer to the boundary for head items (1390) although only 20% of the items are on the right side of this boundary. The profiles are much less skewed and the median is much closer to the mean. Since fewer tail items and items on the lower spectrum of the mid category are recommended in comparison to the user profiles, we would argue for a popularity bias. However, the mean metric and popularity bias do not reflect this bias.
This was also indicated by Kowald, Schedl and Lex [64] who argue that popularity bias is not a robust metric and could not be suitable for large-scale datasets, e.g., in the music domain. To avoid this, future work should either standardize the popularity measurement or analyse the distribution of song popularity in more detail (e.g., percentiles), to fully grasp the effects of the algorithms.

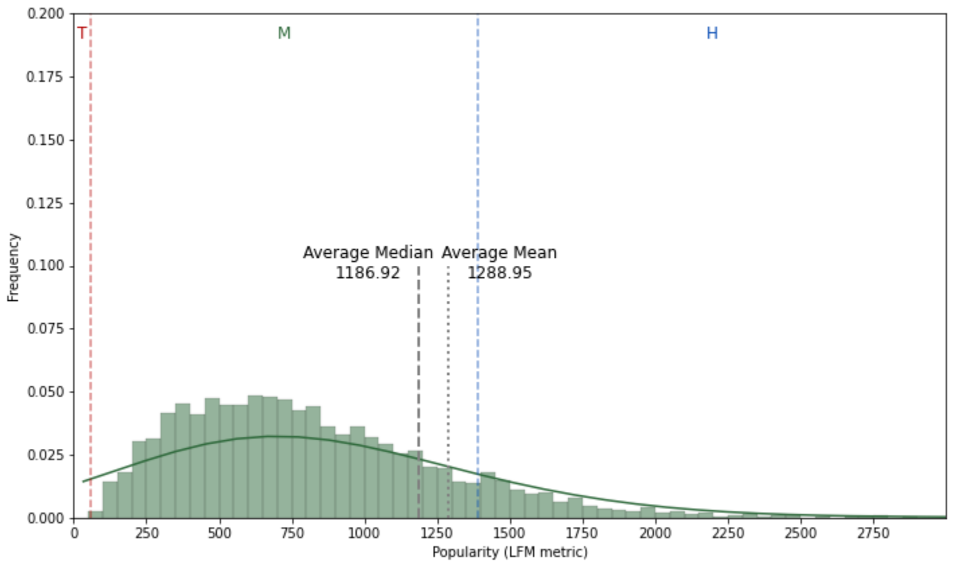
For the analysis of the results and computing the metrics, we created always 25 recommendations. This is a straightforward process and appropriate for gaining insights into the differences between algorithms. Nevertheless, this approach might not be optimal for an entirely thorough evaluation. For example, metrics like coverage and aggregate diversity rely heavily on the number of recommended items and their results have to be always observed under this assumption. Furthermore, performance metrics also rely on the number of predicted items. If a user has, for instance, 100 songs in their profile, and we create 50 recommendations, it is much more probable to predict many fitting ones than when we only create 25. This could be avoided by, for example, creating recommendation lists that match the length of the number of songs in the test set for each user. A similar approach was used by Lesota et al. [69] for analyzing the popularity distributions and changes between the user profiles and the recommendation lists. They chose to create as many recommendations as the user has in their user profile to create fully comparable lists in terms of popularity distributions. Since we will, in the user study in the next step, always only consider the top-25 recommendations, the approach used by us gives us accurate insights into how the lists will look for users. Nevertheless, for works that focus entirely on the algorithmic evaluation of the algorithms, the approach by Lesota et al. should be considered.
Furthermore, we had limited resources regarding time and computing resources. Hence, we had to make some simplifications regarding fully comparing the models. Some decisions relied on informal observations on smaller subsets. For instance, the selection of 5,000 songs for the initial recommendation lists was made after observing that smaller lists often barely add tail items to the recommendation lists, while larger lists do not add any value to the final lists. In future evaluations, the size of the initial lists should also be validated. Additionally, a rather small subset was used for evaluating the mitigation strategies. This is suitable for comparing the strategies, but a more thorough analysis is also needed in this case. Lastly, we reported the presumption that a larger model would improve the performance of the model as well as establish a stronger popularity bias. This presumption has to be validated by future work. Nonetheless, it has to be considered that this model would be even larger and probably not easy to handle on servers anymore in appropriate computing time.

Additionally, this work leaves much room for future analysis. Although the user-centred mitigation algorithm enables improved calibration in terms of popularity, this does not imply content calibration [7]. The content might not be suitable for the user. Future work could consider evaluating this, but it is also possible to achieve genre calibration [107]. Future work could aim at combining those techniques.
Finally, it is worth mentioning that the created system was quite extensive. There are various scripts, functions, libraries and tools used to achieve this evaluation. No clear framework like Librec[17] was used for this evaluation. The `RankALS` model's implementation is quite complex and the creators of the algorithm mention themselves that an implementation without errors is not trivial [108]. The code was developed with extensive care by comparing the code to the pseudocode in the original paper, as well as implementations in another programming language. The implementation was tested by comparing its predictions to the predictions of an `ImplicitALS` model provided by the `implicit` library[18]. Various metrics showed similar results. The creators of the algorithm validated their model by also implementing a "naive derivative evaluation and comparing the two variants" [108]. To further validate the results, this should be done in future work, additionally to code review by other developers.

[17]https://guoguibing.github.io/librec/index.html
[18]https://github.com/benfred/implicit



(a) Profiles



(b) Recommendations

Figure 4.17: Histograms depicting the distribution of songs in the user profile and base recommendations based on the used popularity metric (using the subset of the mitigation analysis as explained in section 4.5.3)
The x-axis is cut off at a score of 3000 for visibility reasons. The maximum popularity score in the user profiles and in the recommendations is 21,044.

Chapter 5

User Study

The user study aims at answering RQ2 and RQ3 directly. While RQ2 ("To what degree does a mitigated popularity bias, created by different recommendation strategies, have an impact on the user perception and satisfaction with music recommendations in an exploration setting?") can be answered through questionnaires after presenting the users to recommendations, RQ3 ("Does a mitigated popularity bias in an exploration setting lead to increased motivation for exploring long-tail music items and indicate potential changes in user behaviour towards fairer music consumption?") remains more difficult to answer. Since we cannot measure the future behaviour of the users directly, we have to make predictions about it. For this, we will investigate the behavioural intentions of the user and make use of the "theory of planned behaviour"[40].

5.1 Properties

For generating the study design, we will conclude the important findings of our research.
Firstly, we conclude that popularity bias exists in the music industry. There are many very popular songs which receive much attention while there is a long tail of songs that are rarely listened to by users. This effect is further increased by the popularity lift. Recommenders tend to recommend mainly highly popular songs, even though they might be less relevant. This leads to unfair treatment of the artists if their songs are less recommended simply because they are less popular. Users are negatively affected by the popularity bias if recommendations that are less fitting are preferred because they are more popular. Users are treated unfairly if there are some user groups who are more strongly affected by the popularity bias than others and, therefore, receive worse recommendations.
There are attempts to solve those fairness issues by mitigating them algorithmically. For artists, this is attempted through item-centred strategies that aim at equaling the distributions between popular and less popular items or by promoting long-tail items. For users, it is done through user-centred strategies which try to match the popularity distributions of the recommendations to the distributions of the user's listening history. Evaluations of those algorithms are mainly done algorithmically, but insights into the users' perceptions and experience can only be achieved by conducting a user study. In real settings, the impact of different levels of popularity is not well studied, and might even not have an impact on the user's perception of the music [39].
We want to investigate the potential of reduced popularity bias for user fairness by creating more equal chances for artists from each popularity level. We created algorithms and found that fairer recommendations for artists come with the detriment of lower performance. In this user study, we explore whether seemingly less optimal recommendations can still create some worth for the users. Are users satisfied if recommendations are less popular than in their listening history? If yes, this will create chances for accomplishing fairer music consumption. Additionally, we want to investigate the potential for fostering fairer listening behaviour in the user's organic consumption.
The potential and effectiveness of fairer recommendations are not clear and dependent heavily on the

situation and setting. We explained that somewhat popular recommendations can be useful and the effectiveness depends on the user and the type of session. Since popularity can also create a feeling of familiarity, popularity can also create positive effects. Familiarity can increase the attractiveness of a playlist, but we presume that less popularity, achieved by mitigation algorithms, is especially useful for lean-in exploration settings where the user actively searches for novel items. In this case, less popularity might have a positive influence similar to high diversity, which improves the attractiveness of playlists if they enrich the user's taste (discovery) [38, 43]. Therefore, we create a fixed setting aiming to reduce situational factors and to create a specific interaction goal.
We make use of mitigation strategies explained and developed in section 3 to create a personalised recommender system. Additionally, we consider those properties to develop a user study aimed at investigating the effects of mitigation strategies in an exploration setting on the users' perception of the system, their satisfaction and its impact on behavioural intentions.

5.1.1 Related studies

There are few studies directly investigating the effects of popularity bias on the users' perception of a music recommender system. In this section, we will discuss related studies and frameworks and discuss design decisions to create the study design for this work.
Knijnenburg et al. [60] developed a well-known framework for evaluating the user experience of recommender systems. The model gains thorough insights into the users' perceptions by creating aspects that influence each other and are related. Through this framework, they analyse and explain how a system influences the users' perceptions. The objective system aspects (e.g., the used algorithm) influence the subjective constructs, which influence the user experience and the user's behaviour. Those effects are mediated by situational and personal characteristics. Conducted studies are mainly done by presenting items (like videos) over a specific time frame, showing them lists of items, or giving the users the task of finding a fitting item. After users were confronted with the recommendation lists, they were asked to fill in questionnaires regarding the perception and usage of the recommendations. The advantage of this framework is that it not only gives insight into how the users perceive their experience but also into mediators that determine why they like or dislike certain systems. An analysis of the questionnaires and the relationships is done by exploratory factor analysis and structural equation modelling. To successfully use an SEM in the analysis of the experiment, the required sample size ranges from 30 to 460 participants [114]. Since a structural equation model is not possible, Knijnenburg et al. [61] propose a pragmatic analysis of the results by analysing correlations and t-tests between the factors. Overall, this framework offers important insights into the conduction of recommender model evaluations. Foremost, we will consider the measurement categories when defining the metrics that we use in this study (see section 5.3.4). A similar Framework was developed by Pu et al. [90], which categorizes factors of user experience similarly. Additionally, it expands the user experience by the construct of behavioural intention. Behavioural intentions represent whether a user decides to use the system again or consume some of the recommended items in the future. Behavioural intentions are directly linked to the attitudes of the user. Based on the ”Theory of Planned Behaviour” [40], a behavioural intention (or use intention) explains the user's intention to use a recommender system again in the future [111]. They play an important role in answering RQ3.
We investigate three studies that investigate the effect of popularity bias in music recommender systems. Graus et al. [43] investigate the effect of active engagement on the perception of popularity bias in music recommendations. Ferwerda et al. [39] and Lesota et al. [68] investigate similarly how the popularity bias influences the perception of popularity and satisfaction with music recommendations. Former create personalised recommendation lists with a target population parameter (high or low popularity) using the Spotify API based on a seed track that was chosen by the user from their top tracks. The latter two

create recommendations with different algorithms which show different levels of popularity bias based on the users' last.fm accounts. All of them recruit participants through crowdsourcing and Graus and Ferwerda investigate mediators (like familiarity) to evaluate their satisfaction with the recommendations. The results of those studies are discussed in section 2.5.4.2.
Studies differ strongly regarding their procedure. For presenting music recommendations, typically two approaches can be seen. Firstly, the recommendation lists can be presented and evaluated **item-based**. That means that each song is listened to and rated separately. The advantage of this approach is that each song can individually be investigated, and effects like outliers can easily be identified. Additionally, within-subjects design can easily be created by creating multiple playlists based on different algorithms and merging the playlists. Since each song is rated individually, the ratings can be observed by relating the ratings to the respective algorithm [14]. If the recommendations are not merged, participants are often asked to fill in another questionnaire about the perception of the playlist they just listened to in general [39, 68, 97]. While this approach shows high potential for analysis and a very clear structure, the task for the user is very limited. Users are typically only asked to rate specific songs. It does not require much engagement and does not reflect an exploratory setting.
Another typical approach is **playlist-based** evaluation. The playlist-based study design does reflect a natural interaction with a recommender system more clearly. Ferwerda et al. [38] provide users with playlists and metadata about the songs, while also linking example songs about the artists. Users were able to freely investigate the playlists and answer the questionnaires about them. In Graus et al. [43], entire playlists are presented to the users, and the users can hover over the song titles to listen to song previews. They were asked to judge each playlist after investigating it. Jin et al. [54] create an adaptation of Knijnenburg et al.'s framework [60] to evaluate a music recommender. While they investigate different levels of user control, this approach can be applied to the evaluation of different algorithms. Users interact with the system based on a specific experimental task. They ask the users to find five good songs that best match the presented scenario and the user's music preference. While also rating each song individually, the main evaluation is done after interacting with the system through post-study questionnaires. This approach has multiple advantages over the item-based approach. Firstly, it enables the user to have more interaction with the system. The researchers can give the user a clear setting or scenario and the user has a clear task. Additionally, since the users have more interaction freedom, the interaction data (like interaction time, clicks etc.) can be saved and analysed. Nonetheless, this approach has other disadvantages like less control and structure of the experiment.
In conclusion, we highlighted the importance of an exploratory setting of the user study. This can be best represented by a playlist-based study design. The user can be confronted with a clear scenario and task that clarifies the exploratory setting, and the user can freely explore the presented playlists. Therefore, we will create a playlist-based user study.
Additionally, observations were made regarding conducting user studies of music recommender systems. This is because songs are rarely fully presented to the user, but often in the form of a short preview (e.g., [20, 43, 97]).

The study conducted in this work aims at a different approach than the discussed studies by Graus et al. [43], Ferwerda et al. [39] and Lesota et al. [68] by directly applying a mitigation algorithm and testing its effects on the users' perception. Graus et al. investigate the effect of recommendations given two different popularity levels (25 and 75, based on Spotify's popularity measure). They did no clear evaluation or analysis of the effects of the recommendation algorithm that created those recommendations. Additionally, their main research focus was the effect of active engagement on popularity bias perception. Ferwerda et al. and Lesota et al. on the other hand focused on three different recommendation algorithms. Although they show different levels of popularity bias, they do not reduce the effect of the popularity bias. An analysis by Lesota et al. [69] shows that all of them increase the degree of popular items (although

ItemKNN only slightly with $\%\Delta Mean = 9.6$). None of them reduced the popularity level. Additionally, both studies do not offer the user the possibility to listen to the songs. Presumably, this influences the experience with the recommendations. Especially unknown songs can not properly be rated. We will enable listening to those recommendations.
Therefore, we want to evaluate how a mitigation strategy influences the users' perception in comparison to an unmitigated algorithm. We do not aim to investigate different levels of popularity bias but to mitigate the bias for user and artist fairness. Therefore, we will present the users with recommendations that are either not manipulated or manipulated by one of the previously discussed re-ranking algorithms. To compare the effects of mitigation biases, we will test the base algorithm as well as an item-centred and a user-centred algorithm. Additionally, we want to investigate a broad range of system aspects, mediators and potential attitudes of the users.

5.2 Algorithm Selection

Looking back at the results of the evaluation of the algorithms, we select `CP` as the user-centred mitigation algorithm. We showed its performance to improve personalised recommendations by matching the user profile. Additionally, it can achieve improved performance.
For the item-centred method, we have to choose between `XQ` and `FA*IR`. While `XQ` performs very well, it is not able to promote long-tail items to the same degree as `FA*IR` does. Since we want to identify the effects of strong mitigation on user perception and satisfaction, `XQ` might not create sufficiently unique recommendation lists but lists that are close to the original lists created by the base algorithm. `FA*IR` on the other hand can create re-ranking lists that differ much from the original lists. Hence, `FA*IR` is selected as the item-centred algorithm.
In terms of the weighting of the mitigation, it is important to consider that `FA*IR` as well as `CP` do not disregard the scoring of the base algorithm for high weighting. `FA*IR` creates two lists and favours the best of the protected items if the list requires more of those. Otherwise, it takes the best available item. `CP` weights the score of the base recommendation with $1 - \lambda$, and the fairness score is given by how much the added item would improve the user popularity deviation. This second part of the score is identical for each item from the same category. Therefore, even a mitigation weight of $\lambda = 1$ would lead to the selection of the best item from the initial list from the group which receives the highest fairness score. Therefore, mitigation of $\lambda = 1$ necessarily leads to matching distributions if enough items from each category are available. A lower weight might disregard the next best item from a category because the base recommendation score exceeds the fairness score.
While a very high weight might seem suboptimal because the influence of the initial scores is minimized, this achieves maximum re-ranking by promoting either long-tail items as much as possible, or by ensuring matching user popularities. This approach might lead to less optimal performance, but it achieves an optimal comparison between recommendations influenced by different goals: The base algorithm aims for accuracy but is impacted by the popularity bias; the item-centred algorithm focuses on the promotion of long-tail items with the detriment in performance; the user-centred algorithm aims for reduction of miscalibration for various user groups and can achieve positive results in terms of performance. For observing the user's perception of recommendations that are influenced by those varying degrees of fairness, a high level of mitigation can create insights into the impact on the users. While the lower performance of the item-centred algorithm indicates suboptimal results for the users, the investigation of the attitudes of the user can answer the question of whether the promotion of long-tail items adds some other value for the user. Therefore, we select a weight of $\lambda = 0.99$ for `CP` and $\alpha = 0.98$ for `FA*IR`.

5.3 Study Design

The following study design is designed based on the findings from the evaluation of the algorithms discussed in section 4.5. We follow Knijnenburg et al.'s guidelines for creating the study design [59]. Users can experiment from their homes on their computers. Users should interact and listen to playlists. Afterwards, they are asked to fill in questionnaires to gain insights into their perception and satisfaction with the recommendation lists.
The study will be concluded by interviewing five participants to gather more qualitative feedback regarding the users' perceptions and experiences, as well as their criteria for filling in the questionnaires. The qualitative analysis will expand the understanding of the quantitative results.

5.3.1 Participants

Since the recommender tool creates personalised recommendations, the participants are required to have a Spotify account which they used regularly for at least 3 months before the start of the study to ensure that a proper user profile can be created. Participants are gathered through the researcher's student and friend network.

5.3.2 Conditions

We create a within-subject design. The independent variables are either only the base algorithm or the base recommendation with an additional mitigation algorithm. For simplicity reasons, we call them recommendation algorithms. Each participant interacts with each of the following recommendation algorithms.

- **Base** (`RankALS`)
  The base `RankALS` represents the general recommendation algorithm, which has shown a medium degree of popularity bias and popularity lift.
- **Item-centred mitigation** (`FA*IR`)
  `FA*IR` is the item-centred mitigation strategy that aims for artist fairness. Its goal is higher exposure to long-tail items.
- **User-centred mitigation** (`CP`)
  `CP` does not attempt to reduce the popularity bias platform-wide, but it aims to create optimal recommendations for individual users. Therefore, it tries to match the user's preferences for popularity distribution by reducing miscalibration.

The order of the recommendation algorithms is randomized to avoid fatigue and learning effects. The participants receive recommendation playlists of the top 25 items based on those algorithms.

5.3.3 Procedure

5.3.3.1 Distribution

For gathering participants, an initial questionnaire will be sent to them informing them about the goal of the study, the approximate duration of the study, privacy information, as well as further requirements. Additionally, they are asked to provide the E-Mail address of their Spotify account as well as a broad time frame in which they want to participate in the study. This is done to grant the users access to the tool.
Participants who agreed to take part in the study are sent the link to the study tool. They are free to participate in the experiment by themselves but are asked to conduct the study in a quiet environment without distractors around. Additionally, they should open the tool in a browser on their computer or laptop since the application was not developed for mobile interfaces.

5.3.3.2 Set-up

After opening the study tool users can log in with their Spotify accounts. While the system is loading, each participant is informed about the goal of the study, data handling and privacy, as well as their right to withdraw.
While the recommendations are generated, the participants fill in the pre-questionnaire (see table B.1) asking about demographics and personal characteristics like musical sophistication and active engagement (see section 5.3.4). After this, they are presented with the three study trials.

5.3.3.3 Study Trials

Each study trial begins with an instruction screen clarifying the setting of a lean-in exploration scenario. The experimental task is to find 5 items in the list that match their personal music taste. Users are asked to explore the items and their objective is to "discover 5 songs that genuinely resonate with you and that you would like to add to your playlist".
Afterwards, the participants are presented with the recommendation playlist consisting of 25 songs. Many studies use playlists of about 10 items (e.g., [39, 68]). Since the participants were asked to explore the playlists freely and to enable more choices, we extended the length to 25 items. Other studies rely on the anonymization of song metadata. We make use of the Spotify API. To comply with Spotify's guidelines, metadata like the artist, the song title, and the album cover are provided. The participants can freely interact with the system and listen to previews of each song. The button for finishing the task and filling in the questionnaire only activates after finishing the experimental task. After accomplishing their task the participants are supposed to fill in various post-study questionnaires (see table B.2) inquiring about their perception of the recommendations and their attitude towards the playlist (see section 5.3.4). Attention checks are incorporated into the questionnaires. After the last questionnaires, the users are asked to state their general preferences and explain why they preferred their selected playlists.

5.3.3.4 Profile Validation

For creating appropriate user profiles, the implementation has to rely on some workarounds. Those will be explained in section 6. To validate that the used user profiles are appropriate, users are presented with a playlist consisting of a random selection of 50 songs from their user profile. They are asked to briefly observe the playlist, and assess how familiar they are with the presented selection and whether it fits their listening preferences. Afterwards, the final questionnaire asks the participants for their perceptions.

5.3.3.5 Wrap-up

After accomplishing the experiment, users are asked to fill in the final questionnaire in which they are asked to accomplish the profile validation. Additionally, users can provide free text comments. Furthermore, they are thanked for their participation, and they can save the playlists to their Spotify account. Finally, participants are asked whether they would be willing to participate in a short interview to reflect on their experience.

5.3.4 Metrics

During the study, the users fill in multiple questionnaires and create data through their interaction. Based on [60] and [90], we created a framework that includes the important metrics. It explains how the system factors (conditions) influence the users' perception and user experience. Figure 5.1 gives an overview of the metrics, as well as possible interactions between components. The *Recommendation Algorithm* is the independent variable. Other OSA, like *Popularity Lift* or *User Popularity Deviation* depends on the algorithm and the user profile. The SSA are mediators for the attitudes and behaviour of the user. Finally, we conclude that the attitudes influence the "Behavioural Intentions" of the users. The questionnaire questions and interaction data are saved in a log file. The full questionnaires can be seen in Appendix B. We make use of existing questionnaires and measure all of them on a 7-point Likert/Agreement scale for consistency reasons. The scales range from "Completely disagree" to "Completely agree". The metrics are gathered from Knijnenburg's framework [60] and other related studies discussed in section 5.1.1.

Objective System Aspects (OSA)

The OSA relate to the condition or recommendation/mitigation algorithm. Nevertheless, we acknowledge that there might be individual differences in the average popularity or the popularity lift. Those factors are influenced by the users' listening behaviour and the algorithm. These can also be analysed and create important insights.

- **Recommendation Algorithm**
  The recommendation algorithm is the condition of the current trial and determines the personalised recommendations and the popularity of those.
- **Average Popularity**
  This metric relates to the popularity of the recommendation list measured by descriptive metrics.
- **Popularity lift**
  Popularity lift (or ΔGAP) is defined as the difference between the average popularity of the consumption history and the recommendation list. In section 5.3.4, it is defined for user groups, but it can be equally measured for each user. This determines whether the popularity level increased or decreased on average.

- **Popularity distribution match (JSD)**
  The Jensen-Shannon divergence (JSD) defines how well the popularity distribution in the consumption history matches the distribution in the recommendation list. It is a user-centric metric and shows the degree of deviation or miscalibration. Lesota et al. [68] have shown a correlation between the perceived domination of popular items and the Jensen-Shannon distance. Nevertheless, they did not investigate the direction of miscalibration, which is investigated through ΔGAP.
- **Popularity category ratios:** The head ratio, mid ratio and tail ratio determine the distribution of the popularity categories in the recommendation lists.

Situational Characteristics (SC)

Situational characteristics influence the user experience. We try to minimize the effects of those. There might be other factors like the user's mood, the location or the weather (cf. [54]). For this study, we do not investigate those.

- **Situation/Setting**
  This metric remains unchanged but influences the perception of the user. We aim to create a lean-in exploration scenario for the users.

Personal Characteristics (PC)

Personal characteristics like domain knowledge or engagement influence how users engage with a system and are, therefore, observed by us.

- **Demographics**
  Demographics like age and gender are measured.
- **User Profile**
  By using Spotify data, we can analyse the user's profiles. We can categorize the users into different categories (see section 3.1.1) and analyse their typical interest in popularity. Additionally, we can investigate the average popularity of songs in their user profile and popularity distribution, measured by the ratios of the popularity categories.
- **Musical Sophistication**
  Since musical sophistication can influence how users interact and perceive music recommenders [55], the "General Musical Sophistication" sub-scale of Goldsmiths Musical Sophistication Index (Gold-MSI) [81] is used to control for this effect (cf. [54])
- **Musical Engagement**
  Graus et al. [43] have shown the effects of musical engagement on the perception of popularity bias in music recommendations. Therefore, we investigate this factor. Although it is partly already measured by the "General Musical Sophistication" sub-scale, we additionally add the other questions to separately investigate this factor. The "Active Engagement" sub-scale of the Gold-MSI [81] is used to measure this effect.

Subjective System Aspects (SSA)

Subjective system aspects link the objective properties to the user experience. While this link is typically weak, SSA can explain their relationship by working as mediators. In the framework [60], OSA influence SSA, and SSA influence EXP. For familiarity, discovery and perceived popularity, we make use of the questionnaires created by [43], which allow ratings of the entire playlist.

- **Perceived popularity**
  Perceived popularity explains how popular the recommendations were perceived by the users [43].
- **Discovery**
  Discovery explains to which extent playlists allowed participants to discover new music and refine their musical taste [43].
- **Perceived fairness** While mentioning fairness directly might introduce subjectivity biases because fairness is a subjective matter, the degree to which users perceived playlists as fair can influence their experience [39]. Therefore, we measure it indirectly by inquiring how balanced playlists were perceived. We adapt Ferwerda's metric.
- **Familiarity**
  We discussed that popularity can create a feeling of familiarity. Therefore, we observe how familiar the songs are to the users [43].
- **Perceived Recommendation Quality**
  In [60], the authors define perceived recommendation quality as a metric that influences satisfaction with the recommendations. We adapted their questionnaire.

Experience (EXP)

The user experience determines the evaluation of the user of the system.

- **Recommendation satisfaction**
  Satisfaction or liking is the most common measure for evaluating recommender systems. It determines how satisfied users are with the provided recommendation playlist. We use the music-specific questionnaire that was used by Graus et al. and Ferwerda et al. [39, 43].
- **Choice satisfaction**
  Similarly to *recommendation satisfaction*, choice satisfaction asks for satisfaction with their choices. This questionnaire was adapted from [60].
- **Effectiveness**
  Effectiveness measures how effective the system was in providing fitting recommendations. This questionnaire was also adapted from [60].
- **General Preference**
  To directly investigate whether the participants preferred recommendation playlists over each other, participants are asked to state their general preference by ranking the three playlists after investigating all three.

Interaction (INT)

Interaction data of the user shows their behaviour and can be linked to the user's behaviour.

- **Time spent**
  The user has much freedom when interacting with the system. Therefore, they can vary in the time they spend with recommendation lists. This can give insights into the engagement of the user. This interaction data is saved in the log file. It saves how long the users interacted with the playlists, how long they listened to each song, and in which order they listened to the songs.
- **Selected songs**
  The selected songs can additionally be analysed and create insights. Factors that can be analysed in this regard are, for instance, the position of the selected items or the popularity of the songs.

Behavioral Intentions (BI)

Behavioural intentions express whether the user has an interest in engaging with more similar items. This can indicate that recommendation lists influence the organic listening behaviour of the user.

- **Use intention**
  We define use intention as the users' intention to use the system again to receive recommendations. It shows whether the users show interest in interacting more frequently with the system and the provided recommendations. We adopt "use intention" by Pu et al. [90] to create a measurement for the intention to use the music recommender system again in the future.
- **Choice listening intention**
  Pu et al. [90] define "purchase intention" as the intention of the user to buy the recommended items. We adapt this question to inquire the user whether they would add the chosen songs to their playlist or whether they would listen to them again
- **Openness to similar recommendations**
  We formulate those questions to gain insight into a "give-me-more" attitude of the user [14]. They are supposed to show the user's interest in receiving and listening to more similar items like those presented in the recommendation playlist. This indicates a general interest in listening to similar songs. We use the item provided by Bogdanov [14] as a basis to create similar questionnaires.

5.3.5 Interview

For the interview, 5 participants who responded that they are willing to participate in an interview are asked to join a 30-minute online call. This aims at gathering more insights into the users' experience when interacting with the system. Therefore, we conduct a semi-structured interview inquiring about their perception of the recommendation lists, their thought processes while picking their songs, and how they defined popularity and fairness when filling in the questionnaires. Therefore, we will be able to explain our quantitative results with qualitative statements from the users.

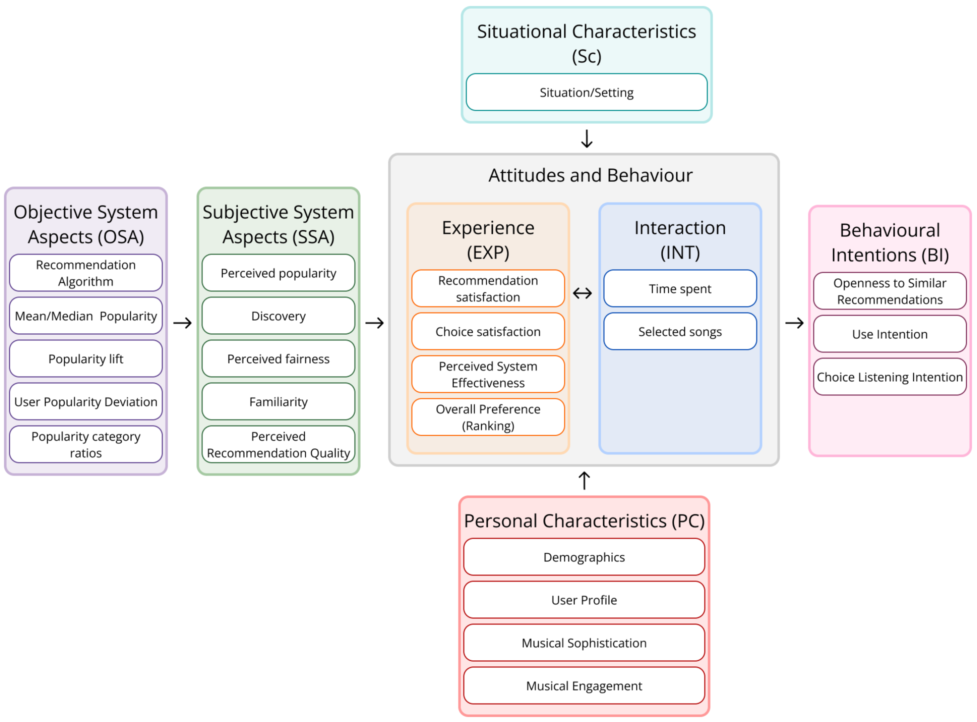

Situational Characteristics (Sc)
Situation/Setting
Attitudes and Behaviour
Objective System Aspects (OSA)
Recommendation Algorithm
Mean/Median Popularity
Popularity lift
User Popularity Deviation
Popularity category ratios
Subjective System Aspects (SSA)
Perceived popularity
Discovery
Perceived fairness
Familiarity
Perceived Recommendation Quality
Experience (EXP)
Recommendation satisfaction
Choice satisfaction
Perceived System Effectiveness
Overall Preference (Ranking)
Interaction (INT)
Time spent
Selected songs
Behavioural Intentions (BI)
Openness to Similar Recommendations
Use Intention
Choice Listening Intention
Personal Characteristics (PC)
Demographics
User Profile
Musical Sophistication
Musical Engagement

Figure 5.1: User-centric evaluation framework of the recommendation algorithms used in the experiment

5.3.5.1 Interview guide questions

We follow the following questions to create a conversation about the user's perception. They will be expanded by follow-up questions.
Those questions aim to create insights into the users' experiences and interactions, their perception of the different playlists, and how they selected the songs. Finally, we inquire how users perceive and judge popularity and fairness in music recommendations.

- Tell us about your experience using the different recommender systems. Did you have any issues using the system? Were there any difficulties?
- Your task was to explore the playlist, can you briefly explain how you did this?
- What did you perceive positively about the recommendation lists? What did you not like about the lists?
- Did you perceive differences between the different lists? How did those differences influence your ratings?
- Based on which criteria did you pick the songs?
- Your setting was an exploration task. Do you perceive that this task influenced your perception?
- When filling in the questionnaires, what factors did you consider for determining popularity?

- What factors do you think contribute to a fair recommendation list for you personally?
- How does popularity contribute to fairness?
- Do you think fairer recommendations influence your satisfaction with a playlist?
- Would you, generally, want recommendations to be fairer? Should recommendations be manipulated to be fairer?

Afterwards, participants receive a short explanation of the methods used in this experiment and are asked whether they have additional thoughts and comments regarding the study.

Chapter 6

Study Tool Development

After validating the recommendation and re-ranking approach, the recommender system is integrated into a recommendation tool that is connected to the Spotify API[1]. The API was used in preceding works to extract musical preferences, create recommendations and play songs [43, 72]. The recommender system that is designed with the support of the API can create recommendations based on the user's listening history and apply the base recommendation algorithm as well as the possibility to re-rank those given the fairness metric.

6.1 Spotify Data Analysis

To validate this approach, thorough data analysis is required. In this section, we will investigate data provided by the Spotify API, the relation to our current data set and recommendation algorithm, and we will evaluate the feasibility and usability of this data [2].
Initially, the `spotify-uris` dataset of LFM-2b is used to match the track ids to the Spotify uris. Afterwards, we retrieve data about these songs from the Spotify API.

6.1.1 The Spotify API

The Spotify API is a tool that allows developers to interact with the Spotify music streaming platform and access its collection of music data. With the Spotify API, we will extract detailed information about songs, create personalised recommendations, gather user data, and play songs directly within our application.
To access the Spotify API and utilize its functionalities, we will utilize the Python library `Spotipy`. `Spotipy` is a Python library that provides an interface for interacting with the Spotify API, simplifying the process of making API requests and handling responses. It enables the integration of Spotify's features into our application.
Authentication tokens are essential for securely accessing the Spotify API. When we interact with the API for a certain user, we obtain an authorization token. This token is acquired through an authentication process where the user grants permission for our application to access their Spotify data. Once we obtain the authentication token, we can include it in API requests to authenticate our application and gain access to the user's data, such as their playlists, saved tracks, and listening history. In development, the Spotify API enables access for 25 users at a time.
In summary, the Spotify API empowers us to integrate Spotify's music streaming capabilities into our study tool. By utilizing `Spotipy` and authenticating through access tokens, we can extract song informa-

[1]https://developer.spotify.com/documentation/web-api
[2]The code and data of this evaluation are available at https://git.science.uu.nl/0982717/mitigatingpopularitybiascode.git

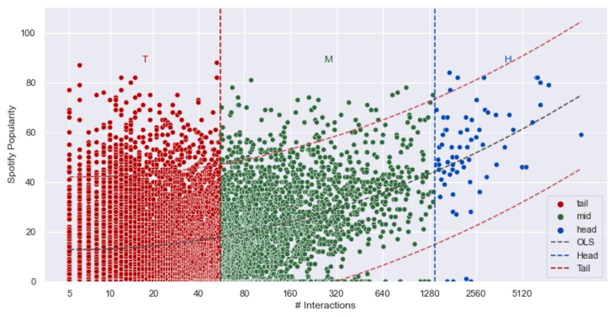


Figure 6.1: Comparison of popularity measure on LFM dataset and Spotify popularity measure

tion, create personalised recommendations, gather user data, and play songs within the tool.

6.1.2 Popularity Measure Comparison

The Spotify API provides a unique popularity metric for each song. We will compare the metrics to make inferences about the measurements of Spotify. The Spotify measurement ranges from 0 to 100. The Spotify API explains that the metric is computed by the algorithm and is based, for the most part, on the total number of plays the track has had and how recent those plays are. On the other hand, the metric used for assessing the popularity of songs in the LFM dataset is the sum of users that interacted at least twice with a track (see section 3.1.2).

To get an overview of how the metrics are related, we created a subset of 10,000 songs and compared the two popularity measures. We randomly selected 10,000 items from the LFM dataset and retrieved the popularity measure from Spotify using *Spotipy*. An initial observation of the data shows some occurrences where the scores deviate much. The Pearson correlation coefficient shows a low correlation between the two measurements $r = 0.2712$.

Figure 6.1 provides valuable insights into the relationship between the number of interactions and the Spotify popularity of items. The scatter plot reveals a positive relationship, indicating that as the number of interactions increases, Spotify popularity tends to increase as well. However, it is important to note that the relationship is not strictly linear.

The scatter plot also highlights the categorization of popularity levels as 'head', 'mid', and 'tail'. While the general trend suggests that items with a higher number of interactions tend to have higher Spotify popularity, there are numerous data points that deviate from this pattern. This indicates that other factors beyond the number of interactions might influence the Spotify popularity of items. For instance, this could be the release date of a song. A song might be classified as a head item because it received many interactions in the past. Because the song is not listened to much anymore nowadays, Spotify might classify its popularity as lower.

This comparison shows the difference between Spotify's approach to measuring popularity and the classical approach used on datasets that track user-item interactions. For instance, differences in popularity over time are not considered in the classical approach that we used for the algorithmic analysis.

Type	Ratio Songs before 04/2020	Total unique songs
Recent	0.2487	197
Short-term	0.2817	142
Medium-term	0.3232	198
Long-term	0.4681	197

Table 6.1: Ratio of songs from before April 2020

6.1.3 Retrieving User Data

Four users are asked for their consent to provide their profile data to conduct initial data analysis. A simple application is generated for this purpose. The app runs on a local host. User login and Spotify authentication are accomplished with `Spotipy`. The users are asked for their consent according to the following scope: `user-read-private user-read-email user-read-recently-played user-top-read`. The scope determines to which items of the user's Spotify account the application has access.
After logging in, the users recently listened to tracks and their top tracks according to the short-term, mid-term and long-term categories were retrieved. According to the Spotify API, long-term top tracks are calculated from several years of data and include all new data as it becomes available, medium-term top tracks include listening events from approximately the last 6 months, and short-term ones of approximately the last 4 weeks. Each category provides a maximum of 50 items. The users are able to investigate the items afterwards.
Those items are stored in a csv file and if those items are also in the filtered LFM dataset, the popularity category and the number of interactions are stored as well.

6.1.4 Initial Data Analysis

After retrieving the data of the four users, we were able to create a set of 751 songs with 487 unique songs. We compare those songs to the filtered LFM dataset. An analysis shows that a majority of the songs are not included in the LFM dataset. Figure 6.2 shows the ratio of items not in the LFM dataset sorted by type. The recent, short-term and mid-term types have an average of 0.88 songs that do not exist in the LFM dataset. Only the long-term category includes more songs that are known with an average of 0.76. This is probably due to the fact that many of the recently consumed songs are relatively new songs. Therefore, those cannot be included in the LFM dataset since it only includes listening events until March 2020. When analysing the release dates of those songs, this presumption can be validated. Overall, only 35.3531% of all retrieved songs are from before April 2020. An in-depth analysis of the types shows that categories that track recent listening behaviour include more recent songs than the long-term type (see table 6.1).

Additionally, we analysed the distribution of popularity categories across the items. Figure 6.3 shows the frequency of each user category per type. It can be seen that head items occur relatively sparsely. The long-term distribution seems to represent actual user profiles most accurately. Typically, 20% of the interactions are with the head, 20% with the tail and 60% with the mid items. This distribution is most

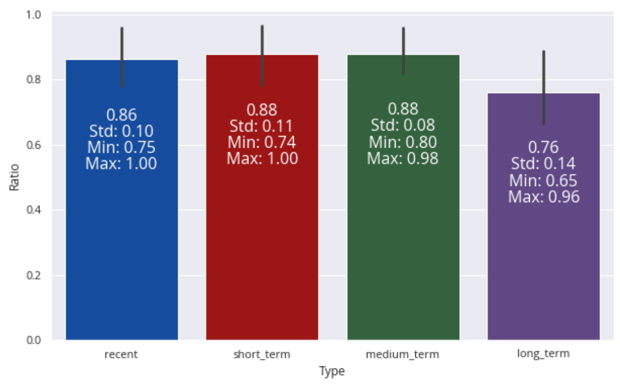


Figure 6.2: Ratio of songs in the Spotify profiles that are not included in the LFM dataset

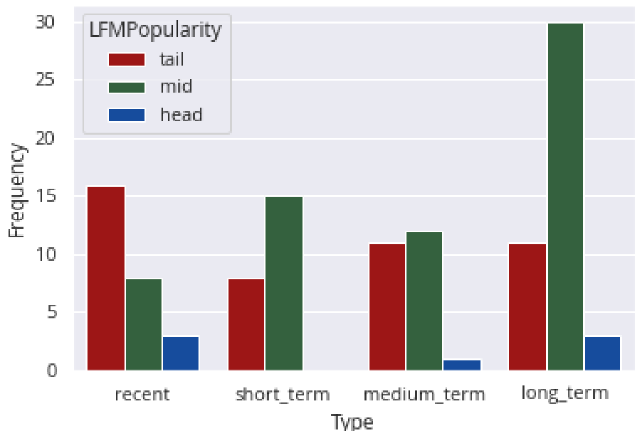


Figure 6.3: Frequency of user categories by type

similar to the long-term type.
Overall, this analysis creates insights into the structure of the data that can be provided by the Spotify API. For creating direct recommendations using the base algorithm created with the LFM dataset, the provided data might be too sparse. Depending on the user, only a small part of the data might be fitting for constructing user profiles to create recommendations. This issue cannot be resolved by retrieving more items since the API only provides 50 items per type. Additionally, only the long-term type seems to ensure available songs at all. It provides the most items overlapping with the LFM dataset and seems to have the most natural distribution of items. Since we assume that we will not be able to create highly accurate recommendations using the base recommender with the provided data by Spotify, we investigate the recommendation algorithm provided by Spotify.

6.1.5 Spotify Recommendations

For creating recommendations, `recommendations` requests are made to the Spotify API using `Spotipy`. The request creates up to 100 recommendations given a maximum of 5 seed tracks. We use this to create personalised recommendations by selecting the top long-term tracks as seed tracks.
Since recommendations are different every time, multiple iterations were done to create a large pool of recommendations. For each user, ten times in a row, 100 recommendations were generated for every 5 songs in the user's top tracks. Overall, for the four participants, this resulted in an average of 9750 recommendations per user with 3311.25 unique songs. From those, on average 2069.5 recommendations consisting of 742 unique tracks were created that were in the LFM data set. Nevertheless, one user reported that they barely used Spotify. Analysis has shown that their recommendations were almost not present in the LFM data set. Only 2% of the recommendations for them were found. Therefore, we removed this user for the following analysis. This results in an average of 2682 recommendations and 949.67 unique songs that were in the recommendations and the LFM data set.
Spotify does not disclose any information on whether its recommendations are ranked. To gain more insights into the structure of the recommendations we compared the rank of the songs in the recommendations with the popularity as described by Spotify's popularity metric. The heatmap in figure 6.4 shows that the popularity seems relatively consistent over the ranks. Generally, there seems to be a tendency towards items in the middle popularity range. Although the popularity does not necessarily give us indications about a ranking in the recommendations, there is no indication for any ranking of the recommendations. When filtering for items that are included in the LFM set, especially items with low popularity are removed (compare figure 6.5).
Furthermore, we compare how similar the distribution of popularity is between items in the user profile and in the recommendations. When investigating all recommendations, the distributions between the user profiles and the recommendations are very similar (see figure 6.6). When only observing the subset that includes songs that appear in the LFM dataset, the presumption that mainly items with low popularity are removed can be confirmed. The ratio of highly popular items becomes higher (see figure 6.7). Overall, the distribution remains similar. The average popularity of the user profiles is 56.81433, the average popularity of the recommendations is 49.9882, and the average popularity of the recommendations in the subset is 57.6435.
Finally, we observe the distribution of popularity categories as defined for the original LFM data set (see figure 6.8). While tail items seem overrepresented in the recommendations in comparison to the user profiles, and head items seem underrepresented, an overall similar distribution between user profiles and recommendations can be observed.
Overall, we conclude that Spotify recommendations create very balanced recommendations that, filtered by usable data for the recommendation algorithm seem to reflect the user profile appropriately. The distributions seem skewed in the filtered subset, but we assume a close relation to the user profile.

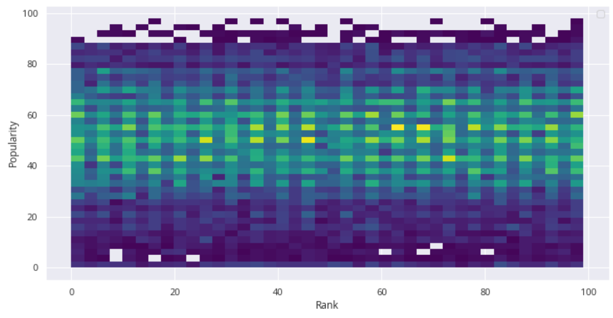


Figure 6.4: Comparison of Spotify recommendation rank and popularity on all recommendations

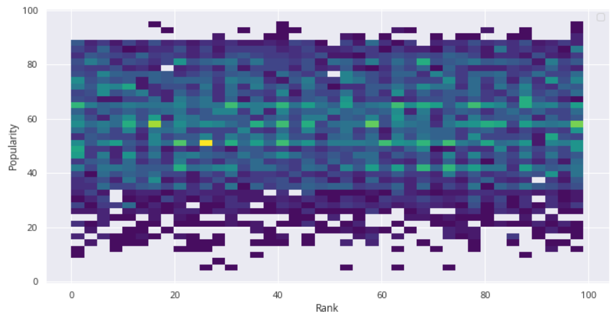


Figure 6.5: Comparison of Spotify recommendation rank and popularity on recommendations filtered by inclusion in LFM data set

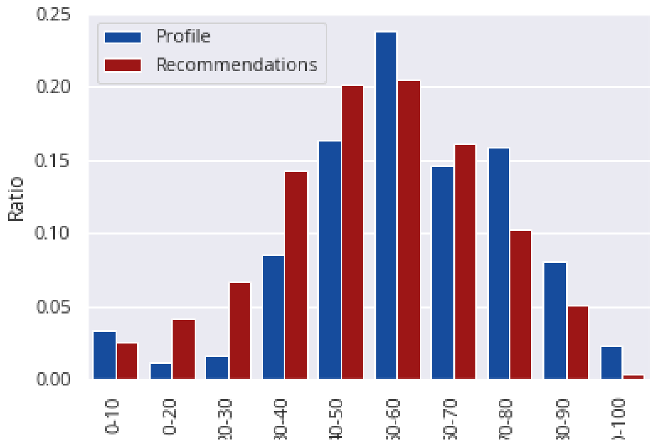


Figure 6.6: Distribution of song popularities as described by Spotify's measure on user profiles and all recommendations

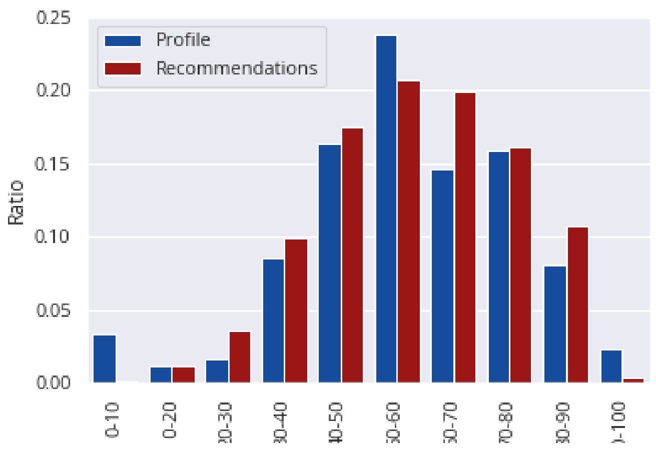


Figure 6.7: Distribution of song popularities as described by Spotify's measure on user profiles and recommendations filtered by inclusion in LFM data set

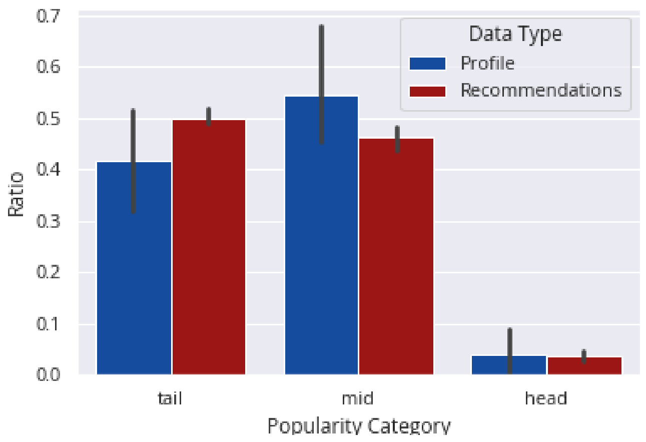


Figure 6.8: Distribution of song popularities as described by LFM category measure on user profiles and recommendations

6.1.6 The Data Scarcity Problem

The Spotify data analysis shows issues with the recommendation pipeline for the study tool. The initial approach was to use the Spotify user profiles for creating recommendations. Unfortunately, the Spotify API offers only a few usable tracks by the base recommender per user. We assume that those will not be enough to properly create a user profile, and create personalised recommendations. Since we aim at creating personalised recommendations, we have to create a workaround.
For this purpose, we assume that personalised recommendations created by Spotify recommendations as done before can represent the user profile. We have shown an approach to gathering recommendations that are, at least from a popularity perspective, comparable to the user profiles. Therefore, we will make use of the Spotify recommendations as representations of user profiles. We acknowledge that this is not the optimal representation of the user profile, but it offers the opportunity to use the validated recommendation algorithm while making use of personalised user profiles. Nevertheless, we have to consider that the distribution of head, mid and tail items in the user profiles does not match the distributions found in the original LFM dataset. Tail items are overrepresented and head items are underrepresented (see figure 6.8). This analysis only relies on 4 users. Since the presented approach will be used in the study, this will be further investigated and considered in the analysis.
The created recommendations also do not avoid tracks that already occurred in the user profile. Therefore, we assume that a large number of songs would be in the user profile. We will create an approach to validate this assumption during the study.
Another important factor is that some user data might still not be usable. For example, for one user, only 2% of the recommendations were usable by the LFM base recommender. We assume that this will not be enough, and we will have to exclude users with very few usable recommendations from the user study.
In the following chapter, we will explain the recommendation pipeline and creation of the user study tool in detail.

6.2 The Study Tool

For conducting the study, we develop a tool that enables us to generate and present song recommendations based on Spotify data[3]. In the previous section, we explained to what degree data retrieved from the Spotify API is feasible for usage by the model trained on the Last.fm dataset. Simply using the recommendations provided by Spotify is not applicable since it does not provide ranked recommendations and no scores that can be used by the mitigation algorithm. Additionally, the retrievable songs from the user profiles are not sufficient for creating appropriate user profiles. Therefore, we use Spotify recommendations to represent the user profile.
This user profile is further used to create recommendations with the base algorithm trained on the LFM-2b dataset. We can apply the mitigation strategies to those. While the created recommendations are not entirely comparable to those from the algorithmic evaluation due to the workaround of using recommendations as profiles, we assume that similar effects can be observed.
The recommendation tool aims to generate recommendations given different recommendation/mitigation strategies. The user should be enabled to investigate recommendation playlists and accomplish the task given by the researchers. The tool should be self-explanatory and should be operated easily by the users without support from the researchers. It operates in parallel to filling in the questionnaires. We develop the following properties:

- Firstly, the user can log in with their Spotify account. The Spotify API[4] enables retrieval of data from the user's top tracks.
- Based on the retrieved data, Spotify recommendations are created which represent the user profile.
- The main part of the tool is the creation of recommendations. In random order, the user receives personalised recommendation playlists, either unchanged or manipulated by item or user-centred mitigation.
- The user is presented with an interface in which they can interact with the tool. The user can listen to song previews and select songs based on the given task.
- After accomplishing the tasks, the user can add the recommendation playlists to their Spotify account.
- All important interaction data of the user is stored in a log file.

The recommendation tool is developed in Python, which opens HTML templates that are the interaction points for the user. User authentication with Spotify is accomplished with the pPython library `Spotipy`. Furthermore, scripts and models that were implemented in chapter 3 are adapted and used for creating personalised recommendations.

6.3 The recommendation pipeline

Based on the previously mentioned properties, we create a recommendation tool which uses the pipeline in figure 6.9. In the following, we explain the development and implementation of the steps of the recom-

[3]The code and data of this evaluation are available at https://git.science.uu.nl/0982717/mitigatingpopularitybiascode.git
[4]https://developer.spotify.com/documentation/web-api

mendation pipeline.
After gaining access to the study tool, the user can access the study tool and can authorize access to their Spotify account. After this, the user profile is computed by making use of Spotify recommendations.
Initially, the top 50 long-term tracks of the user are retrieved by making requests to the Spotify API. Furthermore, those are used as seed songs for creating recommendations. For ten iterations, the 50 songs are split into random batches consisting of 5 songs. Those are used as seed songs and 100 recommendations are created per batch, resulting in 1,000 songs per iteration, and 10,000 songs overall. Those are filtered by duplicates and by appearance in the used LFM-2b subset to ensure usability by the base recommender. We assume that this filtered set of songs can represent the user profile as the data analysis has shown (see section 6.1).
This user profile representation is used to achieve personalised, ranked recommendations with the base algorithm. The profile is used to compute the user factors of the matrix-multiplication collaborative filtering algorithm `RankALS`. The user factors for one user can be computed by accomplishing the P step for the train set consisting of only the interactions for the new user with fixed item factors. After doing so, 5,000 recommendations are computed for the user; the initial base recommendations. The 25 highest ranked of those are selected for the base recommendations. After this, the mitigation algorithms are applied to the initial recommendations, creating 25 re-ranked recommendations.
Finally, the recommendation lists are prepared for displaying them to the user. The track attributes like cover, song title, artist name and cover image are retrieved from the Spotify API and displayed in order to comply with the Spotify API guidelines. The lists are stored in temporary csv files for retrieval and display during user interaction. The order in which the playlists are displayed to the users is recommended to avoid order effects.

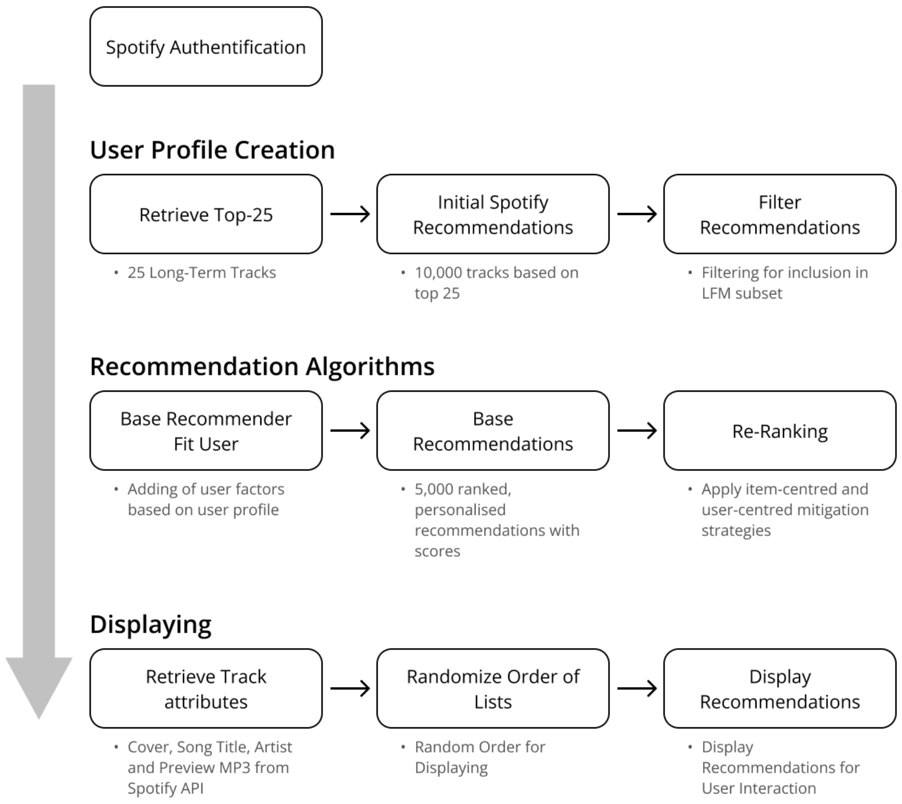
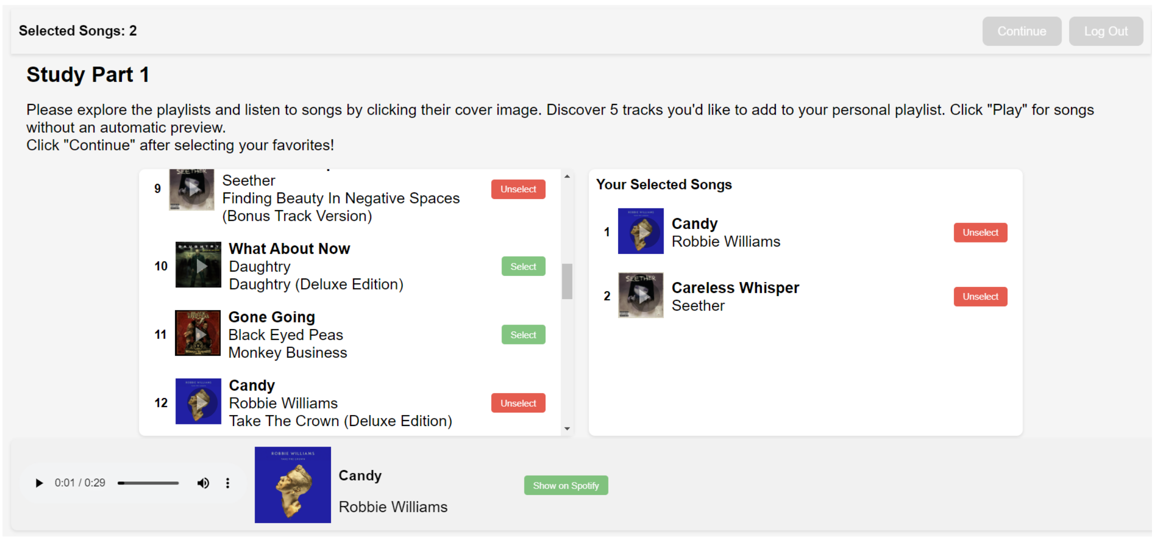

Spotify Authentification
User Profile Creation
Retrieve Top-25
• 25 Long-Term Tracks
Initial Spotify Recommendations
• 10,000 tracks based on top 25
Filter Recommendations
• Filtering for inclusion in LFM subset
Recommendation Algorithms
Base Recommender Fit User
• Adding of user factors based on user profile
Base Recommendations
• 5,000 ranked, personalised recommendations with scores
Re-Ranking
• Apply item-centred and user-centred mitigation strategies
Displaying
Retrieve Track attributes
• Cover, Song Title, Artist and Preview MP3 from Spotify API
Randomize Order of Lists
• Random Order for Displaying
Display Recommendations
• Display Recommendations for User Interaction

Figure 6.9: Recommendation Pipeline for Creating Base and Re-ranked Recommendations for the User Study

6.4 User interaction

The user interaction consists of three parts, each followed up by questionnaires. The first three are interactions with the three recommendation lists, and the fourth one is the validation of the user profile. All of those interactions start with an instructions screen, which asks the user to confirm that they reached the respective part of the questionnaire. After confirmation, the instructions for the next interaction are displayed (see figure 6.11). After clicking on ”next”, the user is shown two columns, one displaying the recommendations, and the other showing the selected songs (see figure 6.10). By clicking on the songs, a 30-second preview of the song is played. For some songs, no preview is provided. The full song is provided in those cases. By clicking on the ”select” button, the song is added to the selected songs and displayed in the right column. Additionally, the number of selected songs is displayed on top of the screen. After confirming their selection by clicking on ”Continue”, the user is redirected to the questionnaire. For the profile validation, a similar interface as for the recommendation screens was developed. It is not ranked and consists of only one column. After completing the last questionnaire, the user can save the all recommended songs as a playlist to their Spotify account.


Selected Songs: 2

Study Part 1

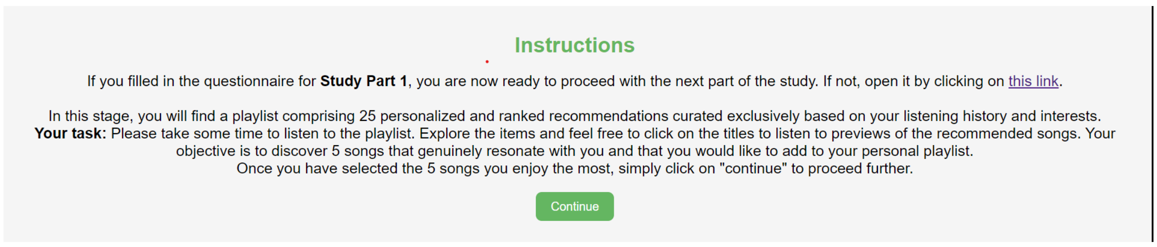
Please explore the playlists and listen to songs by clicking their cover image. Discover 5 tracks you'd like to add to your personal playlist. Click "Play" for songs without an automatic preview.
Click "Continue" after selecting your favorites!

9 Seether, Finding Beauty In Negative Spaces (Bonus Track Version) — Unselect
10 What About Now, Daughtry, Daughtry (Deluxe Edition) — Select
11 Gone Going, Black Eyed Peas, Monkey Business — Select
12 Candy, Robbie Williams, Take The Crown (Deluxe Edition) — Unselect

Your Selected Songs
1 Candy, Robbie Williams — Unselect
2 Careless Whisper, Seether — Unselect

0:01 / 0:29 Candy Robbie Williams Show on Spotify

Figure 6.10: The interface for interactions with the recommendation lists
On the top, the number of selected songs is displayed. Below, a short summary of the instructions can be seen. The focus of the applications is two columns. On the left, the recommendations are displayed in a ranked manner. In the right column, the selected songs are shown.
On the bottom of the screen, a player UI element can be seen playing the previews of the songs.



Instructions

If you filled in the questionnaire for **Study Part 1**, you are now ready to proceed with the next part of the study. If not, open it by clicking on this link.

In this stage, you will find a playlist comprising 25 personalized and ranked recommendations curated exclusively based on your listening history and interests.
Your task: Please take some time to listen to the playlist. Explore the items and feel free to click on the titles to listen to previews of the recommended songs. Your objective is to discover 5 songs that genuinely resonate with you and that you would like to add to your personal playlist.
Once you have selected the 5 songs you enjoy the most, simply click on "continue" to proceed further.

Figure 6.11: The instruction screen for the three main interactions with the application

6.5 Interaction Data

After computing the personalised recommendations, important metrics for the evaluation of the user profile and the recommendations are computed and saved to a csv file. The following metrics are computed for the user profile:

- **User id:** A randomly generated user id, generated for matching the interaction data with the questionnaire data.
- **# Profile songs:** The number of initial Spotify recommendations in the user creation step (see figure 6.9).
- **# Filtered profile songs:** The number of items in the user profile.
- **Head ratio, mid ratio, tail ratio:** The ratio of items of the respective category in the user profile.
- **Mean interactions:** Mean number of interactions with the songs in the LFM-2b dataset.
- **Median interactions:** Median number of interactions with the songs in the LFM-2b dataset.

For the recommendations, the URI of each song is saved as well as the following metrics:

- **User id**
- **Condition:** The recommendation list (*base*, `FA*IR`, or `CP`)
- **Head ratio, mid ratio, tail ratio:** The ratio of items of the respective category in the recommendation list.
- **Mean interactions:** Mean number of interactions with the songs in the LFM-2b dataset.
- **Median interactions:** Median number of interactions with the songs in the LFM-2b dataset.
- **Popularity lift:** The popularity lift of the recommendations in comparison to the user profile.
- **UPD:** The Jensen-Shannon divergence between the songs in the user profile and the songs in the recommendation list.

After the user interacts with the recommendation lists, interaction data is saved as well. The time stamp and the time the user interacted with the playlist are saved as well as the URIs of the songs that were selected by the user. To ease the post-processing, some metrics were immediately computed. Those are the same as the ones for the recommendation lists but are limited to the subset of choices that the user made.

6.6 Technical Details

This section provides a comprehensive overview of the technical aspects of the application. It leverages HTML templates and JavaScript to create an interactive user interface. The application is deployed on the cloud platform Heroku. This choice of hosting enables easy deployment and ensures that the application is accessible to users from various locations. Python 3.8.17 is the programming language employed in the application, which offers a wide array of libraries and tools for efficient development. The application uses one basic Dyno for running the application and opening HTML templates. A worker Dyno is used for computing the recommendations.
The application integrates seamlessly with the Spotify API to gain access to user listening data and facilitate personalised recommendation generation. This interaction with the Spotify API is made possible through the `Spotipy` library, a Python wrapper for the Spotify Web API. When a user accesses the application, they are prompted to log in to their Spotify account and grant necessary permissions which are inquired using the scope parameter. Upon successful authentication, the application obtains an access token, which is used to make subsequent API requests on behalf of the user. This approach was explained in section 6.1.1. The scope consists of various permissions for accessing user data through the Spotify Web API. For this application, it includes permissions for accessing a user's private profile information and email associated with their Spotify account. Additionally, it allows retrieving recently played tracks and the user's top tracks and artists based on their listening behaviour. This is necessary to create user profiles. Furthermore, the scope grants the ability to modify both private and public playlists owned by the user. This is used to enable the user the option of saving the recommendations of the study to their account.
To ensure a smooth and responsive user interface, the application employs a separate worker to handle computationally intensive tasks, particularly the recommendation pipeline (see figure 6.9). The worker is a separate process that runs in the background alongside the main web application on the Heroku platform. By delegating these tasks to the worker, the main application remains available to respond to user requests without delays caused by resource-intensive computations. While the worker is running, the user is redirected to a loading screen and they are asked to fill in the pre-questionnaire. To enhance the worker's functionality and scalability, Redis[5], an in-memory data store, is utilized as the message broker. Redis acts as a communication hub between the main application and the worker, enabling data exchange and coordination. When a user initiates the recommendation process by accessing the application and granting Spotify permissions, the main application enqueues a recommendation task to Redis. The worker continuously monitors the Redis queue, awaiting incoming tasks.
The computational power of the worker is limited, and particularly the base model consumes much computational power for a short time. To handle this issue, the creation of the base recommendations is accomplished on a local server. The application uses NGROK[6], a tool that creates secure tunnels to localhost, to access a base model running on a local server. NGROK ensures secure communication between the application and the local server, protecting sensitive data during transmission.

[5]https://redis.io/
[6]https://ngrok.com/

To handle data storage efficiently, the application creates designated directories to store recommendation files and log data during the recommendation process. This organization prevents data conflicts and enables concurrent use by multiple users. For secure storing of the log and interaction files, the files are automatically sent by e-mail to the researcher who further processes and stores those files. For this purpose, the library `smtplib`[7] is used to interact with Simple Mail Transfer Protocol (SMTP) servers to send emails programmatically.
The application incorporates robust error-handling mechanisms to address potential issues that may arise during the recommendation generation and data processing phases. For instance, if there is insufficient data available to generate personalised recommendations, the user is directed to a failure screen, and the error is logged for further investigation and improvement.
To streamline the user experience during the study, the application automatically redirects users to appropriate questionnaires hosted on Qualtrics[8], a popular survey platform. This integration is achieved through HTML scripts that provide the user's unique ID and condition as URL queries. These queries are then used by Qualtrics to associate questionnaire responses with the correct user, simplifying data collection and analysis.
Users have the option to save recommended songs to a personalised Spotify playlist directly from the application. This functionality is achieved by facilitating the creation of a playlist on the user's Spotify account with a single click, enhancing the user experience.
To keep track of user-specific data and access tokens throughout the recommendation process, the application manages user sessions. Users can log out at any point if they decide to withdraw from the study, which clears their session and revokes the application's access to their Spotify account, ensuring data privacy and security.

[7]https://docs.python.org/3/library/smtplib.html
[8]https://www.qualtrics.com/uk/

6.7 Evaluation Model

For analysing the user perception and effects of the different conditions, we adopt a methodology similar to Knijnenburg et al. [60]. The initial goal of this analysis is to build a structural equation model (SEM). Overall, following this methodology, we aim to achieve a robust and valid representation of the underlying relationships between the variables in our research, building on the foundation laid by prior validation studies and refining the model based on our data.
In line with [60], we assume some hypotheses for the directionality of causal effects between the factors of the concepts. We test the following directions:

- PC $\rightarrow$ SSA; PC $\rightarrow$ EXP; PC $\rightarrow$ INT; PC $\rightarrow$ BI
- OSA $\rightarrow$ SSA; OSA $\rightarrow$ EXP; OSA $\rightarrow$ INT; OSA $\rightarrow$ BI
- SSA $\rightarrow$ EXP; SSA $\rightarrow$ INT; SSA $\rightarrow$ BI
- EXP $\rightarrow$ INT; EXP $\rightarrow$ BI
- INT $\rightarrow$ EXP; INT $\rightarrow$ BI

Since the Structural Equation Model (SEM) is not feasible to use due to the limited number of users, we opt for a pragmatic approach similar to Knijnenburg et al.'s methodology [61] to evaluate the results. They indicate that this approach can be applied to studies with a minimum of 20 users per condition. In this alternative method, we make assumptions about the same directional causal effects as in the SEM. Subsequently, our investigation focuses on exploring the effects of the factors on each other. We computed the average questionnaire scores and normalized them from a 1 to 7-point Likert scale to the range -3 to +3. This way, the neutral option is 0. For the effects of numeric factors, we calculate Pearson's correlation coefficients to understand their relationship and potential influences. However, for factors that are categorical, such as "Condition" and "Trial Number," we employ repeated measures one-way ANOVAs to analyze the variance and determine any significant differences between their respective groups. In line with [61] we provide the p-value and the effect size r. The effect size of the correlation is Pearson's r for the correlation tests and measures the strength and direction of the relationship between two variables. Eta-squared (η^2) is provided as the effect size for the repeated measures one-way ANOVA. It measures the association defined as the ratio of variance in the outcome variable explained by the predictor variable. For analysing the strengths of the correlation, we rely on standard definitions shown in table 6.2
We compute correlations and ANOVA tests between all appropriate metrics. By adopting this pragmatic procedure with an exploratory nature, we aim to gain meaningful insights into the underlying relationships between the variables without requiring an extensive user base that a traditional SEM might demand. This approach allows us to remain clear and concise while ensuring a thorough analysis of the data and its implications for our research.

Correlation Strength	**Correlation Coefficient (r)**	**Interpretation**
Negligible	$-0.1 \leq \lvert r \rvert < 0.1$	Little to no linear relationship
Weak	$0.1 \leq \lvert r \rvert < 0.3$	Weak linear relationship
Moderate	$0.3 \leq \lvert r \rvert < 0.5$	Moderate linear relationship
Strong	$0.5 \leq \lvert r \rvert < 0.7$	Strong linear relationship
Very Strong	$0.7 \leq \lvert r \rvert < 0.9$	Very strong linear relationship
Near Perfect	$0.9 \leq \lvert r \rvert < 1$	Almost perfect linear relationship
Perfect	$\lvert r \rvert = 1$	Perfect linear relationship

Table 6.2: Strength Levels of Correlation

Chapter 7

Study Evaluation

7.1 User Profiles

34 users participated in the experiment. 18 users identify as male, and 16 as female. No users indicated that they identify as non-binary or that they prefer not to say. The mean age of the participants is 25.82 ($SD = 5.43$) with the youngest being 22 and the oldest being 52.
On average, 1092.56 ($SD = 408.39$) unique tracks were extracted to represent the user profiles. In figure 7.1, descriptive metrics of the popularity of the songs in the user profiles can be seen. Generally, the mean (average of 676.61, $SD = 330.29$) and median (average of 144.93, $SD = 122.72$) popularity is much lower than the popularity of the user profiles in the algorithmic evaluation with an average mean popularity of 1246.25 and an average median popularity of 586.77 (cf. appendix A). The median popularity is much lower than the mean popularity, indicating the low popularity of many items.
When investigating the popularity ratios in the user profiles, a similar effect can be observed. Figure 7.2 shows the ratio of head, mid and tail items in the user profiles. When comparing this to the profiles in the algorithmic evaluation (cf. figure 4.2a), it can be seen that tail items are over-represented in the user profiles of this study in comparison to even niche users in the original dataset. Additionally, the ratio of head items in the user profiles of the study participants is also under-represented compared to blockbuster-focused and diverse users and over-represented in comparison to niche users.
When using the boundaries for blockbuster-focused, mid and niche users from the algorithmic evaluation (see section 4.2), 17 users can be classified as niche users and 17 as diverse users. No user can be classified as a blockbuster-focused user.
Participants were presented with a profile validation playlist. Afterwards, they had to rate how many of the songs in the playlists were known to them (in %) and whether they would say that the songs matched their taste. 18 users indicated that they knew less than 50% of the songs, and 7 users indicated that the match was less than 50%. The average score for whether the songs were known is 46.68% ($SD = 23.77$), and the average score for the preference match is 66.24 ($SD = 20.00$).
Some observations can be made when investigating the influences of the musicality of the participants on their user profiles. Figure 7.3 shows that *Musical Engagement* correlates negatively with the mean popularity of the songs in the user profile as well as with the ratio of head items in the profile. On the other hand, it correlates positively with the mid ratio.
Furthermore, *Musical Sophistication* has associations with each of the popularity metrics in the user profiles. *Musical Sophistication* indicates lower popularity in terms of median and mean popularity, smaller profile sizes, and a smaller ratio of head and mid items. On the other hand, *Musical Sophistication* is positively associated with the ratio of tail items. Finally, Age and Gender do not have any significant relationships with any of the other OSA.

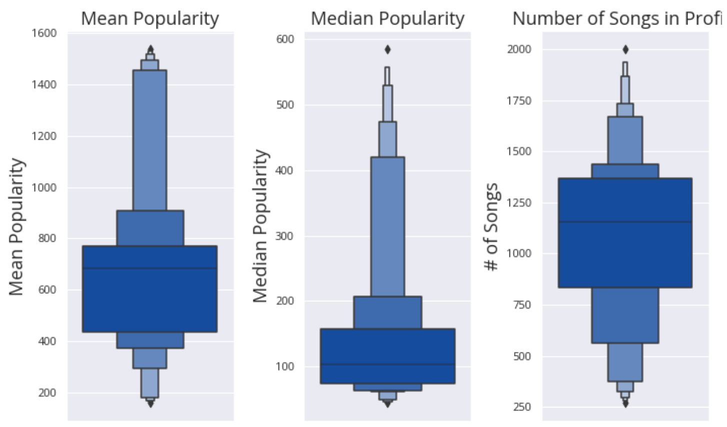


Figure 7.1: Descriptive popularity metrics of the user profiles

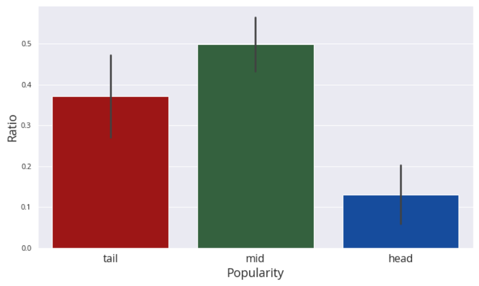


Figure 7.2: Distribution of popularity groups in user profiles
Mean (SD):
Tail: 0.3713 (0.1009), Mid: 0.5985 (0.0662), Head: 0.1302 (0.0718)
Minimum - maximum:
Tail: 0.1287 - 0.5453, Mid: 0.3477 - 0.6493, Head: 0.0123 - 0.3069

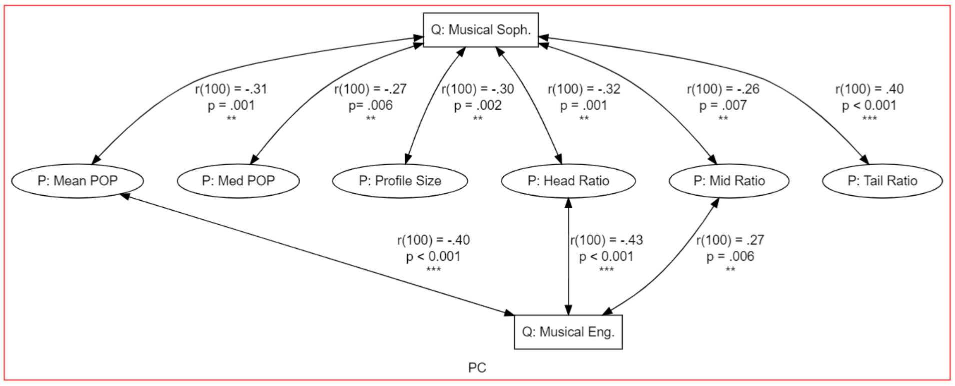

Q: Musical Soph.
r(100) = -.31 p = .001 **
r(100) = -.27 p= .006 **
r(100) = -.30 p = .002 **
r(100) = -.32 p = .001 **
r(100) = -.26 p = .007 **
r(100) = .40 p < 0.001 ***
P: Mean POP
P: Med POP
P: Profile Size
P: Head Ratio
P: Mid Ratio
P: Tail Ratio
r(100) = -.40 p < 0.001 ***
r(100) = -.43 p < 0.001 ***
r(100) = .27 p = .006 **
Q: Musical Eng.
PC

Figure 7.3: Correlations of the *Musical Engagement* and *Musical Sophistication* questionnaires with other PC

7.2 Recommendation Analysis

When observing the recommendations created by the base algorithms and the mitigation algorithms, we can observe clear effects of the algorithms. Table 7.1 shows that the condition affects the mean and median popularity, the popularity lift, the UPD as well as the ratios of the popularity categories. In figure 7.9 and 7.10, we conducted more in-depth posthoc tests. Paired t-tests show that there are significant differences between each algorithm for all metrics.

When observing the distribution of the mean and median popularity (see figure 7.4), it can be seen that the base algorithm creates mean recommendations that have similar mean popularity to the user profiles on average (Profile: 144.92; Base: 658.07). The median popularity is almost 5 times higher (Profile: 144.93; Base: 583.68). `CP` creates recommendations with lower mean and median popularity on average than the base. Its mean popularity is lower than the one of the profiles, while the median popularity is higher in comparison to the median of the profiles. `FA*IR` on the other hand shows much lower mean and median popularity values on average with only a few outliers extending a mean of 400 and a median of 100.
When analysing the user-centred metrics (see figure 7.5), it can be seen that the algorithms accomplish their specified tasks. `FA*IR` massively reduces the popularity lift while `CP` reduces the miscalibration in terms of *UPD*, measured by the Jensen-Shannon Divergence, to almost 0. This is reflected in the ratios of the popularity categories (see figure 7.6). While the base algorithm mainly recommends mid items and some head items, `FA*IR` recommends mainly tail items. An in-depth analysis shows that `FA*IR` almost ensures 92% tail items. Out of 31 participants, 25 users received 92% tail items. This is due to the nature of `FA*IR` which tries to ensure a minimum amount of unprotected items. `CP`, on the other hand, promotes tail and head items in comparison to the base algorithm and achieves a similar distribution to the user profiles.
The results show that the condition/algorithm has significant effects on the structure of the recommendation lists. For further insights into the differences between the playlists, we compute the *Jaccard Similarity* between the different recommendation lists. It indicates the overlap of songs between the different playlists. Table 7.2 shows the average similarity between the algorithms. This shows that the recommendations deviate quite highly from each other. Especially `FA*IR` has very low similarity with the recommendation list created by the base algorithm, presumably due to the high number of added tail items that do not appear in the original list.

Factor	Effect size (η^2)	Statistic ($F(2, 100)$)	p	$p < .05$
R: Mean POP	0.59	252.57	<.001	***
R: Med POP	0.49	123.19	<.001	***
R: Pop Lift	0.59	135.25	<.001	***
R: UPD	0.78	190.97	<.001	***
R: Head Ratio	0.47	77.23	<.001	***
R: Mid Ratio	0.87	372.17	<.001	***
R: Tail Ratio	0.87	463.17	<.001	***

Table 7.1: Statistics of the one-way ANOVA analysing the impact of the algorithms on the popularity metrics

Algorithm 1	Algorithm 2	Mean	SD	Minimum	Maximum
Base	FA*IR	0.1133	0.1786	0.0417	0.8519
Base	CP	0.4570	0.0846	0.3158	0.6129
FA*IR	CP	0.3018	0.1201	0.1905	0.7241

Table 7.2: Jaccard Similarity between the recommendation lists created by the different algorithms

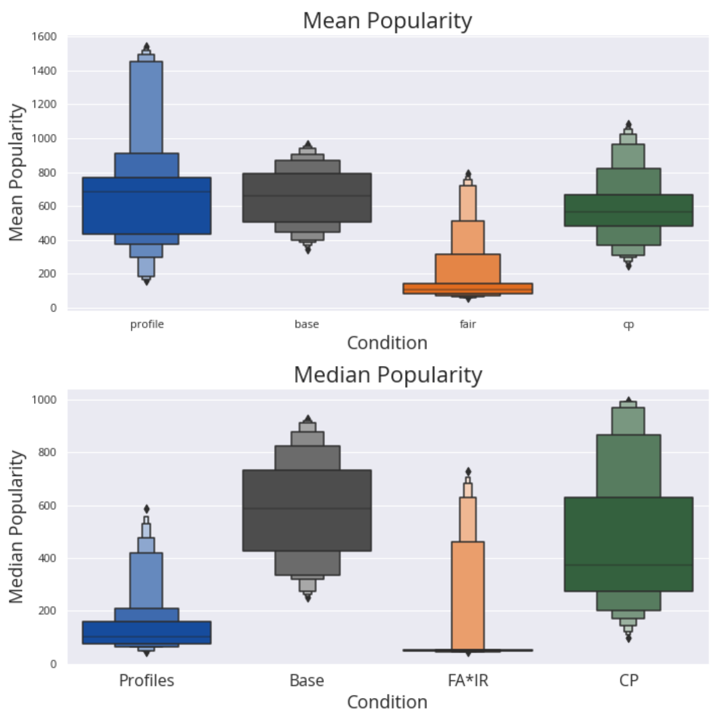


Figure 7.4: Comparison of descriptive metrics between the algorithms on the recommendations created for participants

Metric	Algorithm	Mean	SD	Min	Max
Mean POP	Profile	676.6120	330.2900	159.7352	1538.0619
	Base	658.0694	170.9416	345.6800	963.0800
	FA*IR	171.4988	178.3492	59.5200	790.5200
	CP	593.4929	198.5704	250.0800	1081.2800
Median POP	Profile	144.9265	122.7164	44.0000	585.5000
	Base	583.6765	195.1075	249.0000	926.0000
	FA*IR	98.8529	166.3654	44.0000	729.0000
	CP	460.9412	263.7734	96.0000	994.0000

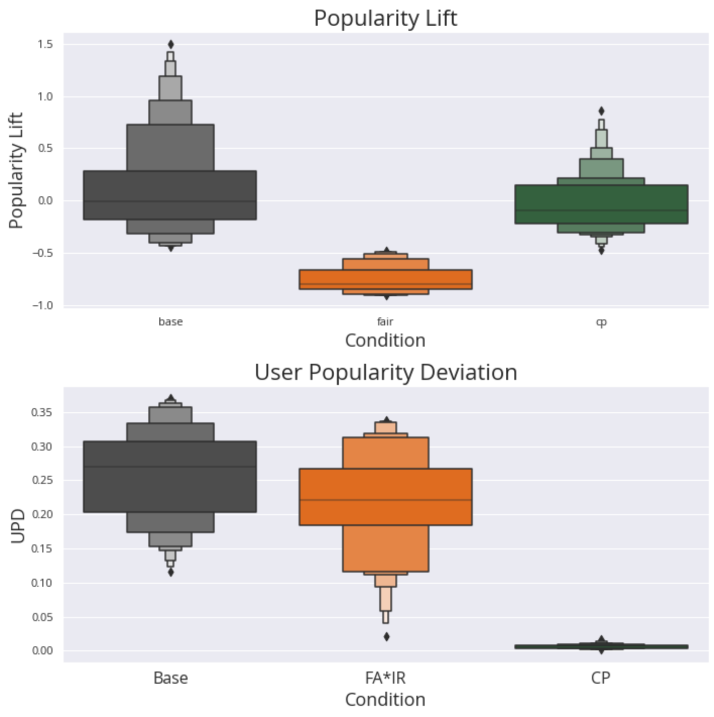


Figure 7.5: Comparison of user-centred metrics between the algorithms on the recommendations created for participants

Metric	Algorithm	Mean	SD	Min	Max
POP Lift	Base	0.1403	0.4803	-0.4389	1.4966
	`FA*IR`	-0.7579	0.1282	-0.9085	-0.4860
	`CP`	-0.0306	0.2809	-0.4735	0.8663
UPD	Base	0.2591	0.0681	0.1162	0.3698
	`FA*IR`	0.2173	0.0762	0.0215	0.3373
	`CP`	0.0061	0.0031	0.0020	0.0168

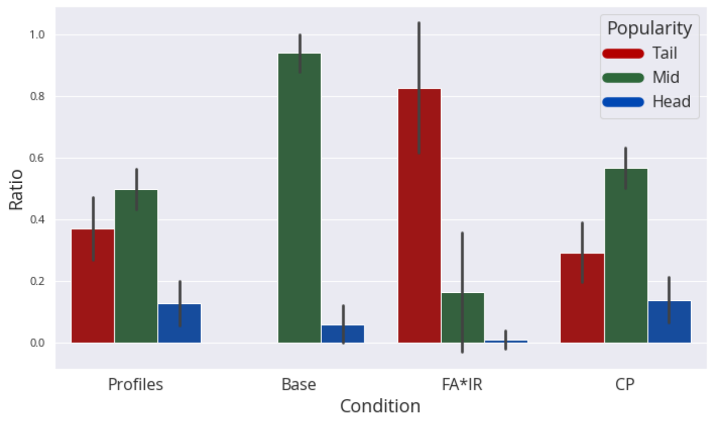
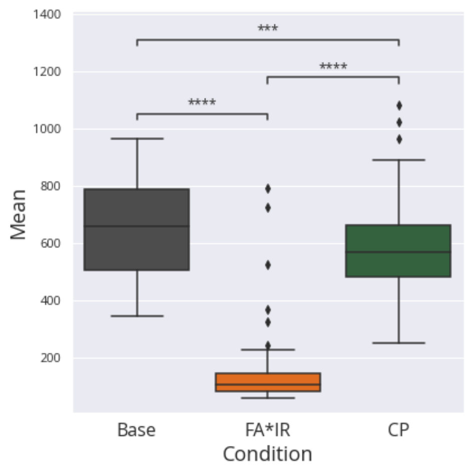


Figure 7.6: Comparison of category ratios between the algorithms on the recommendations created for participants

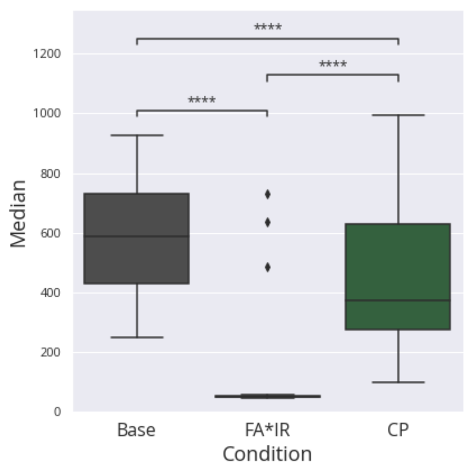
Ratio	Algorithm	Mean	SD	Min	Max
Head	Profile	0.1302	0.0718	0.0123	0.3069
	Base	0.0612	0.0601	0.0000	0.2400
	`FA*IR`	0.0106	0.0300	0.0000	0.1600
	`CP`	0.1400	0.0745	0.0400	0.3200
Mid	Profile	0.4985	0.0662	0.3477	0.6493
	Base	0.9388	0.0601	0.7600	1.0000
	`FA*IR`	0.1635	0.1926	0.0400	0.8800
	`CP`	0.5671	0.0642	0.4000	0.7200
Tail	Profile	0.3712	0.1009	0.1287	0.5453
	Base	0.0000	0.0000	0.0000	0.0000
	`FA*IR`	0.8259	0.2112	0.0800	0.9200
	`CP`	0.2929	0.0958	0.0800	0.4800



(a) Mean

Results of paired t-tests:

Condition 1	Condition 2	$t(101)$	p
Base	FA*IR	16.77	<.001
Base	CP	3.684	<.001
FA*IR	CP	-18.71	<.001

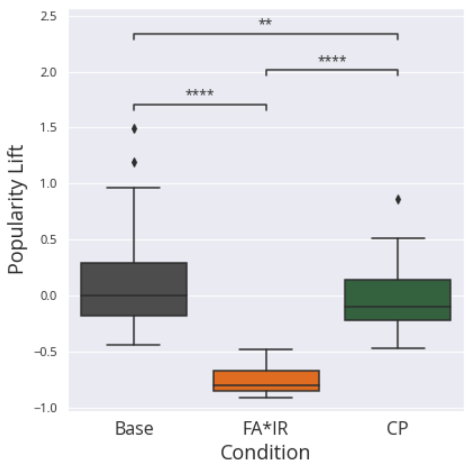


(b) Median

Results of paired t-tests:

Condition 1	Condition 2	$t(101)$	p
Base	FA*IR	14.54	<.001
Base	CP	4.64	<.001
FA*IR	CP	-10.09	<.001

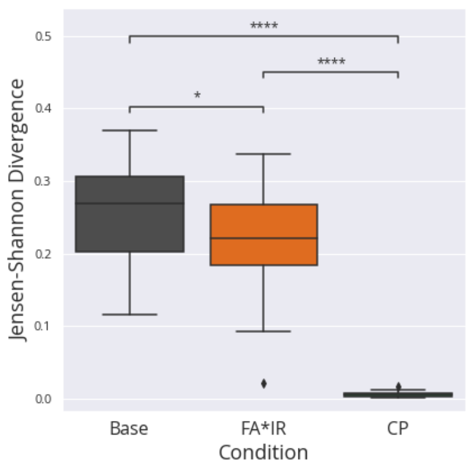
Figure 7.7: Descriptive Metrics



(a) POP Lift

Results of paired t-tests:

Condition 1	Condition 2	$t(101)$	p
Base	FA*IR	-11.18	<.001
Base	CP	3.47	.001
FA*IR	CP	-15.49	<.001



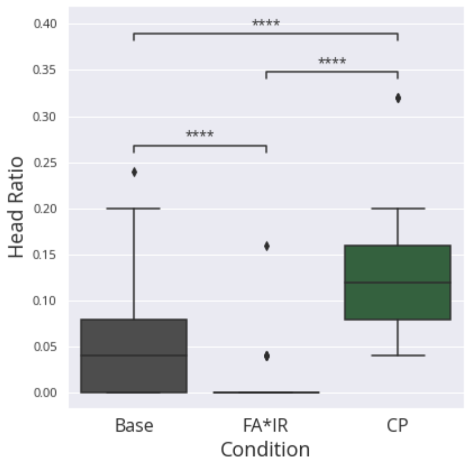
(b) Jensen-Shannon

Results of paired t-tests:

Condition 1	Condition 2	$t(101)$	p
Base	FA*IR	2.62	.013
Base	CP	21.90	<.001
FA*IR	CP	16.06	<.001

Figure 7.8: User-centred Metrics

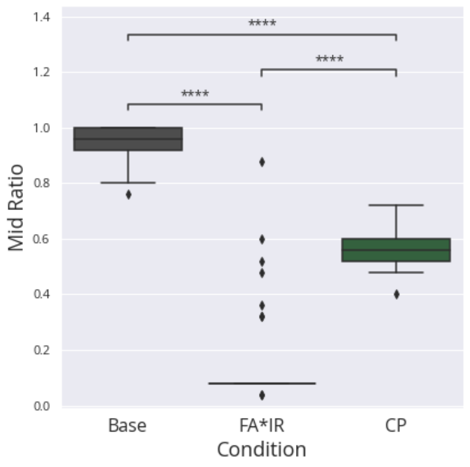
Figure 7.9: Post-Hoc Analysis: Paired t-tests between the conditions for various metrics



(a) Head Ratio

Results of paired t-tests:

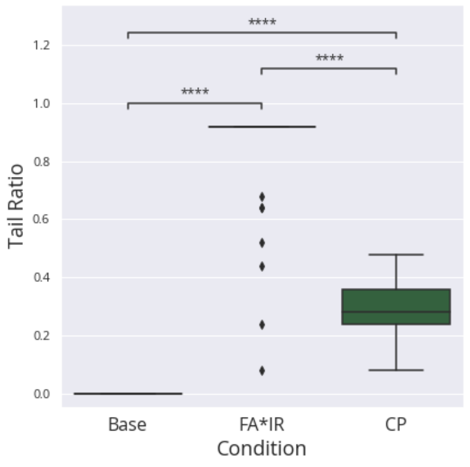
Condition 1	Condition 2	$t(101)$	p
Base	FA*IR	5.63	<.001
Base	CP	-6.57	<.001
FA*IR	CP	-11.23	<.001



(b) Mid Ratio

Results of paired t-tests:

Condition 1	Condition 2	$t(101)$	p
Base	FA*IR	21.47	<.001
Base	CP	27.62	<.001
FA*IR	CP	-13.25	<.001



(c) Tail Ratio

Results of paired t-tests:

Condition 1	Condition 2	$t(101)$	p
Base	FA*IR	-22.80	<.001
Base	CP	-17.83	<.001
FA*IR	CP	20.30	<.001

Figure 7.10: Post-Hoc Analysis: Paired t-tests between the conditions for category ratios

7.3 Ranking

At the end of the post-questionnaires, the participants were asked to provide a ranking of the three playlists they interacted with and to provide an explanation for their ranking. In figure 7.11, it can be seen how often each Rank was selected for each algorithm. To further investigate those results, we employed a Chi-Square test for independence to examine the relationship between these categorical variables. The resulting cross-tabulation table was visualized using a heatmap, depicted in figure 7.12. Notably, the figure illustrates the distribution of the rankings across the 'condition' categories, with the colour intensity indicating the frequency of observations. The Chi-Square test shows a significant association between the variables ($X^2(4, N = 34) = 16.24, p = 0.003$). The expected value for each cell is 11.3333. Based on the heatmap, we can identify that the base is much less chosen as the best playlist while `FA*IR` receives the most first ranks. Interestingly, while being placed most in the first place, `FA*IR` is more often placed in the third place than in the second. It was only 4 times placed second. `CP`, on the other hand, is evenly distributed with 11 or 12 placements per rank.

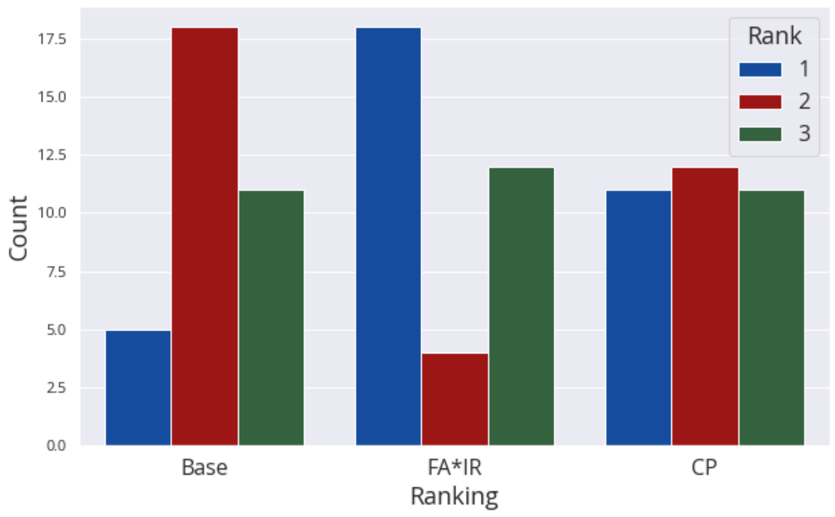


Figure 7.11: Counts of Ranking Category by Condition

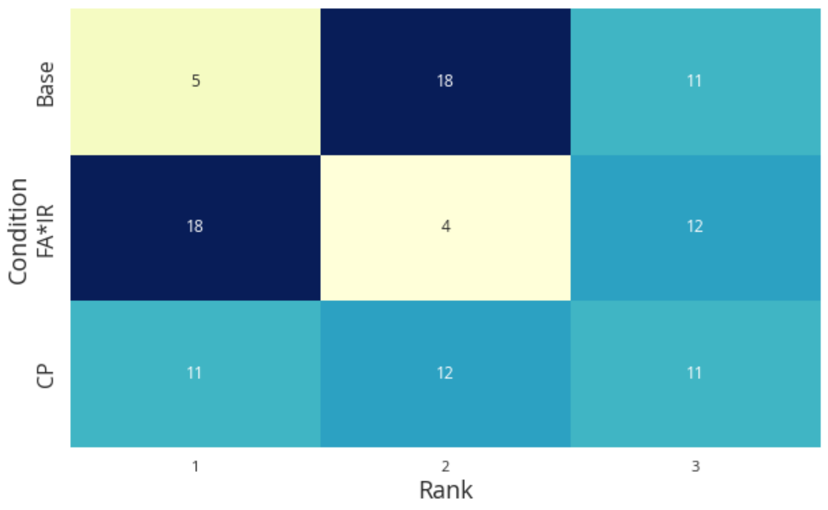


Figure 7.12: Crosstab Heatmap depicting the distribution of rankings across distinct different conditions. Colour intensity indicates the frequency of observations

7.3.1 Qualitative Analysis

The participants provided various explanations for their rankings. First of all, eight users expressed that there was repetition between the playlists. Mainly, users reported high overlap between the songs in the playlist consisting of the base recommendations and the CP recommendations. Additionally, two users pointed out that CP appeared to be a mix between the other two playlists. Generally, repetition was negatively perceived as it seemed uninteresting and repetitive to the users.
General factors that contributed to users liking the system were a good *"mix of genre and style"*, surprising and exciting choices, and those that matched the user's preferences. 7 users mentioned positively that some of the playlists matched their taste, preferences, style or expectations in a recommender. 4 users mentioned a mismatch between their preferences and the provided songs. For example, one user annotated that the *"songs were not of bad quality, but did not fit my preferences"*.
Another criterion that was pointed out by multiple users is the difficulty of picking multiple songs. A playlist was ranked lower when it was difficult to select 5 songs according to the experimental task. Nevertheless, two participants mentioned that they preferred playlists that had few very good songs that were easy picks instead of a playlist that was overall good but didn't have highlights.
The factor that was mentioned most prominently was how familiar the users were with the songs. Familiarity with the songs could impact the rankings either positively or negatively. On the one hand, known songs could impact the ranking positively because the playlist *"contained many songs that [the users] already know and like"*. For example, one participant expected songs from artists they already knew, *"so [they] actually discover music that [they] might like"*. 4 participants mentioned familiarity with songs in the playlist positively and two users perceived it negatively if they did not know many songs or artists in the recommendations. Furthermore, 4 users pointed out positively that the recommendation list included older songs. This was perceived positively if they led to re-discovery of those songs or they were perceived as *"good old goldies"*.
On the other hand, a majority of the users perceived familiarity with songs negatively (5 users) and rated

unknown items positively (10 users). The participants did not see value in already known items because it did not allow them to discover anything new, or it was perceived as boring. Unknown songs led to "surprising and interesting" recommendations and led to the discovery of new tracks.
Popularity, however, was mentioned rarely. One user complained about `CP` having too many popular items while one other user highlighted the high number of niche items in `FA*IR` positively. Furthermore, two users pointed out a good balance between popular and unpopular songs (one for `FA*IR` and one for `CP`), which was described by one of them as the *"best of both worlds"*.

7.4 Pragmatic Evaluation

Initially, a model including all factors and paths explained in section 6.7 is created. In the future steps, we will refer to algorithmic factors (e.g., *Head Ratio*, *Popularity Lift*, *UPD*) with a prefix indicating whether it was measured on the user profile (P), e.g., "P: Mean POP" is short for the mean popularity of the songs in the user profiles. The recommendations are highlighted by an "R" and the choices, meaning the 5 selected songs in the experiment, with an "C". Questionnaires are indicated by a Q. In the following, we will not present all correlation scores between each pair. We highlight significant relationships and refrain from presenting non-significant correlations. All measurements and questionnaire scores are publicly available[1] and can be replicated.
The full model lacks clarity of meaning since it includes many factors and significant relations between factors. Upon further inspection, we investigated many correlations between factors within the INT factors, and high correlations between the OSA and the INT factors, as well as between the PC and the INT. Those correlations can mainly be attributed to the workwise of the recommender. For instance, if the user profile includes many head items, the recommender predicts more head items as well, increasing the ratio of head items in the recommendations. Consequentially, the choices of the users tend to include more head items too. Similarly, if the condition is `FA*IR`, the recommendations will include more tail items and finally, the user will choose more tail items. In the main analysis of the pragmatic evaluation, we will focus on the perception of the users as described by the questionnaires. Nevertheless, we will observe the relationships between OSA, PC and INT in the following section. To do so, we will observe smaller subsets of the whole model of the pragmatic evaluation. Factors that do not have a significant relationship with the other displayed factors will not be displayed.

7.4.1 Algorithmic effects

To investigate the effects between OSA, PC and INT, we observe the relationships between those factors. In this analysis, we mainly focus on the algorithmic effects. We include the factors that describe the songs directly in terms of popularity. This includes all popularity measures: Median and mean popularity, head, mid, and tail ratio, *Popularity Lift* and *UPD*, as measured by the Jensen-Shannon Divergence.
Intuitively, high correlations between the factors within one group exist. For example, popularity correlates negatively with the ratio of tail items in the recommendations. This is an obvious effect because less popularity indicates less popular items - tail items - in the recommendations. For an overview, we show the correlations within the concepts in the Appendix in the plot C.2 for OSA, C.1 for PC, and C.3 for INT.
Additionally, many relationships between the factors exist. We present a small subgraph in figure 7.13. We limit the analysis to the *Median Popularity* and *Tail Ratio* in the user profiles, recommendations, and choices. The graph is highly interconnected, showing many edges between the clusters. The *Median Popularity* is highly correlated with other median metrics across all concepts. Similarly, the *Tail Ratio* shows negative correlations to the *Median Popularity* in the same concept but also correlates to the tail ratio of other concepts as well as to the median popularity in other concepts. When comparing other factors, similar effects can be seen. This indicates a high cross-correlation and close relationships between the user profiles, the recommendations and the choices of the users.
Furthermore, we have shown in section 7.2 that the condition has a high impact on the OSA. When observing the results found in this section, we can argue that the condition also impacts the choices of the users. For instance, lower *Median Popularity* in the recommendations by applying `FA*IR` achieves

[1]https://git.science.uu.nl/0982717/mitigatingpopularitybiasuserstudy.git

lower *Median Popularity* in the choices of the users because the available songs have less popularity. This effect is highlighted by table 7.3 which shows that the condition has an impact on the INT factors.
This analysis shows that there is a high relationship between the algorithmic effects of the OSA, PC and INT. The*Mean Popularity* and *Median Popularity*, the group ratios are highly impacted by the user profiles. This is, finally, influential to the choices of the user. For instance, if the user has more highly popular recommendations available, they tend to choose popular recommendations too. For further analysis, we will not include the effects of the algorithmic factors between the OSA, PC and INT. The created graphs are highly inter-correlated and adding them to the further analysis creates no additional value for understanding the perception of the user.

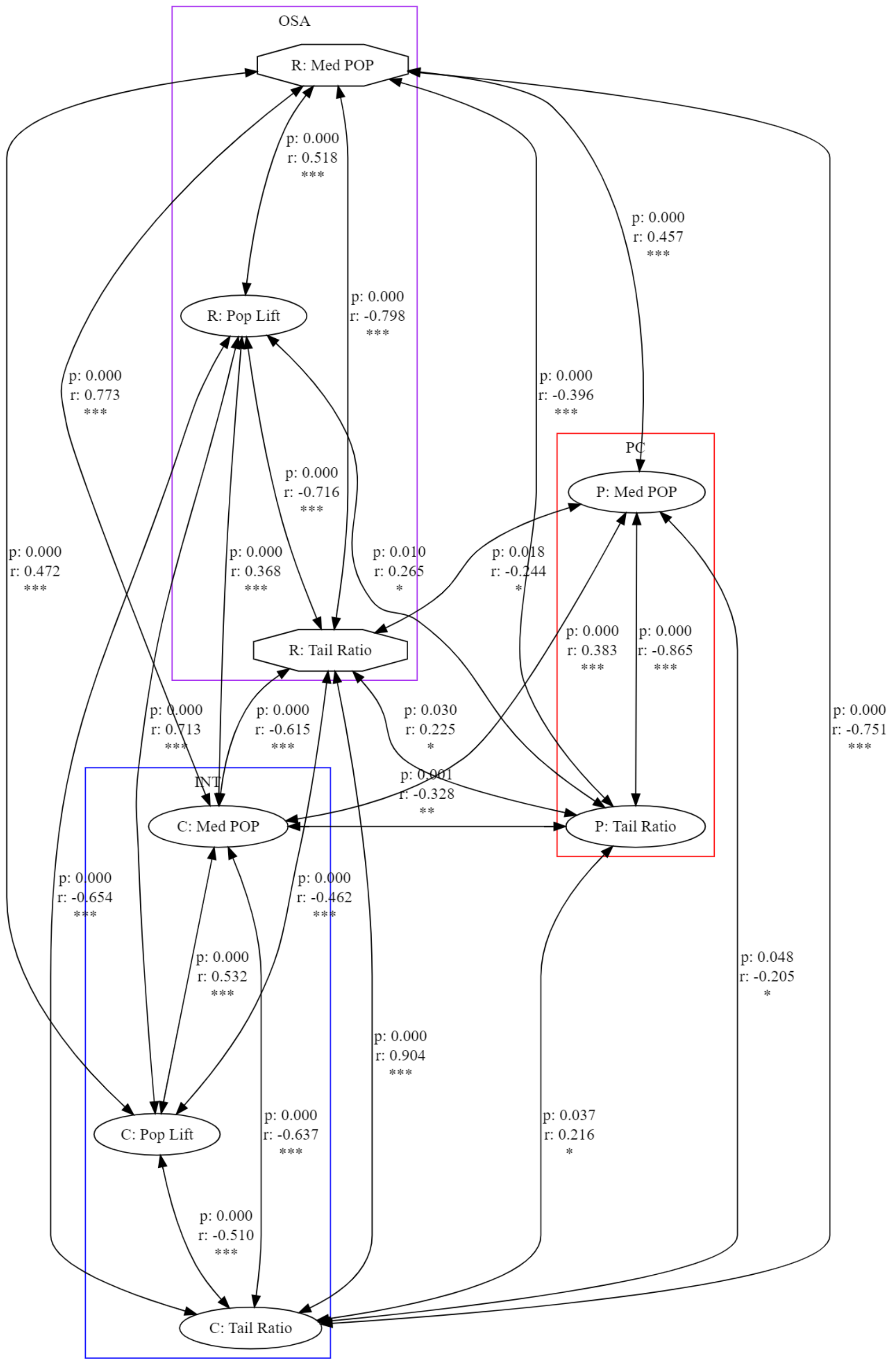
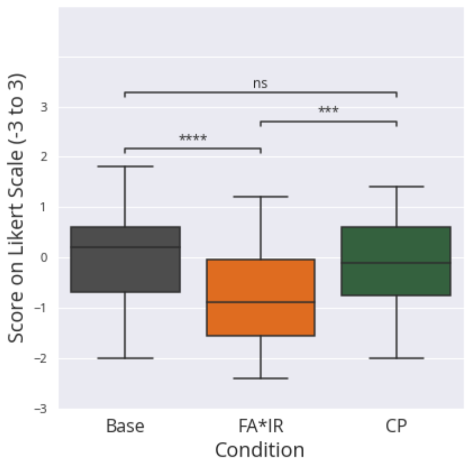


Figure 7.13: Example of effects between INT, OSA and PC factors

7.4.2 Impact of the Condition

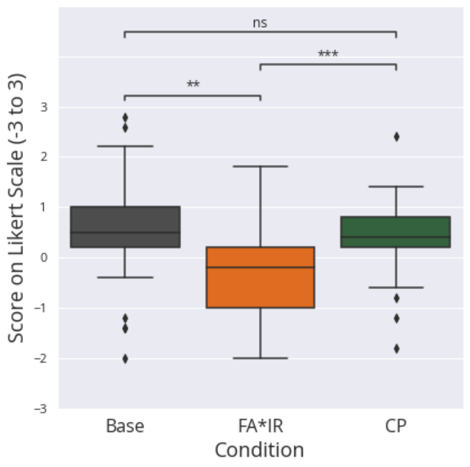
Initially, we want to investigate whether the condition directly impacts any other factors directly. For measuring the impact of the condition, we perform one-way ANOVAs. We test the impact of the condition of every other factor except the PC since those are the same for each condition for a user. The condition has no impact on the personal characteristics.
Table 7.3 shows the results of the ANOVA. In section 7.2, we explored that the condition has a strong impact on the objective system aspects. It determines the popularity distribution of the recommendations. When observing the choices made by the user, every factor is influenced by the condition. We refer to the previous section 7.4.1, which explains the relationship between the OSA and the INT. The condition does not have a direct impact on the time the user spends with the playlist. Furthermore, there were only two significant effects of the condition on Questionnaires. The condition significantly impacts the *Perceived Popularity* and *Familiarity*. Post-hoc tests (see figure 7.14) show that the ratings in terms of *Familiarity* and *Perceived Popularity* differ significantly between the base recommendations and `FA*IR` and between `FA*IR` and `CP`. There were no significant differences between the base recommendations and `CP`. In section 7.2, we found that the mean and median popularity in the playlists created by the base algorithm and `CP` are quite similar when comparing them to `FA*IR`. This is reflected in the post hoc analysis. The only perceptible difference in popularity can be seen for `FA*IR` in comparison to the base and `CP`.



(a) Q: Perceived Popularity

Results of paired t-tests:

Condition 1	Condition 2	$t(101)$	p
Base	`FA*IR`	4.81	<.001
Base	`CP`	1.433	.161
`FA*IR`	`CP`	-4.31	<.001



(b) Q: Familiarity

Results of paired t-tests:

Condition 1	Condition 2	$t(101)$	p
Base	`FA*IR`	3.60	.001
Base	`CP`	0.59	.559
`FA*IR`	`CP`	-4.28	<.001

Figure 7.14: Comparison between Popularity and *Familiarity* Questionnaire Scores per Condition

Concept	Factor	Effect size (η^2)	Statistic ($F(2, 100)$)	p	$p < .05$
OSA	R: Mean POP	0.59	252.57	.001	***
	R: Med POP	0.49	123.19	.001	***
	R: Pop Lift	0.59	135.25	.001	***
	R: UPD	0.78	190.97	.001	***
	R: Head Ratio	0.47	77.23	.001	***
	R: Mid Ratio	0.87	372.17	.001	***
	R: Tail Ratio	0.87	463.17	.001	***
SSA	Q: Perc. Popularity	0.13	16.31	.001	***
	Q: Familiarity	0.12	10.44	.001	***
	Q: Discovery	0.01	1.84	0.166	o
	Q: Perc. Rec. Quality	0.00	0.12	0.888	o
	Q: Perc. Fairness	0.0	0.08	0.923	o
EXP	Q: Rec. Satisfaction	0.01	0.68	0.512	o
	Q: Choice Satisfaction	0.00	0.33	0.719	o
	Q: Perc. System Effectiveness	0.00	0.11	0.893	o
INT	Time Spent	0.05	2.47	0.092	o
	C: Mean POP	0.19	42.29	.001	***
	C: Med POP	0.35	51.36	.001	***
	C: Pop Lift	0.25	30.77	.001	***
	C: UPD	0.24	13.76	.001	***
	C: Head Ratio	0.20	17.21	.001	***
	C: Mid Ratio	0.62	83.90	.001	***
	C: Tail Ratio	0.71	160.06	.001	***
BI	Q: Choice Listening Intention	0.01	1.06	0.354	o
	Q: Openness to Sim. Rec.	0.01	0.84	0.438	o
	Q: Use Intention	0.00	0.29	0.751	o

Table 7.3: Statistics of the one-way ANOVA analysing the differences between the conditions

7.4.3 Mediators, Experience, and Behavioural Intentions

7.4.3.1 OSA as Mediators of the Condition

In this section, we will explore whether mediators, like *Popularity Lift* which is impacted by the different algorithms (as shown in section 7.2, affect other factors directly. We have shown that the condition alone only affects the *Familiarity* and *Perceived Popularity*. By investigating the impact of other factors, we can make inferences about the mediating effects of popularity.
We investigate relationships between the OSA and all other concepts except PC and INT since we already have shown a high relationship between those (see section 7.4.1).
We can observe only few direct impacts of the OSA on EXP and BI factors (see figure 7.16). *Popularity Lift* shows a small negative association with the Questionnaire *Choice Satisfaction* and *Choice Listening Behaviour*.
We found that the condition impacts the SSA *Perceived Popularity* and *Familiarity*. A similar effect of the OSA on the SSA can be observed when investigating the other popularity metrics (see figure 7.15). The *R: UPD* and the *R: Head Ratio* do not have a significant association with any other factors. *R: Median Popularity*, *R: Mean Popularity* and *R: Mid Ratio* all have a weak to moderate positive association with *Familiarity* and *Perceived Popularity*. *R: Popularity Lift* has a weak positive correlation to *Perceived Popularity*, but no correlation with *Familiarity*. These effects explain that users can perceive popularity since they recognize playlists with higher popularity and more mid items, which are more popular than tail items. Items with high popularity are more easily recognized, thus they score higher in *Familiarity*. On the other hand, an increased number of tail items is negatively correlated with *Familiarity* and *Perceived Popularity* showing the opposite effect. More tail items lead to less known items and the playlists are rated less popular.
Although popularity does not seem to have a direct impact on the users' perception and experience, we observe that a positive popularity lift impacts the satisfaction with the user's choices and their intention to listen to them in the future negatively. Since the correlations are very low, we plot the *R: Popularity Lift* against the questionnaire metrics in the appendix in figure C.4 for qualitatively analysing their relationship.
In conclusion, we can conclude that users can perceive increased popularity, caused by the different algorithms, in terms of recognizing them (perceiving them as familiar) and detecting their popularity. Interestingly, we did not find any effect of user popularity deviation. We could not find effects that the decreased user deviation popularity impacts the perception of the user.

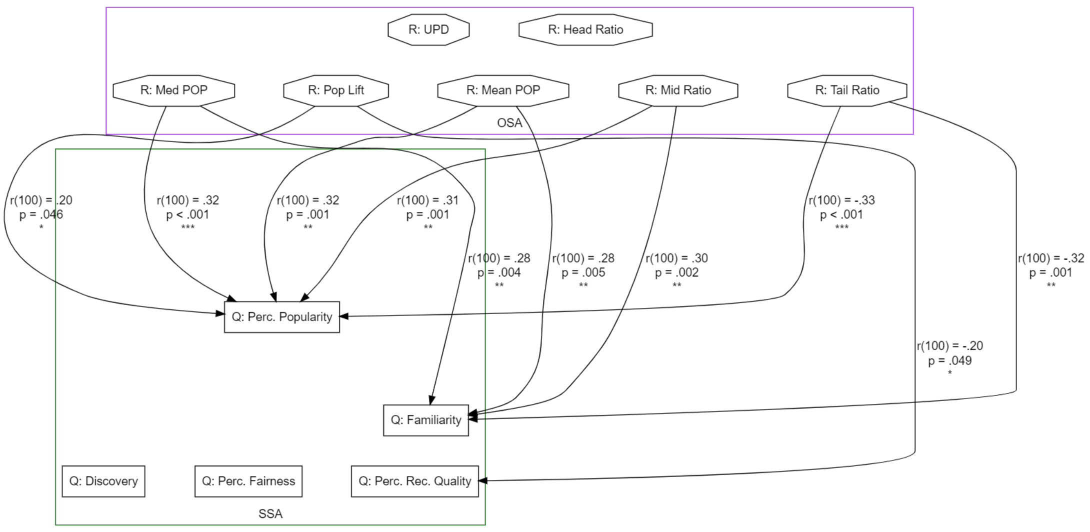


Figure 7.15: Effects of the OSA on SSA factors

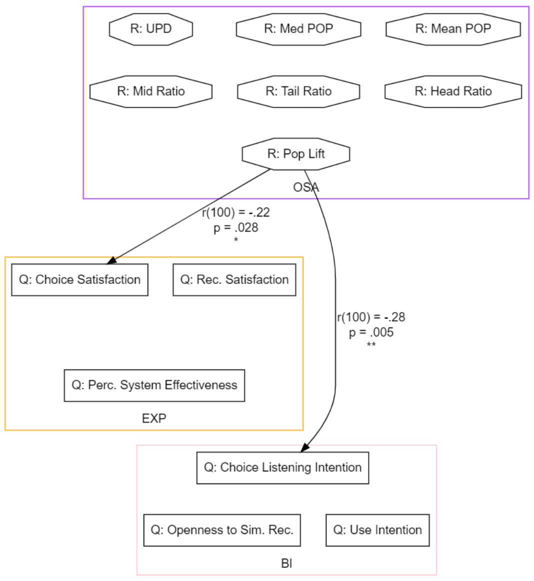


Figure 7.16: Effects of the OSA on EXP and BI factors

7.4.3.2 SSA as Mediators of algorithmic effects

We have explored how the SSA are impacted by conditions and other algorithmic factors, like the *Popularity Lift* or the *Mean Popularity*. Following this, we will explore their effect on other factors. We observe correlations between the SSA and other EXP and BI factors, fully grasping the experience of the users. Figure 7.17 shows the existing associations between factors in the SSA. We explored that *Familiarity* and

Perceived Popularity are directly impacted by the condition and the algorithmic effects. A relationship between those factors can be seen as well. The factors are strongly positively associated, indicating that familiar songs appear more popular and vice versa. Interestingly, perceived popularity does not interact with any other factors within the SSA. However, *Discovery* has a moderate negative relationship with *Familiarity*, indicating that familiar songs lead to less opportunity for discovering new songs and refining the user's musical taste. Furthermore, *Discovery* has a strong relationship with *Perceived Recommendation Quality* and a low relationship with *Perceived Fairness*. *Perceived Fairness* and *Perceived Recommendation Quality* have a moderate positive relationship. We do not define directional effects between factors within one concept; therefore, it is not clear whether increased discovery leads to higher perceived quality or whether the higher quality enables users to discover new music.
Nevertheless, the associations between factors can generate insights when starting from the condition. The different algorithms influence the perceived popularity and familiarity with the songs in the playlists, mediated by algorithmic effects, like the ratio of mid and tail items and the popularity of the songs in the playlists. Lower popularity, for example by using the `FA*IR` mitigation algorithm leads to less familiarity, which can influence the discovery of music positively. High *Discovery* is closely related to perceiving the quality of the recommendations positively and moderately related to perceiving them as fair. While not having a direct impact on those factors, the correlations between the factors indicate an impact of the condition, moderated by *Familiarity* and *Discovery*.

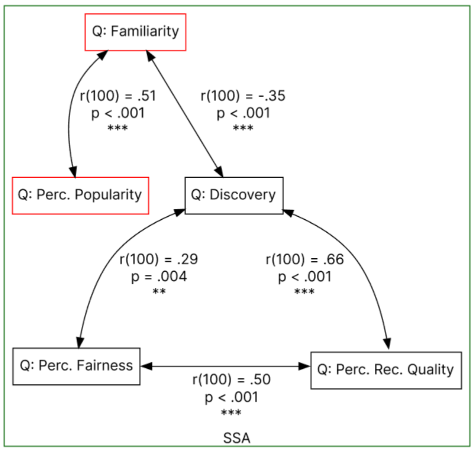
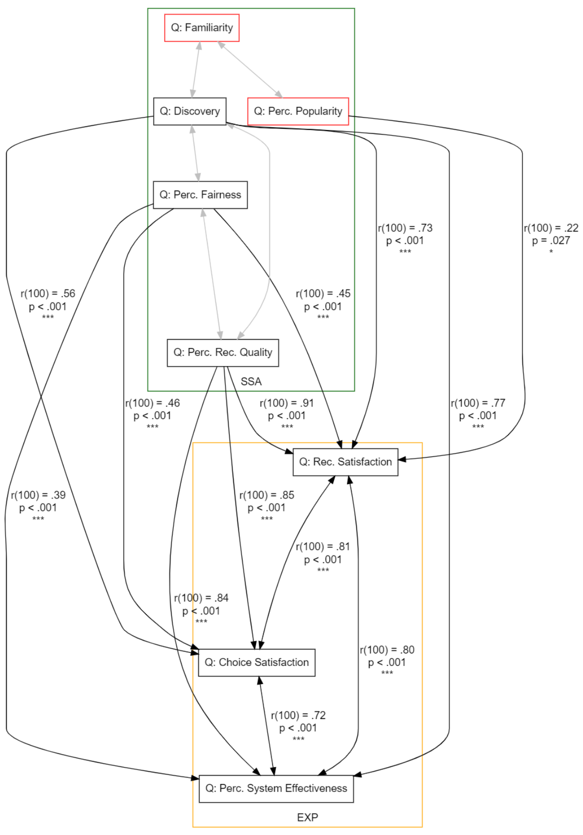


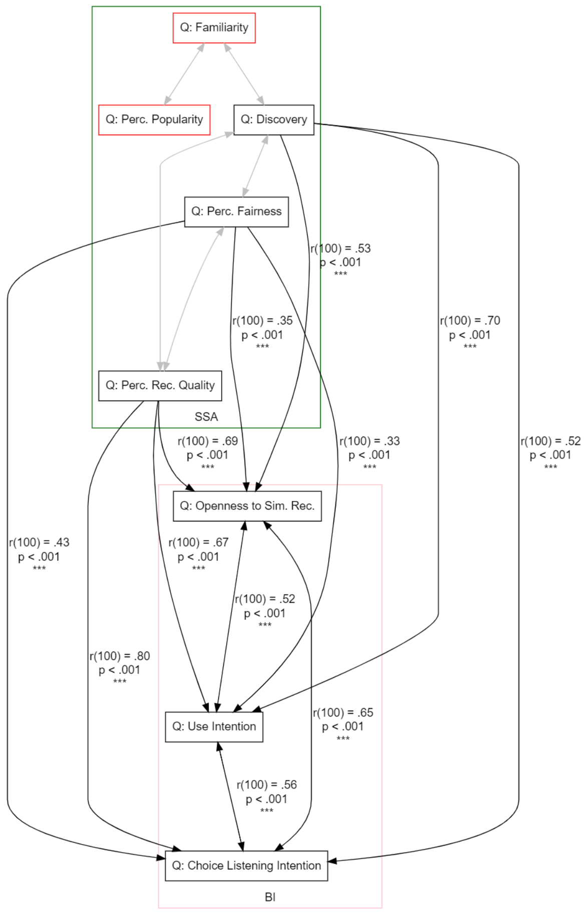
Figure 7.17: Effects between factors in the SSA
The red boxes indicate factors that are directly impacted by the conditions and OSA.

To further investigate the impact of those factors, we observe the impact of the SSA on EXP and BI factors. Regarding the EXP factors (see figure 7.18a), one important factor that can be observed is a high correlation between the factors within the concept. the factors *Recommendation Satisfactionm Choice Satisfaction* and *Perceived System Effectiveness* all show very strong associations ($r > 0.7$) between each other. *Perceived Fairness* has moderate relationships to all EXP factors while *Discovery* shows strong relationships to all factors and *Perceived Recommendation Quality* correlates very strongly to the factors and has even an almost perfect correlation with *Recommendation Satisfaction*. Additionally, we find that *Perceived Popularity* is weakly positively correlated with *Recommendation Satisfaction*.
A similar observation can be made regarding the BI factors (see figure 7.18b). All factors within the concept show strong correlations with each other and they are all positively impacted by the previously

discussed SSA. *Perceived Fairness* has moderate correlations to all BI metrics, *Discovery* has strong correlations to all of them, and *Perceived Recommendation Quality* has strong or very strong relationships to those factors.
Finally, when observing the relationships between EXP and BI (see figure 7.19), additional effects can be seen. All factors have significant strong to very strong linear relationships. Therefore, we assume that the experience of the users impacts their behavioural intentions positively. When users are highly satisfied with their recommendations, choices or the effectiveness in providing fitting items, users are more likely to express intentions to use the system further in the future and listen to similar items and their choices. One important factor is that between the three SSA factors *Discovery*, *Perceived Fairness* and *Perceived Recommendation Quality* and all factors in the EXP concept and all factors in the BI concept significant relationships are present, indicating an association between each of those 9 factors. The SSA factors impact the responses in the EXP questionnaires and the BI questionnaires. Since the EXP factors also have a positive relationship with the BI factors, these mediate the effect between the SSA and BI factors.


Q: Familiarity
Q: Discovery
Q: Perc. Popularity
Q: Perc. Fairness
Q: Perc. Rec. Quality
SSA
Q: Rec. Satisfaction
Q: Choice Satisfaction
Q: Perc. System Effectiveness
EXP
r(100) = .73 p < .001 ***
r(100) = .22 p = .027 *
r(100) = .56 p < .001 ***
r(100) = .45 p < .001 ***
r(100) = .46 p < .001 ***
r(100) = .91 p < .001 ***
r(100) = .77 p < .001 ***
r(100) = .39 p < .001 ***
r(100) = .85 p < .001 ***
r(100) = .81 p < .001 ***
r(100) = .84 p < .001 ***
r(100) = .80 p < .001 ***
r(100) = .72 p < .001 ***

(a) Effects of SSA on EXP factors


Q: Familiarity
Q: Perc. Popularity
Q: Discovery
Q: Perc. Fairness
Q: Perc. Rec. Quality
SSA
r(100) = .53 p < .001 ***
r(100) = .35 p < .001 ***
r(100) = .70 p < .001 ***
r(100) = .69 p < .001 ***
r(100) = .33 p < .001 ***
r(100) = .52 p < .001 ***
Q: Openness to Sim. Rec.
r(100) = .43 p < .001 ***
r(100) = .67 p < .001 ***
r(100) = .52 p < .001 ***
r(100) = .80 p < .001 ***
Q: Use Intention
r(100) = .65 p < .001 ***
r(100) = .56 p < .001 ***
Q: Choice Listening Intention
BI

(b) Effects of SSA on BI factors

Figure 7.18: User-centred Metrics
The red boxes indicate factors that are directly impacted by the conditions and OSA.
For clarity reasons, the correlations between the factors in the SSA concept are not displayed. They can be seen in figure 7.17

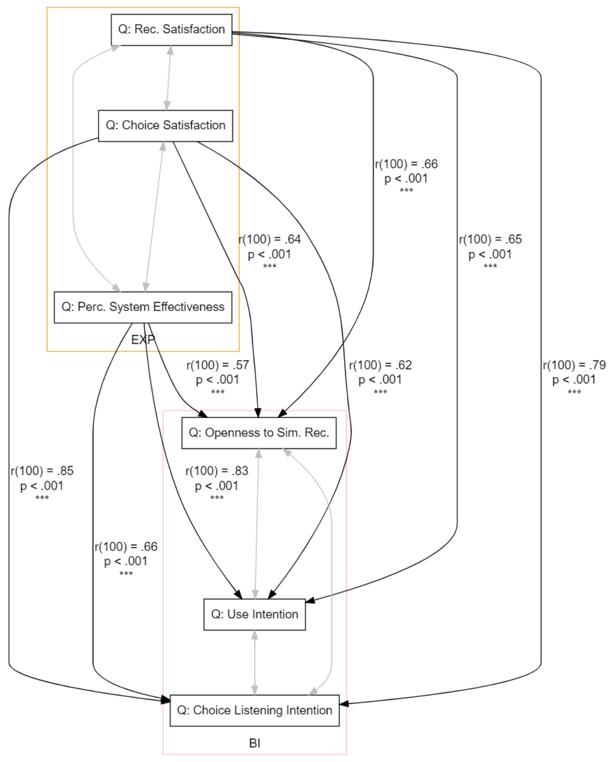

Q: Rec. Satisfaction
Q: Choice Satisfaction
Q: Perc. System Effectiveness
EXP
r(100) = .66 p < .001 ***
r(100) = .64 p < .001 ***
r(100) = .65 p < .001 ***
r(100) = .57 p < .001 ***
r(100) = .62 p < .001 ***
r(100) = .79 p < .001 ***
Q: Openness to Sim. Rec.
r(100) = .85 p < .001 ***
r(100) = .83 p < .001 ***
r(100) = .66 p < .001 ***
Q: Use Intention
Q: Choice Listening Intention
BI

Figure 7.19: Effects of EXP on BI factors
For clarity reasons, the correlations between factors within one concept are not displayed. They can be seen in figure 7.18a and 7.18b.

7.4.4 Personal Characteristics

Furthermore, we evaluate the impact of the personal characteristics next to algorithmic effects. We already explained that a higher popularity in profiles indicates a higher popularity in recommendations and choices among other relationships. Furthermore, we can find various weak correlations between personal characteristics and SSA, EXP, and BI factors. Those include, for instance, a relationship between the tail ratio in the user profiles and the *Recommendation Satisfaction* ($r(100) = -.22, p = .028$) or a relationship between the mean popularity in the user profile and *Choice Listening Intention* ($r(100) = .22, p = .026$. Since all correlations are weak, we refrain from presenting them in a graph and present them in table C.1 in the appendix instead. As shown in section 7.4.1, *Musical Sophistication* and *Musical Engagement* explain some effects of the algorithmic personal characteristics like the popularity of songs in the user profile. Furthermore, *Musical Sophistication* indicates higher scores in an increased *Use Intention* ($r(100) = .24, p = .015$). *Musical Engagement* is negatively associated with *Perceived Popularity* ($r = -0.24, p = .014$). Finally, various weak to moderate correlations exist between the *Musical Engagement* and *Musical Sophistication* questionnaire scores and INT factors. It has to be considered that those effects could be impacted by the relationship between the *Musical Sophistication* and *Musical Engagement* questionnaires and the popularity in the user profiles (see figure 7.3).

7.4.5 Interaction Data

While disregarding algorithmic effects between INT and PC as well as OSA, some effects on the interaction data can be seen (see table C.2). A Majority of the effects are weak correlation and some are moderate. Most effects seem to be related to the algorithmic effects between the INT factors and the OSA. For example, a negative association between *Familiarity* and *C: Tail Ratio* ($r(100) = -.32, p = .001$) could be explained by the association between *Familiarity* and the *Tail Ratio* in the recommendations. The *Tail Ratio* between the recommendations and choices is highly associated. Since the *R: Tail Ratio* is related to *Familiarity*, the effect can also be seen in the choices of the user. Most effects seem to relate to this.
One interesting factor is the *Popularity Lift* in the choices of the users. *Popularity Lift* in the recommendations (see figure 7.15 and 7.16) had only impact on *Perceived Popularity*, *Perceived Recommendation Quality*, *Choice Satisfaction* and *Choice Listening Intention*. The *C: Popularity Lift* is associated with various factors. Popularity lift of the choices is negatively associated with *Discovery* ($r(100) = -.26, p = .007$) and *Perceived Recommendation Quality* ($r(100) = -.30, p = .002$). Furthermore, it is negatively associated with *Recommendation Satisfaction* ($r(100) = -.27, p = .006$), *Choice Satisfaction* ($r(100) = -.30, p = .002$) and *Perceived System Effectiveness* ($r(100) = -.21, p = .036$). Finally, it has weak negative relationships with *Openness to Similar Recommendations* ($r(100) = -.30, p = .036$) and *Choice Listening Intention* ($r(100) = -.34, p < .001$). This indicates the negative effects of *Popularity Lift* in the choices of the users on the perceptions and future intentions of the choices. While all correlations are weak to moderate, it is still surprising that *Popularity Lift* is the only popularity metric of the INT factors that impacts the experience and behavioural intentions. Because the correlations are also very low, we plot the *R: Popularity Lift* against the questionnaire metrics in the appendix in figure C.5 for qualitatively analysing their relationship to interpret those effects.

7.5 Qualitative Insights

To gather more insights into the users' experience, we analyse the open-ended questions from the questionnaire and the user answers from the interviews.

7.5.1 Post-Experiment Interviews

We conducted 5 interviews. The participants were interviewed within three days after they participated in the experiment. The interviews were performed online via Microsoft Teams. For the interviews, we followed the script in section 5.3.5. After asking those questions, users were informed about the goal of the study and asked for additional comments. The interviews were recorded and transcribed. Finally, the interviews were coded using NVivo 14.

7.5.1.1 Results

Interaction with the Study Tool

Generally, none of the interviewed users expressed major issues or bugs when using the system. Three out of five users responded that there were a few songs that were not playable. They mention that there were 5 up to (at most) 15 tracks that could not be played over all playlists.
When asked about their experience with the system, different approaches to exploring the playlists could be found. P2 expressed that they initially scrolled through the playlist and played songs that seemed interesting to them. All the other participants explained that they listened through the playlist sequentially, from rank 1 to rank 25. P4 scrolled through the playlist initially to *"see what the playlist consists of"* and then listened from top to bottom while skipping songs they already knew. All participants reported that they did not listen to the entire playlist in most cases. The participants reported that they typically got an impression from the first seconds that decided whether they wanted to skip or continue listening to the songs. P1 points out that they were more likely to listen to the whole sample when they knew the artist because they perceived it would be likelier that they would like the song. Additionally, they need a *"good hook, like a good beat or anything like that to really commit to listening to the song".*
When asked specifically about the impact of the exploration setting as expressed by the task, all participants responded that they did not perceive that they were influenced by the task. All participants pointed out that this would be the most natural interaction with a playlist, and how they would interact with a playlist on a streaming platform. P5 expressed that this *"is somehow the most obvious thing to do, to listen to everything chronologically and then think about "what's the best thing about it?"* while P4 said that they would *"probably end up making a playlist anyways"* if one gave them a list of recommendations. Similarly, P1 explained that they approached the interaction in the same manner as they do with recommendations on a streaming platform.

Choice Selection

Furthermore, we asked interviewees how they selected the 5 songs. All participants except P5 immediately selected the songs that they liked. P5 first listened to all samples and then went back to add the songs they liked best. While P3 expressed that they intuitively chose the songs that they would add to their

list, P1, P2, and P4 only added songs initially when they were very sure that they liked the songs a lot. P4 pointed out that they were considering other songs, but weren't sure until they listened to the rest of the playlist. Two out of the five participants specifically stated that they considered whether they would add the songs to their private playlists.
P1 stated that it was difficult for them to add songs that they didn't know because the 30-second preview was too short to evaluate the entire song. Therefore, if they heard a familiar song, they did tend to add it early on. P1: *"Oftentimes, the ones that I really liked were songs I already heard somewhere. They were not necessarily already in one of my playlists, I had on Spotify"*. Similarly, for P2, selecting familiar songs was easier for them because they knew what to expect. P5 explained that they mainly chose songs they already knew or those that *"you heard more often; they were on the radio more often"*. Nevertheless, they expressed that most of the songs they did not know were also many that didn't fit their preferences.
On the other hand, P3 ignored songs they already knew for their selection, and P4 skipped those entirely when listening to the playlist. They aimed to select songs they did not know.

Playlist Perception

When asked about their perception of the music in the playlists, P5 noted that the genre was *"relatively the same at times"* with a focus on pop music. Other genres they listened to were barely included in the recommendations. They mention that some well-known songs were included which they liked. In contrast, P3 annotates that some old songs were included that felt redundant, and P4 annotates that there were songs included that they *"listened to, almost religiously, 2 years ago"*. Furthermore, some songs did not match P4's preferences but matched the preferences of another person who also listens to music on their Spotify account.
Overall, P2 and P3 point out positively that many recommendations were new to them or quite obscure. P1 was *"happy to gain new impressions from it"* and P3 liked that it was a diverse mix that enabled them to select a colourful mix.
Interestingly, all participants perceived some differences between the playlists although not all of them were able to point out the exact differences. P1 perceived differences in quality and how easy it was for them to select songs. P5 stated that the first and third playlists had a lot of overlap while the second one differed more. Nevertheless, they were not able to point out in what terms they differed.
P2, P3, and P4 all explained that they perceived differences between the popularity of the songs in the playlists. For instance, P1 expressed that they didn't know the songs from the first playlists, the second one consisted of familiar songs they don't listen to much, and the third one consisted of very popular songs. Similarly, P3 perceived the first playlist to include less familiar songs, the second to include many well-known songs, and the third to be a mixture of those two playlists. Finally, P1 explained while knowing some music from the first playlist and perceiving it as well-known, they were surprised to not know many songs from the second playlist and explained that they feel like *"songs from the bottom half of albums"* that are less known.

Popularity and Fairness

As explained in the previous section, some interviewees were able to perceive differences in popularity between the playlists. Furthermore, we asked them about their definitions of popularity. They generally expressed that a song is popular if it is known by many other people. The participants mention regular exposure to radio (P1, P2, P3), public places like bars, clubs, restaurants and festivals (P1, P2, P5), social media (P4), or in their social environment (P2, P4). Additionally, P1 and P2 mention that some genres

are more popular than others.
P2, P3, and P4 perceived less popular recommendations positively because they were able to explore something new (P2, P4), or they made the recommendations feel more diverse and a good mix (P3). Nevertheless, all participants also expressed that popularity often expresses some degree of quality. For instance, P4 states that if *somebody has thousands or millions of listeners monthly, there's probably a good reason for that"*. Furthermore, P1 expresses that it *"is more likely that people like it because it's popular"* and P3 points out that *"popular artists are popular for a reason"*. P5 points out that they are more likely to a popular song or artist over a less popular one because it indicates quality to them.
Considering fairness, the interviewees considered that popularity might have positive effects but could also create a fairness problem for artists because of fewer chances for unpopular artists to get their music exposed. Nevertheless, no one of the participants states that they directly perceive fairness in music recommendations and that it doesn't impact their satisfaction with musical recommendations.
The interviewees also expressed different views regarding the promotion of unpopular artists to achieve fairer recommendations. P1 and P3 argue for the promotion of underrepresented content. P1 explains that it would *"definitely [be] fair [...] if [underrepresented artists] had a chance to get their music out and get it recommended to people who listen to similar music"*. They highlight the difficulties for new artists and express the need for them to get recommended. Similarly, P3 highlights that completely removing popular content would not be the solution, but that increasing the exposure of unpopular songs would be beneficial to reach fairness. On the other hand, P2, P3 and P5 express that they are generally in favour of promoting unpopular content, but they highlight that the recommendations could be less fitting. P2 and P5 argue for offering the user control over whether they want to listen to more underrepresented content. P4 also states that user control and transparency are important when designing recommender systems. They argue that it would be problematic to force the user to listen to *"small alternative artists when they want to listen to [...] "Shake it off" or whatever"*. Nonetheless, they also consider that user control would lead to it not being used by all users and leading to some users ending up in filter bubbles.

7.5.2 Remarks

After finishing the study, users were able to provide qualitative comments. They were asked whether there was "anything else you observed during this study or that you would like to mention". Participants used this free text field to provide feedback, point out issues and share their observations. After removing comments that did not add any insights (like comments saying "nothing"), 12 comments remain that were coded using Nvivo 14 and are analysed in this section.

Interaction with the Study Tool

Similarly to the interviews, two participants reported that some songs could not be played. Additionally, one participant reported that the playlists were very similar.

Questionnaires

Four participants mentioned difficulties in answering the questionnaires. Primarily, users perceived it as challenging to rate the entire playlist because of differences between the songs. While a high rating in some metrics would be accurate for some songs, this would not be fitting for some other songs. Therefore, one user reports that they often chose the neutral option in those cases. Furthermore, one participant points out that it was annoying to not know their progress when filling in their questionnaires.

Playlist Perception

Various users pointed out that they used the system to discover new songs. For example, one participant points out the opportunity to discover musical genres that they typically won't listen to and they highlight that the playlists did not have a limited range of song types in comparison to other music recommendation systems. Conversely, another participant argues that they typically have a broad musical taste consisting of many genres which was not reflected by the recommendations.
On the other hand, two participants pointed out that they did not like the majority of the songs. While one of them did not like a lot of the items, they were, nonetheless, able to discover many songs that they added to their playlist. However, the other participant reports mostly unfitting songs and comments that the study design does not mimic a discovery setting for them. Especially, they wonder if it would be better if they were able to listen to full playlists instead of 30-second snippets.
Furthermore, two users pointed out that many songs were older or represented their taste from a few years ago. For example, a participant points out that the system recommended songs that the participant liked but hadn't listened to in a while. One participant presumes that the system "primarily focused on works from a few years ago, with fewer selections from the latest music" and proposes recommending more recent songs to improve the system. Additionally, another user annotates that the playlist consisted of songs from artists that the user knew but not these songs in particular.

7.5.3 Findings

All interviews and various comments show exploration behaviour when asking about their interactions with the playlists, actively engaging and clicking through songs. Overall, the interviewees explained that their interaction with the application was similar to their behaviour in exploration sessions on streaming platforms. Some participants clearly stated that they think about songs they would like to add to their playlists while others more focus on their general liking of the song and rely on an intuitive perception. The comments focus on their opportunity to use the playlist to discover songs they did not know previously. In the comments, participants highlighted the limited time frame of the playlists, which did not consist of never songs.
Interestingly, all interviewees stated they perceived differences in the structure and properties of the playlists. Three were even able to identify that they had differences in terms of popularity or how well-known the songs are.
Regarding popularity, all users express that popularity indicates some kind of quality while also leading to unfair chances for less popular artists. When asking whether recommendations should be manipulated to include more unpopular content, two interviewees agreed to do so, while three highlighted that user control and not being forced to listen to less popular content is important and that it should be added as an additional feature.

Chapter 8

Discussion

The findings of the algorithmic analysis and the conducted user study have significant implications for the implementation of fairness-related algorithms, their feasibility and the perception of users. In section 4.5.4, we discussed the effectiveness and aspects of the mitigation algorithms. After adopting those and testing them with actual users, we can make inferences about their impact on the perceptions and satisfactions of the users to answer research questions 2 and 3 (see section 1).

8.1 Users and Recommendations

To begin with, we start by analysing the user profiles that we inferred from the participants' Spotify profiles and the recommendations created by the base algorithm as well as the recommendation strategies. Adapting the recommendation strategy relies on a workaround that creates user profiles based on Spotify recommendations (see section 6.9). Unfortunately, we observe a much higher ratio of tail items in the user profiles created by this workaround in comparison to the profiles in the original dataset. We attribute this to the structure of the original dataset, including the limited time frame of the dataset creation and the definition of popularity in this dataset (see section 4.5.5). The recommendation pipeline does not reflect the user profiles accurately if we assume the LFM-2b dataset, which was used for creating the base algorithm, is an accurate representation. For the used profile definition, a different definition of head and tail items might have been necessary. Based on the original definition (cf. section 3.1.1), head items and tail items should receive 20% of the interactions. In the user profiles of the participants, the tail items receive many more interactions, and the head items have less interaction (cf. figure 7.2). This definition influences the mitigation algorithms because they rely on it to re-rank the items. Additionally, this impacts the analysis negatively because a comparison to the algorithmic analysis might not be given. For example, no blockbuster-focused users were identified in the study since the ratio of head items was very small.
Nevertheless, we have concluded that the method of creating the user profiles reflects the users' preferences. This can, for example, be shown by the fact that all factors representing the popularity in the user profiles are related to the factor *Musical Sophistication* and half of the popularity metrics of the user profile are correlated with *Musical Engagement*. Users who show high *Musical Sophistication* or high *Musical Engagement* listen to fewer head items and have less popular songs in their profiles. Interestingly, high musical sophistication is related to a higher ratio of tail items while *Musical Engagement* does not show a significant relationship to the *Tail Ratio*. More research has to be done to investigate the impact of personal differences on the listening behaviour of users in terms of the popularity of the songs. Liang et al. [73] have shown that higher musical expertise is related to more diverse listening behaviour but also more consistent preferences for their top-listened artists.
Similarly to the algorithmic evaluation, we find little *Popularity Lift* in the base recommendations. We can also conclude that this effect was not found due to an over-representation of mid items in the recommendations while adding fewer tail and head items to the lists when creating recommendations. Figure 7.4

shows that while the *Mean Popularity* is similar between user profiles and recommendations, the *Median Popularity* is much higher for the base recommendations in comparison to the user profiles. A further observation of the distributions of the popularity categories shows that almost no tail items are recommended and that the ratio of head items is also reduced when compared to the user profiles. Those results indicate a high popularity bias because of the lack of tail items in the recommendations. Additionally, the over-representation of tail items in the user profiles confirms the presumption of a high popularity bias since their high ratio in the user profiles is not reflected in the recommendations. There are almost 40% tail items in the user profiles on average. The ratio of tail items in the base recommendations is nearly 0%. While we did not find a clear difference in popularity lift between user profiles and base recommendations, differences in this metric between the different recommendation strategies can provide insights into the effects of the mitigation strategy. The measurement *Popularity Lift* is crucial since it directly compares the user profiles with the recommendation lists.
The mitigation algorithms fulfil their respective goals and have a clear impact on the observed metrics. This is shown when investigating their impact on the metrics that measure their goals. The item-centred algorithm `FA*IR` massively reduces the popularity lift, indicating that the recommendations re-ranked by this algorithm reduce the overall popularity in the recommendation list, promoting less popular items. The user-centred mitigation algorithm `CP` creates a UPD of nearly 0, showing that the miscalibration of popularity between user profiles and recommendation lists is massively reduced. Since we chose to use high weights ($\lambda = 0.99$ for `CP` and $\alpha = 0.98$ for `FA*IR`), this effect was expected. High weights put more emphasis on reaching the mitigation goal than on achieving high performance. The condition shows clear effects on the popularity metrics, which were further analysed through paired t-tests. Based on the algorithmic evaluation, we presume a decreasing performance for `FA*IR` and almost no change in performance for `CP` (cf. figure 4.10). To investigate whether this effect impacts the users' perception and satisfaction, we discuss the questionnaire scores and interaction data.

8.2 User Perception and Satisfaction

As explained earlier (see section 7.4.1), the PC (defining the level of popularity in the user profiles), the OSA (measuring the popularity in the recommendations), and the INT factors (measuring the popularity in the choices made for the experimental task) are highly inter-correlated. Therefore, direct inferences from the condition and other OSA on the INT factors are not conclusive. For example, it can be observed that the ratio of tail items in the user's choices is higher when the ratio in the recommendations is higher. This is unsurprising because if the recommendations to choose from have specific attributes, those will reflect in the choices. Nonetheless, interesting conclusions can be drawn from the users' perceptions of these algorithmic factors as indicated by the questionnaire scores.

We want to show the impact of mitigation algorithms on the users' perception and experience. For this purpose, we defined the research question: *"To what degree does a mitigated popularity bias, created by different recommendation strategies, have an impact on the user perception and satisfaction with music recommendations in an exploration setting?"*. To further observe this, we investigate sub-questions. Initially, we observe the direct impact of the condition and the popularity metrics of the recommendations on the satisfaction metrics (EXP) and other metrics that indicate different perceptions of the recommendation lists (SSA) to answer RQ 2.1 (*"How does the manipulation of recommendation strategy (base algorithm, item-centred mitigation algorithm, user-centred mitigation algorithm) impact user perception of popularity and popularity lift in the recommended music, and how does this influence their satisfaction*

with the recommendations and choices?")
Our results show that the condition has no direct impact on any of the satisfaction metrics, leading to the presumption that generally no recommendation list was perceived more positively than any of the others. Furthermore, the OSA, which highlights differences in the popularity of the recommendations, are also not associated with experience factors. Our study did not find a direct impact of the popularity on the users' satisfaction and experience.
In terms of the general perception of the recommendation lists, the condition only clearly affects the two variables *Perceived Popularity* and *Familiarity*. Furthermore, differences can only be identified between the item-centred algorithm `FA*IR` and the other two conditions. There is no significant difference between the user-centred mitigation technique `CP` and the base recommendations on any questionnaire score. This could be explained by the apparent difference in popularity between `FA*IR` and the other two recommendation lists. Although base and `CP` still differ in mean and median popularity, the recommendation lists of `FA*IR` are less popular and include many more tail items. This effect is presumably mitigated by factors like median and mean popularity and the tail and mid ratio of the recommendation lists. We have shown that those popularity metrics are impacted by the condition, and they also correlate positively with *Perceived Popularity* and *Familiarity*. Those found correlations show that playlists showing higher levels of popularity are rated higher in *Perceived Popularity* and *Familiarity*, indicating that users can perceive varying levels of popularity, which was also confirmed by the interviews. Popularity is closely related to familiarity. Less popular songs are typically less well known, and therefore, rated as less familiar. This is consistent with results found by Graus et al. [43]. The authors show that popularity impacts *Perceived Popularity* significantly, and in the case of users with low musical engagement, it also impacts *Familiarity* significantly. Furthermore, they also find a high relationship between *Perceived Popularity* and *Familiarity*. The impact of the mitigation method `FA*IR` can also be highlighted when comparing their effect to the findings by Ferwerda et al. [39]. When presented with various recommendation lists by different recommendation algorithms with varying levels of popularity bias, the users were not able to perceive any difference in terms of popularity. This contrasts with our study, where we applied actual mitigation techniques and found that users were able to perceive those differences in terms of popularity. This difference highlights that employing an actual mitigation technique can influence the perception of the playlist. In another study, Lesota et al. [68] find that users can perceive varying degrees of popularity based on their population with mainstream titles. They find a relationship between *UPD* (in terms of *JSD*) and the perceived popularity and perceived per-track miscalibration. In our work, *UPD* did not correlate with any questionnaire score.
User Popularity Deviation(*UPD*) measures the difference between the ratio of the head, mid and tail items in the user profiles and the recommendation lists. A low deviation expresses high user fairness. Since the user-centred algorithm, `CP`, is mainly related to *UPD* and *UPD* did not correlate with any other questionnaire score, we conclude that `CP` does not have a direct impact on the perceptions of the users in comparison to the base recommendations. This can have various factors. We presume that the impact of the mitigation technique barely impacted the user perception because they had comparably many items in common as highlighted by the low average difference between the recommendation lists. Additionally, the mean and median popularity was quite similar between the base and `CP`. The differences in *UPD* did not impact the perception of the users in this study, but the reduced *Popularity Lift* caused by `FA*IR` impacted the users' perceptions. In the following, we can further observe that `CP` has no additional impact on the other factors, like SSA, EXP or BI factors. Since those factors are also not impacted by the *UPD*, which `CP` aims to improve, we conclude that the mitigation by the user-centred algorithm does not impact the perception and satisfaction of the users. Only `FA*IR` seems to impact the users' perceptions by reducing the popularity of the songs and adding tail items. This effect becomes especially interesting considering the goal of the mitigation technique. Abdollahpouri et al. [5] argue that popularity bias leads to unfairness for users because different users are unequally affected by it. This is measured as *User Popularity Deviation*. They argue that users with a higher *UPD* receive worse recommendations than those

with lower deviation because the recommendations fit their taste less well. We are not able to replicate this presumption, which was mainly validated by algorithmic evaluations, in our user study. *UPD* did not impact any other metrics. Similarly, Lesota et al. [68] explore that users can perceive miscalibration but they do not find a correlation to user preferences. Based on the results of our study and Lesota et al.'s findings, we argue that miscalibration alone does not influence user perception negatively.

Our results do not show a clear impact of mitigation techniques on the user's satisfaction. However, we have shown that the item-centred mitigation algorithm can impact subjective system aspects *Perceived Popularity* and *Familiarity* negatively. Differences in the objective popularity metrics are also perceived in terms of different ratings of those two SSA. To further observe whether these differences in perceptions impact the user experience, we observe their correlations to other SSA and experience factors. By doing so, we aim to answer RQ 2.2 (*"How do mediators, such as perceived popularity, perceived fairness, and perceived familiarity, contribute to the user perception and satisfaction with the music recommendation lists generated by the different recommendation strategies, and how do these factors interact with the recommendation strategy?"*)
Interestingly, our findings show that neither *Familiarity* nor *Perceived Popularity* have a direct relationship to any of the experience factors. In line with results by Graus et al. [43] and Ferwerda et al. [38], satisfaction is highly dependent on *Discovery.* Ferwerda argues that diversity only improves the attractiveness of playlists if it allows users to refine their musical taste, which can be defined as *Discovery.* Similarly, Graus found that *Discovery* is an important mediator for satisfaction. We find a negative relationship between *Familiarity* and *Discovery.* If users are less familiar with playlists - what can be caused by mitigation and reduced popularity - they perceive that they can refine and enrich their taste using those playlists; *Discovery* scores are increased. This further impacts the remaining SSA, namely *Perceived Recommendation Quality* and *Perceived Fairness*, positively. Those factors and *Discovery* are closely related. Particularly *Discovery* and *Perceived Recommendation Quality* have a strong positive relationship. Those three factors are also all associated with all experience factors. Particularly *Discovery* is highly correlated with all experience factors ($r > 0.5$). Overall, this highlights the importance of *Familiarity* and *Discovery* in our experiment.
In conclusion, different degrees of popularity are perceived by the users, but they do not have any direct effects on the users' satisfaction. Nonetheless, less popular songs are also perceived as less familiar. *Familiarity* has no direct impact on the experience factors, but *Discovery* mediates the perception and experience of the users. If low *Familarity* evokes *Discovery*, the feeling that the playlist enriches the user's taste, the perceived fairness, quality, effectiveness and satisfaction with the recommendation and choices are improved.
The importance of *Familiarity* is highlighted by the comments of the users. When asked about their choices (see section 7.3.1), users reported popularity as one factor they recognized changed, but their main criterion was how familiar they were with the songs. While being positive for some users, most users did perceive familiarity negatively because unknown songs were more exciting and enabled them to discover music. This is in line with Ferwerda et al.'s findings [38] where either *Familiarity* has a positive effect or diversity impacts satisfaction positively if it refines the user's taste by increasing *Discovery.* The rankings of the users (see section 7.3) show that the `FA*IR` playlist is the most polarizing one. It is barely ranked second but mainly first or last. This indicates that users either prefer the less popular and familiar recommendations or they leave a negative impact on the user experience. Base on the other hand is barely rated first while `CP` is equally distributed over all ranks.
The previous conclusions can lead to the presumption that the users that rank `FA*IR` first rate *Discovery* higher than those who rank it last. We found an indirect impact of reduced *Familiarity* on *Discovery* which is positively associated with satisfaction metrics. Therefore, it could be argued that liking the `FA*IR` playlist is dependent on the *Discovery* metric. This would correspond to the findings by [38]

who show that reduced *Familiarity* can evoke *Discovery* and, consequentially, have a positive impact on satisfaction. To test this presumption, an independent t-test was performed, but it did not show a significant difference ($t(29) = 0.7981, p = 0.4381$) in *Discovery* scores between users who ranked it first ($M = 0.633, SD = 0.980$) and those who ranked `FA*IR` last ($M = 0.217, SD = 1.303$).
In other works [38, 43], either low *Familiarity* has a positive effect by evoking *Discovery* (as found in our experiment) or high *Familiarity* impacts satisfaction positively directly. We propose that the second effect was not found in this study because a clear exploration setting was defined for the users. When asking the participants in the interviews about the experimental task, users did not report any direct impact of the task on their behaviour. Nonetheless, their reported behaviour matches a lean-in exploration session. Users explained to use the system similarly to features like "discover weekly" from Spotify, and they asked themselves whether they would like to add the songs to their playlists. Furthermore, interviewees focused on songs they did not know. This matches a lean-in exploration setting, which has typically the goal of selecting candidates for immediate or future listening by creating a playlist from recommendations [100], for instance. As proposed earlier, in those settings, popular and familiar recommendations might not be appreciated since they add no value if they are well-known. We argue, therefore, that we successfully created a lean-in exploration setting for the users, which led to *Familiarity* having less of a positive effect on users. We can identify the opposite effect, where reduced *Familiarity* can have a positive effect on evoking *Discovery*, which increases the satisfaction of the users.

Another interesting observation is the negative impact of *Popularity Lift* on EXP and BI factors. A weak negative correlation between *R: Popularity Lift* and *Choice Satisfaction* and *Choice Listening Intention* was found. Although the effects are weak ($-.3 < r < -.2$), it indicates that *Popularity Lift* impacts the satisfaction with the users' choices and their motivation to keep listening to those. This effect is amplified by observations made from the INT factors (see section 7.4.5). Our analysis found some correlations between INT factors and SSA and EXP factors. Most effects in this part of the analysis could presumably be explained by algorithmic effects. For example, *C: Mid Ratio* correlates highly with *R: Mid Ratio* ($r = .85, p < .001$), and *R: Mid Ratio* correlates with *Perceived Popularity* ($r = .31, p = .001$). Therefore, a correlation between *C: Mid Ratio* and *Perceived Popularity* can easily be explained. This derivation cannot be made for the correlations between *C: Popularity Lift* and the EXP and BI factors. There is not a clear correlation between the *Popularity Lift* in the recommendations and the factors the *Popularity Lift* in the choices correlates with. However, the results show weak or moderate negative correlations of *C: Popularity Lift* with all EXP factors and all BI factors except *Use Intention*. This highlights an interplay between popularity lift, particularly within the choices, and satisfaction of the users. As indicated by the negative correlations, users seem to be more satisfied with their choices when the recommendations have less popularity lift.
Furthermore, if the *Popularity Lift* in the choices is lower, users seem to be generally more satisfied with the recommendations. Those effects also relate to behavioural intentions. The positive effect of reduced *Popularity Lift* is reflected by the user's motivation to listen to their choices and to explore similar music. Reduced popularity is mainly evoked by `FA*IR`. Although `CP` reduces the *Popularity Lift* in comparison to the base recommendations slightly (see figure 7.5), `FA*IR` reduces *Popularity Lift* by more than 5 times as much on average. Both mitigation algorithms reach a negative *Popularity Lift* on average.
We argued before that *Popularity Lift* in this research area might not be the most suitable measurement for estimating popularity bias (see section 4.5.5). However, it facilitates a direct comparison between the *Mean Popularity* in the user profile and the recommendations. Consequentially, it highlights individual differences in the recommendations.
Overall, the found positive effects of the *Popularity Lift* on satisfaction and behavioural intentions indicate a relationship between popularity bias, satisfaction and behavioural intentions. Nonetheless, it has to be considered that the found correlations are mostly weak. A qualitative analysis (see figure C.4 and

C.5) facilitates the low effect. The negative correlation is barely detectable and is mainly highlighted by points with a high popularity lift that causes low questionnaire scores. In conclusion, the effects lead to the presumption that in the used setting, a negative popularity lift could be able to improve recommendations for user perception.

Additionally to the positive correlations between the three SSA *Perceived Fairness*, *Discovery*, and *Perceived Recommendation Quality* and the satisfaction metrics, an effect of *Perceived Popularity* on an EXP factor was found. A weak positive correlation between *Perceived Popularity* and *Recommendation Satisfaction* can be observed. This is unexpected since *Perceived Popularity* does not correlate with any other SSA except *Familiarity* or any other EXP or BI factor. This contradicts our presumptions and previous research. For instance, Graus and Ferwerda [43] found a significant negative relationship between *Perceived Popularity* and *Satisfaction*. Higher *Perceived Popularity* only had a positive effect if users were familiar with the recommendations. We did not find the positive effect of *Familiarity*. This effect is unexpected and cannot be explained by current research or any mediators in our analysis.

Finally, similarly to [39], we did not find a direct impact of the algorithms on *Perceived Fairness*. However, we also found an association with all of the satisfaction metrics. We followed Ferwerda's approach of assessing fairness by inquiring how balanced the playlists were perceived. Our results indicate that balanced perceived playlists are generally perceived more positively, which has a positive effect on the user experience.

8.3 Behavioural Intentions

After drawing conclusions from the conditions, objective popularity metrics and perceptions of the participants on their experience, we observe the effects on behavioural intentions. We want to investigate whether fairer metrics hold the potential to foster fairer listening by motivating users to continue listening to fair content. We cannot conclude the actual long-term behaviour and impact, but behavioural intentions hold the potential to make predictions about the users' future behaviour. By analysing the behavioural intentions, we aim at answering RQ 3: *"Does a mitigated popularity bias in an exploration setting lead to increased motivation for exploring long-tail music items and indicate potential changes in user behaviour towards fairer music consumption?"* Similarly to the relation between the conditions and the experience factors, no direct relationship between the conditions and the behavioural intentions could be identified. Additionally, *Perceived Popularity* and *Familiarity* have no significant relationship with any of the BI factors. Nonetheless, significant positive relationships exist between the three remaining SSA factors, all EXP factors and the BI factors. This indicates that if users are satisfied with the recommendations, users report intentions to use them in the future. Therefore, if the recommendation lists are perceived positively, future intentions might lead to long-term consumption behaviour of fairer content. Because we found no differences between the three conditions, we presume that users would use the recommender systems with the different algorithms similarly afterwards as long as users are satisfied with the recommendation lists.
Mitigation algorithms can be seen as nudging mechanisms in which the presentation of items is simply changed by re-ranking the lists. Although no clear effects were found in their study, Liang and Willemsen [72] presume a long-lasting effect of nudges. Based on our results, we aim to answer RQ 3.3 (*"Can*

the presence of mitigated popularity bias in the exploration setting indicate potential changes in user behaviour towards fairer music consumption, based on users' responses regarding their planned behaviour and intentions for future music consumption?") and propose that if the mitigation techniques are used in a long-term focused system, users would change their listening behaviour towards the goals of the mitigation. For instance, if the user would get their regular recommendations from an algorithm which promotes long-tail artists, they might continue to use it to the same degree as when the recommendations were not mitigated. Therefore, their organic consumption could also turn out to include more long-tail artists.
While our results lead to the presumption of positive effects, we have to consider that the results in this work were found in a lean-in exploration behaviour. The organic use of a recommender system would not always be in this setting and the impact of mitigation strategies might differ.

8.4 Personal Characteristics

In the previous sections and the algorithmic analysis, we discussed to what degree different users are impacted by the popularity bias. For instance, niche users tend to receive worse recommendations due to the popularity bias (c.f., [63, 64]). We measured various algorithmic factors of the user profiles (e.g., *Mean Popularity* and *Head, Mid, Tail Ratios*) and classified the users into user groups. Finally, we also analyse the *Musical Engagement* and *Musical Sophistication* of the users. We analyse those factors to investigate to what degree personal characteristics might impact the findings of this work (RQ 2.2, RQ 3.2).
To investigate whether differences are apparent between niche and diverse users, we conducted the pragmatic evaluation on subsets of only niche and only diverse users. The analysis did not lead to any clear differences in terms of the effects of the conditions and the correlation between the factors. Additionally, it has to be mentioned that the participant number for these subsets is very small ($N = 17$). Therefore, we omit reporting those results in this work. Nonetheless, we presume that some effects could be mediated by membership in certain user groups. For instance, Graus et al. [43] found that *Discovery* has different effects for high or low-engagement users. Additionally, we have shown that personal characteristics, like *Musical Sophistication*, *Musical Engagement*, or the popularity in the user profiles, can have an impact on the ratings of the users. For instance, *Musical Sophistication* is weakly positively correlated with *Use Intention*. Further analysis is required to grasp the mediating effects of those personal characteristics. Finally, no effects of the demographics, like Age and Gender, were found.

8.5 Main Findings

Overall, we conclude with eight main findings from our user study that can inform future research in recommender systems, specifically in the context of music recommender systems. Figure 8.1 visualises these results with a simplified overview of the results that were discussed in the previous chapter. It has to be noted that these results are drawn from a study aimed at the creation of a lean-in exploration setting. Considering the setting, situational factors and context of users of a recommender system is crucial. While informing the design and implementation of mitigation algorithms in Recommender Systems, other factors could impact the user experience differently in other settings. The main findings are:

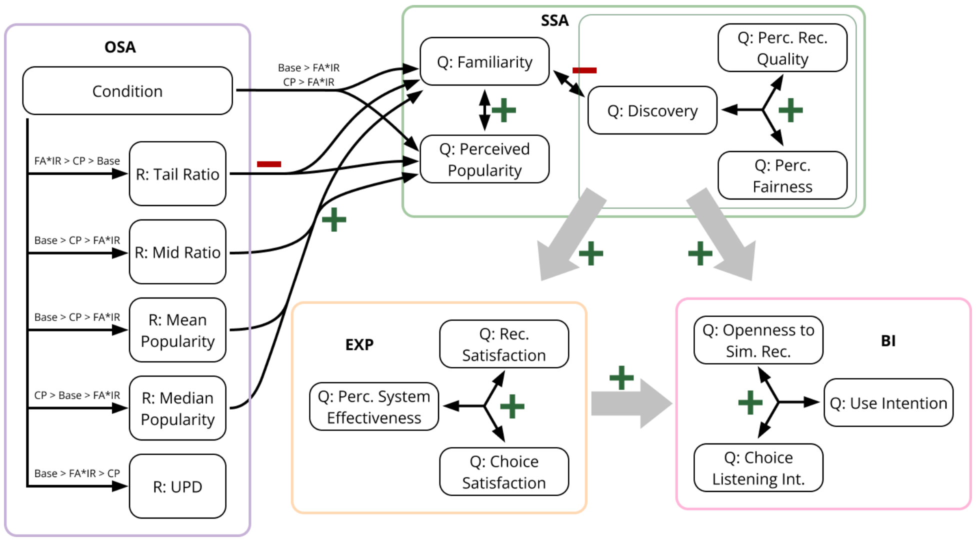


Figure 8.1: Simplified Overview of the main Effects found in the Study
Not all effects are displayed. Only OSA factors which have a significant correlation to any of the SSA, EXP or BI factors are shown. Most algorithmic effects are not shown and PC and INT factors are not displayed due to clarity reasons. Additionally, we filtered for effects with $r > .25$. Finally, we refrain from displaying the effects of *Popularity Lift* on the EXP and BI factors because of clearness reasons and their low general correlations ($.2 < r < .3$). Those effects are discussed in more detail in section 8.2.

1. The different algorithms and mitigation strategies showed no significant impact on the user experience in terms of satisfaction and effectiveness, indicating similar performance.
2. Only the item-centred mitigation algorithm (`FA*IR`) shows significant changes in the perception of the users in terms of *Perceived Popularity* and *Familiarity* in comparison to the base algorithms and the user-centred mitigation (`CP`).
3. Users can perceive differences in popularity by indicating varying levels of *Perceived Popularity* and *Familiarity*.
4. *UPD*, which `CP` aims to improve, is not perceived differently according to any measurement. This indicates that miscalibration does not directly impact how satisfied users are with their recommendations.
5. *Perceived Popularity* and *Familiarity* do not directly impact the experience but have a high relationship. That *Familiarity* does not have a positive impact on the user experience (in contrast to [38] and [43]) could be explained by the lean-in exploration setting of the experiment.
6. *Familiarity* is associated with *Discovery* and if *Discovery* is evoked, this impacts the SSA factors *Perceived Fairness*, *Perceived Recommendation Quality* and all experience factors (*Recommendation Satisfaction*, *Choice Satisfaction*, and *Recommendation Effectiveness*) positively.
7. Although no direct impact of the algorithms on behavioural intentions was found, a strong positive relationship between some of the SSA, the experience factors and the behavioural intentions was found, indicating that users are similarly likely to interact with playlists of different fairness levels, as long as they are satisfied with the recommendations.

8. Some indications for a negative effect of *Popularity Lift* in the recommendations and choices on EXP and BI factors can be found.

Chapter 9

Limitations and Future Work

This work investigated various aspects. We developed a recommendation algorithm in the first part and evaluated it on a large-scale data set LFM-2b. To our knowledge, there was no similar evaluation using a subset this large that investigated the popularity bias. Due to limited resources and the large scale of this set, we investigated some limitations in section 4.5.5. Future work could build up on these limitations, investigate different recommendation and mitigation algorithms, use different training parameters, and test larger models. Additionally, we used a fixed number of 5000 recommendations for the re-ranking. Future work should investigate different sizes of initial recommendations under observation of different mitigation strengths. The selected number of 5,000 recommendations is based on pre-tests to ensure a minimum amount of tail, mid and head items, and is not chosen based on a systematic approach. Finally, we discussed limitations of metrics, like popularity lift, which might not be suitable for a large number of items in the chosen dataset [64]. We propose that future work should further investigate metrics for quantifying the popularity bias in the music domain and to find robust metrics.

The limitations can be transferred to the recommendations generated in the user study. The same approach and model were used for generating the base recommendations and applying mitigation strategies. Therefore, uncertainties about factors, like the initial number of base recommendations, persist. Additionally, some other factors impact those recommendations, mainly due to the workarounds that had to be done to create the user profiles (see section 6.3). The effect of an over-representation of mid items in the Spotify recommendations can be observed, which is in contrast to the typically found popularity bias, which mainly recommends head items. Nonetheless, similar effects, like almost no exposure to long-tail items are still apparent. Since the profiles are created based on the recommendations with the top 50 songs as seed songs, we expected no exact recreation of the user profile and are aware of the limitations this approach brings with it. For instance, we found that many songs that were used as user profiles were not known by the user, as observed in the profile validation. Nonetheless, the users generally reported that the profile validation playlist matched their musical taste.
Additionally, we found that the boundaries of tail and head items defined on the LFM dataset might not work properly on the songs provided by users on Spotify. As shown in section 6.1.2, the LFM metric and Spotify's own popularity metric differ strongly. This might have various reasons: one important factor is the time frame. While Spotify considers the current popularity of a song, LFM relies on the number of overall interactions within the time of the creation of the dataset. Songs that are older will have more time to receive interactions, and no songs from after March 2020 are present. Since the experiments were conducted in August and September 2023, no songs which were released in the last three years are included in the dataset. In the interviews, users reported that many songs matched their taste from a few years ago. We explained previously (see section 2.3.1.2) that user preferences change over time. Therefore, the playlists included songs that the users do not like anymore. This is partly due to the limited time frame of the LFM-2b dataset but also due to the selected category of the seed songs for the creation of the profiles. We selected the top long-term tracks because they showed the largest number of available tracks in the Last.fm tracks. The Spotify API describes the songs in this category to be "calculated from several years of data and including all new data as it becomes available". Therefore, songs that are not listened

to extensively currently, but were very popular in the past still occur in this category, and similar songs are selected for the user profile. For future experiments, we propose to test the feasibility of medium-term and short-term top tracks for creating appropriate user profiles.
In the final step, the study procedure showed some issues that could be resolved in future works. To begin with, participants reported that some of the songs could not be played. Although this bug is distracting, the number of unavailable songs was still low, with a reported 3 to 5 per playlist consisting of 25 songs. Those samples should be filtered to avoid biasing the user experience. Additionally, users perceived the large number of questionnaires as exhausting. We acknowledge that the number of questionnaire items could be reduced, especially since we have shown high relationships between the EXP and BI metrics. Those could be reduced to fewer factors to avoid fatigue effects.
Furthermore, users reported in their comments difficulties in providing accurate questionnaire ratings due to the number of items and different perceptions of songs. This is a downside of the playlist-based evaluation approach that we used. This could be resolved by approaching a mixture of playlist-based and song-based evaluation, similarly to [39]. In their approach, each song is rated individually as well as the playlist as a whole. Nevertheless, we presume that this approach would lack the realism of a lean-in exploration setting. Future work should further explore the potential of different evaluation forms, their realism in comparison to actual settings and their effects.
Due to the low number of participants, a structural equation model was not feasible. Nonetheless, the pragmatic evaluation proposed in [61] has provided meaningful insights into the effects of popularity bias mitigation. Future studies should aim for a larger number of participants. We provided an extensive analysis of the results. Nonetheless, the high number of measurements allows further observation of the collected data. For instance, Graus and Ferwerda [43] have shown differences in the impact of *Discovery* between users with different levels of *Active Musical Engagement.* Creating Multigroup SEMs for different user groups could further improve the insights and show new effects. Another category of user groups that could be further explored is the different user groups based on their interest in popular items (Niche, Diverse, Blockbuster-focused). For example, we have shown that `FA*IR` was mainly ranked first or last while generally no effect of the conditions on any satisfaction metric was found. Further analysis could explore which factors, and presumably personal characteristics, influenced those ratings. We did an initial analysis by applying the pragmatic evaluation on subsets of Niche and Diverse users. This initial analysis did not provide any meaningful differences. It has to be considered that those analyses operate on small subsets. For an in-depth analysis of user subsets and their differences, more participants are needed. Therefore, we propose that future work should make use of crowd-sourcing (c.f. [38, 39, 43, 68]) to gather more participants to achieve more robust evaluations (e.g., SEMs) and to analyse user subsets further.
Although we argued that *Popularity Lift* is not an appropriate metric for measuring popularity bias, we found indications for an impact on various EXP and BI factors. Those effects are all weak or moderate. However, they highlight the importance of a metric that considers the difference between user profiles and recommendations. Future work should further explore appropriate metrics for measuring popularity lift on an individual level to explore these effects in more detail.
Furthermore, we tested two different types of popularity bias mitigation algorithms with different goals and directed towards certain actors. Both could be expanded by additional, more refined goals. For instance, `CP` aims for reduced miscalibration of the popularity of the songs in the user profile and in the recommendations. Steck [107] proposes that the distribution of recommendations should correspond to the distribution across the user profile. Steck explains this on the example of genre distributions of items and the miscalibration of those. Our work, like many of the other mentioned works (e.g., [5, 57, 64], focuses on the popularity distribution and miscalibration. Future work could, on the one hand, investigate the impact of popularity miscalibration mitigation on genre miscalibration and, on the other hand, add genre miscalibration mitigation as another goal of the algorithm to further improve the performance of the algorithms.

Additionally, this experiment defined a lean-in scenario for the users. Future work could investigate the effects of mitigation in different settings. For example, a lean-back scenario could be created by providing an additional task for the user and playing the music in the background.
Finally, we propose that fair recommendations have the potential to foster fairer listening behaviour of the users. We use metrics that measure the behavioural intentions of the users. Nonetheless, the actual behaviour of the users can only be measured with longitudinal studies. Liang and Willemsen [72] provided insights into the conduction of longitudinal experiments with a similar focus. We propose that a similar approach could be used to investigate the impact of popularity bias mitigation on the long-term behaviour of the users. This could be used to validate the behavioural intention metrics and to observe the changing perceptions and behaviour of the users. Furthermore, this enables insights into the behaviour of users in different settings since users are not exposed to one scenario. The longitudinal approach would observe the natural behaviour of the participants. Interesting questions arise, like whether users with different conditions consume the playlists in different settings and in certain scenarios.

In conclusion, this work lays the groundwork for future exploration of many observed factors and findings. We show that general improvements in the algorithms, the metrics and the adaption of the algorithms to the study tool could be accomplished. Furthermore, the findings provide insights and indicate various effects of popularity bias mitigation and its power that have to be validated and further explored.

Chapter 10

Conclusion

In conclusion, this thesis has shed light on the critical issue of popularity bias in recommender systems, particularly in the context of music recommendations. Recommender systems can be invaluable in guiding us through an overwhelming array of choices to present us with the most relevant items based on our specific needs, whether explicitly stated or inferred from implicit feedback. However, we acknowledged the critical issue of fairness and equity in recommender systems, as biases inherent in both data and algorithms can lead to biased recommendations that reinforce existing inequalities and result in unfair outcomes.
We explored the concept of popularity bias, which tends to favour well-known and popular content creators, potentially hindering the visibility of underrepresented or marginalized artists. Therefore, we argue for actively reducing its effects in algorithmic recommendations. Ultimately, we acknowledged that the impact of recommender systems on user behaviour and content consumption raises important questions. While it has been shown that users are generally satisfied with recommendations of less popular items, it remains uncertain whether these recommendations can lead to long-lasting changes in user behaviour. With this context in mind, we embarked on a comprehensive exploration to examine the effects of mitigation strategies for popularity bias in music recommendations. We conducted both algorithmic experiments and a user study to gain insights into the performance, user perceptions, and behavioural outcomes of these strategies. In the following, we will reflect on the insights gathered throughout this work and their implications for the field of recommender systems.

Through a comprehensive algorithmic experiment, we demonstrated that popularity bias is indeed prevalent in the base recommendation algorithm (`RankALS`), as evidenced by various metrics in an analysis of a large-scale music-related dataset. Although we were not able to find an over-representation of highly popular songs, we showed clear indications for biased recommendations, such as the lack of recommended long-tail items. Additionally, we argue that the popularity measurement and the accompanying mean popularity and popularity lift metric might not reflect popularity bias accurately.
Furthermore, we explored two distinct mitigation strategies – one focused on achieving user fairness (`CP`) and the other on item fairness (`FA*IR`). Our findings from the algorithmic experiment revealed that the user-centred algorithm effectively achieved user fairness, while the item-centred algorithm succeeded in promoting item fairness. However, it is worth noting that the item-centered algorithm experienced a decrease in performance compared to the base algorithm. This trade-off between fairness and performance is a crucial consideration for the practical implementation of mitigation strategies in recommender systems. To bridge the gap between algorithmic performance and user perception, we conducted a user study in a lean-in exploration setting, where participants actively engaged with playlists generated by different algorithms. Surprisingly, we found that none of the algorithms significantly outperformed the others in terms of user satisfaction, while achieving different levels of user and item fairness. This result suggests that the algorithmic performance differences between algorithms might not be perceptible by the users, highlighting the prospects of implementing mitigation algorithms as well as the importance of conducting user-centric studies to gauge the real-world impact of recommendation strategies.
While reduced *UPD*, achieved by the user-centred algorithm `CP`, does not impact the user's percep-

tion significantly, we find that users can perceive differences in popularity and familiarity, reached by the item-centred mitigation algorithm `FA*IR`. While prior research has suggested that familiarity with recommended items can positively influence user satisfaction [38, 43], our study did not replicate this finding, possibly due to the unique context of lean-in exploration. Instead, we found that less familiarity, as achieved by the item-centred algorithm, could enhance the discovery factor for users, leading to a more engaging music exploration and enrichment of their musical taste. High discovery leads to higher satisfaction with the recommendations and effectiveness of the system.

Crucially, our study found no significant differences in users' behavioural intentions across the different algorithms. This suggests that, regardless of the specific mitigation strategy employed, users were similarly motivated to engage with future recommendations and maintain their usage of the recommender system. This finding underscores the potential for item-centred algorithms to drive long-lasting changes in user behaviour if users are content with the recommendations provided. Additionally, it provides a strong groundwork for future work to investigate this tendency in more depth by applying longitudinal studies.

In summary, our research contributes to the ongoing discourse on fairness in recommender systems by addressing the relationship between algorithmic performance, user perception, and user behaviour. While achieving fairness in recommendations remains a complex challenge with trade-offs, our study suggests that users may not necessarily perceive these trade-offs and that item-centred algorithms can play a valuable role in promoting diverse and equitable content consumption. Additionally, the reduction of familiarity might lead to positive effects if it evokes the user's discovery.

As recommender systems continue to play a relevant role in shaping user preferences and content consumption, it is crucial to reach a balance between personalisation, fairness, and user satisfaction. While this work focuses on lean-in exploration music consumption sessions, future research in this field should further explore the long-term effects of mitigation strategies on user behaviour and investigate the interplay between familiarity, popularity, discovery, and satisfaction in different recommendation contexts.

Ultimately, the goal of recommender systems should not only be to provide personalised content but also to enable users to discover new and diverse experiences, thereby fostering a fairer ecosystem for all. In this work, we have provided important insights into the feasibility of achieving fairness in music recommender systems while ensuring similarly high-quality user experiences.

Bibliography

[1] Himan Abdollahpouri. “Popularity bias in recommendation: A multi-stakeholder perspective”. PhD thesis. University of Colorado at Boulder, 2020.

[2] Himan Abdollahpouri and Robin Burke. “Multistakeholder recommender systems”. In: *Recommender systems handbook*. Springer, 2021, pages 647–677.

[3] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. “Controlling popularity bias in learning-to-rank recommendation”. In: *Proceedings of the eleventh ACM conference on recommender systems*. 2017, pages 42–46.

[4] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. “Managing popularity bias in recommender systems with personalized re-ranking”. In: *arXiv preprint arXiv:1901.07555* (2019).

[5] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. “The unfairness of popularity bias in recommendation”. In: *arXiv preprint arXiv:1907.13286* (2019).

[6] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. “The connection between popularity bias, calibration, and fairness in recommendation”. In: *Proceedings of the 14th ACM Conference on Recommender Systems*. 2020, pages 726–731.

[7] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, Bamshad Mobasher, and Edward Malthouse. “User-centered evaluation of popularity bias in recommender systems”. In: *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization*. 2021, pages 119–129.

[8] Gediminas Adomavicius and Alexander Tuzhilin. “Context-aware recommender systems”. In: *Recommender systems handbook*. Springer, 2010, pages 217–253.

[9] Enrique Amigó, Yashar Deldjoo, Stefano Mizzaro, and Alejandro Bellogín. “A unifying and general account of fairness measurement in recommender systems”. In: *Information Processing & Management* 60.1 (2023), pages 103–115.

[10] Ashton Anderson, Lucas Maystre, Ian Anderson, Rishabh Mehrotra, and Mounia Lalmas. “Algorithmic effects on the diversity of consumption on spotify”. In: *Proceedings of The Web Conference 2020*. 2020, pages 2155–2165.

[11] Chris Anderson. *The long tail: Why the future of business is selling less of more*. Hachette UK, 2006.

[12] Arda Antikacioglu and R Ravi. “Post processing recommender systems for diversity”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pages 707–716.

[13] Christine Bauer, Marta Kholodylo, and Christine Strauss. “Music recommender systems challenges and opportunities for non-superstar artists”. In: (2017).

[14] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Anna Xambó, Emilia Gómez, and Perfecto Herrera. “Semantic audio content-based music recommendation and visualization based on user preference examples”. In: *Information Processing & Management* 49.1 (2013), pages 13–33.

[15] Robin Burke. “Multisided fairness for recommendation”. In: *arXiv preprint arXiv:1707.00093* (2017).

[16] Rocío Cañamares and Pablo Castells. "Should I follow the crowd? A probabilistic analysis of the effectiveness of popularity in recommender systems". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018, pages 415–424.

[17] Pablo Castells, Neil Hurley, and Saul Vargas. "Novelty and diversity in recommender systems". In: *Recommender systems handbook*. Springer, 2021, pages 603–646.

[18] Oscar Celma. *Music recommendation and discovery: The long tail, long fail, and long play in the digital music space*. Springer Science & Business Media, 2010.

[19] Òscar Celma and Pedro Cano. "From hits to niches? or how popular artists can bias music recommendation and discovery". In: *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*. 2008, pages 1–8.

[20] Òscar Celma and Perfecto Herrera. "A new approach to evaluating novel recommendations". In: *Proceedings of the 2008 ACM conference on Recommender systems*. 2008, pages 179–186.

[21] Junghoo Cho, Sourashis Roy, and Robert E Adams. "Page quality: In search of an unbiased web ranking". In: *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 2005, pages 551–562.

[22] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. "Performance of recommender algorithms on top-n recommendation tasks". In: *Proceedings of the fourth ACM conference on Recommender systems*. 2010, pages 39–46.

[23] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. "" More of an art than a science": Supporting the creation of playlists and mixes". In: (2006).

[24] Yashar Deldjoo, Dietmar Jannach, Alejandro Bellogin, Alessandro Diffonzo, and Dario Zanzonelli. "Fairness in Recommender Systems: Research Landscape and Future Directions". In: (2022).

[25] Mukund Deshpande and George Karypis. "Item-based top-n recommendation algorithms". In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pages 143–177.

[26] Karlijn Dinnissen and Christine Bauer. "Fairness in music recommender systems: A stakeholder-centered mini review". In: *Frontiers in big Data* 5 (2022).

[27] Karlijn Dinnissen and Christine Bauer. *Amplifying artists' voices: item provider perspectives on the influence and fairness of music streaming platforms: 31st ACM Conference on User Modeling, Adaptation and Personalization*. English. Other. 2023. DOI: 10.1145/3565472.3592960.

[28] Virginie Do and Nicolas Usunier. "Optimizing generalized Gini indices for fairness in rankings". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pages 737–747.

[29] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. "Fairness through awareness". In: *Proceedings of the 3rd innovations in theoretical computer science conference*. 2012, pages 214–226.

[30] Michael D Ekstrand, Anubrata Das, Robin Burke, and Fernando Diaz. "Fairness in recommender systems". In: *Recommender systems handbook*. Springer, 2012, pages 679–707.

[31] Michael D Ekstrand, Anubrata Das, Robin Burke, Fernando Diaz, et al. "Fairness in information access systems". In: *Foundations and Trends® in Information Retrieval* 16.1-2 (2022), pages 1–177.

[32] Michael D Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer D Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. "All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness". In: *Conference on fairness, accountability and transparency*. PMLR. 2018, pages 172–186.

[33] Michael D Ekstrand and Martijn C Willemsen. “Behaviorism is not enough: better recommendations through listening to users”. In: *Proceedings of the 10th ACM conference on recommender systems*. 2016, pages 221–224.

[34] Avriel Epps-Darling, Henriette Cramer, and Romain Takeo Bouyer. “Artist gender representation in music streaming.” In: *ISMIR*. 2020, pages 248–254.

[35] Marc Faddoul, Guillaume Chaslot, and Hany Farid. “A longitudinal analysis of YouTube’s promotion of conspiracy videos”. In: *arXiv preprint arXiv:2003.03318* (2020).

[36] Andres Ferraro, Xavier Serra, and Christine Bauer. “Break the loop: Gender imbalance in music recommenders”. In: *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*. 2021, pages 249–254.

[37] Andres Ferraro, Xavier Serra, and Christine Bauer. “What is fair? Exploring the artists’ perspective on the fairness of music streaming platforms”. In: *Human-Computer Interaction–INTERACT 2021: 18th IFIP TC 13 International Conference, Bari, Italy, August 30–September 3, 2021, Proceedings, Part II 18*. Springer. 2021, pages 562–584.

[38] Bruce Ferwerda, Mark P Graus, Andreu Vall, Marko Tkalcic, and Markus Schedl. “How item discovery enabled by diversity leads to increased recommendation list attractiveness”. In: *Proceedings of the Symposium on Applied Computing*. 2017, pages 1693–1696.

[39] Bruce Ferwerda, Eveline Ingesson, Michaela Berndl, and Markus Schedl. “I Don’t Care How Popular You Are! Investigating Popularity Bias in Music Recommendations from a User’s Perspective”. In: Mar. 2023, pages 357–361. DOI: 10.1145/3576840.3578287.

[40] Martin Fishbein and Icek Ajzen. “Belief, attitude, intention, and behavior: An introduction to theory and research”. In: (1977).

[41] Mustansar Ali Ghazanfar and Adam Prügel-Bennett. “Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems”. In: *Expert Systems with Applications* 41.7 (2014), pages 3261–3275.

[42] Andrew V Goldberg. “An efficient implementation of a scaling minimum-cost flow algorithm”. In: *Journal of algorithms* 22.1 (1997), pages 1–29.

[43] Mark P Graus and Bruce Ferwerda. “The Moderating Effect of Active Engagement on Appreciation of Popularity in Song Recommendations”. In: *Diversity, Divergence, Dialogue: 16th International Conference, iConference 2021, Beijing, China, March 17–31, 2021, Proceedings, Part I 16*. Springer. 2021, pages 364–374.

[44] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. “Librec: A java library for recommender systems.” In: *UMAP workshops*. Volume 4. Citeseer. 2015, pages 38–45.

[45] Christian Hansen, Rishabh Mehrotra, Casper Hansen, Brian Brost, Lucas Maystre, and Mounia Lalmas. “Shifting consumption towards diverse content on music streaming platforms”. In: *Proceedings of the 14th ACM international conference on web search and data mining*. 2021, pages 238–246.

[46] Haibo He and Edwardo A Garcia. “Learning from imbalanced data”. In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pages 1263–1284.

[47] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. “Lightgcn: Simplifying and powering graph convolution network for recommendation”. In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 2020, pages 639–648.

[48] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural collaborative filtering". In: *Proceedings of the 26th international conference on world wide web.* 2017, pages 173–182.

[49] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. "Evaluating collaborative filtering recommender systems". In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pages 5–53.

[50] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets". In: *2008 Eighth IEEE international conference on data mining.* Ieee. 2008, pages 263–272.

[51] *IFPI Global Music Report 2022.* Mar. 2023. URL: `https://globalmusicreport.ifpi.org/`.

[52] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. "What recommenders recommend: an analysis of recommendation biases and possible countermeasures". In: *User Modeling and User-Adapted Interaction* 25 (2015), pages 427–491.

[53] Mathias Jesse and Dietmar Jannach. "Digital nudging with recommender systems: Survey and future directions". In: *Computers in Human Behavior Reports* 3 (2021), page 100052.

[54] Yucheng Jin, Nyi Nyi Htun, Nava Tintarev, and Katrien Verbert. "Contextplay: Evaluating user control for context-aware music recommendation". In: *Proceedings of the 27th ACM conference on user modeling, adaptation and personalization.* 2019, pages 294–302.

[55] Yucheng Jin, Nava Tintarev, and Katrien Verbert. "Effects of personal characteristics on music recommender systems with different levels of controllability". In: *Proceedings of the 12th ACM Conference on Recommender Systems.* 2018, pages 13–21.

[56] Martijn Kagie, Michiel Van Wezel, and Patrick JF Groenen. *Map based visualization of product catalogs.* Springer, 2011.

[57] Anastasiia Klimashevskaia, Mehdi Elahi, Dietmar Jannach, Christoph Trattner, and Lars Skjærven. "Mitigating Popularity Bias in Recommendation: Potential and Limits of Calibration Approaches". In: *Advances in Bias and Fairness in Information Retrieval: Third International Workshop, BIAS 2022, Stavanger, Norway, April 10, 2022, Revised Selected Papers.* Springer. 2022, pages 82–90.

[58] Peter Knees and Markus Schedl. "A survey of music similarity and recommendation from music context data". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 10.1 (2013), pages 1–21.

[59] Bart P Knijnenburg and Martijn C Willemsen. "Evaluating recommender systems with user experiments". In: *Recommender systems handbook* (2015), pages 309–352.

[60] Bart P Knijnenburg, Martijn C Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. "Explaining the user experience of recommender systems". In: *User modeling and user-adapted interaction* 22 (2012), pages 441–504.

[61] Bart P Knijnenburg, Martijn C Willemsen, and Alfred Kobsa. "A pragmatic procedure to support the user-centric evaluation of recommender systems". In: *Proceedings of the fifth ACM conference on Recommender systems.* 2011, pages 321–324.

[62] Yehuda Koren, Steffen Rendle, and Robert Bell. "Advances in collaborative filtering". In: *Recommender systems handbook* (2021), pages 91–142.

[63] Dominik Kowald, Gregor Mayr, Markus Schedl, and Elisabeth Lex. "A Study on Accuracy, Miscalibration, and Popularity Bias in Recommendations". In: *arXiv preprint arXiv:2303.00400* (2023).

[64] Dominik Kowald, Markus Schedl, and Elisabeth Lex. "The unfairness of popularity bias in music recommendation: A reproducibility study". In: *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*. Springer. 2020, pages 35–42.

[65] Karl Krauth, Sarah Dean, Alex Zhao, Wenshuo Guo, Mihaela Curmei, Benjamin Recht, and Michael I Jordan. "Do offline metrics predict online performance in recommender systems?" In: *arXiv preprint arXiv:2011.07931* (2020).

[66] Jin Ha Lee, Liz Pritchard, and Chris Hubbles. "Can We Listen To It Together?: Factors Influencing Reception of Music Recommendations and Post-Recommendation Behavior." In: *ISMIR*. 2019, pages 663–669.

[67] Oleg Lesota, Stefan Brandl, Matthias Wenzel, Alessandro B Melchiorre, Elisabeth Lex, Navid Rekabsaz, and Markus Schedl. "Exploring Cross-group Discrepancies in Calibrated Popularity for Accuracy/Fairness Trade-off Optimization". In: *CEUR Workshop Proceedings*. Volume 3268. RWTH Aachen. 2022.

[68] Oleg Lesota, Gustavo Escobedo, Yashar Deldjoo, Bruce Ferwerda, Simone Kopeinik, Elisabeth Lex, Navid Rekabsaz, and Markus Schedl. "Computational Versus Perceived Popularity Miscalibration in Recommender Systems". In: (2023).

[69] Oleg Lesota, Alessandro Melchiorre, Navid Rekabsaz, Stefan Brandl, Dominik Kowald, Elisabeth Lex, and Markus Schedl. "Analyzing item popularity bias of music recommender systems: are different genders equally affected?" In: *Proceedings of the 15th ACM Conference on Recommender Systems*. 2021, pages 601–606.

[70] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, Juntao Tan, Shuchang Liu, and Yongfeng Zhang. "Fairness in recommendation: A survey". In: *arXiv preprint arXiv:2205.13619* (2022).

[71] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. "Variational autoencoders for collaborative filtering". In: *Proceedings of the 2018 world wide web conference*. 2018, pages 689–698.

[72] Yu Liang and Martijn C Willemsen. "Exploring the longitudinal effects of nudging on users' music genre exploration behavior and listening preferences". In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pages 3–13.

[73] Yu Liang and Martijn C Willemsen. "Promoting Music Exploration through Personalized Nudging in a Genre Exploration Recommender". In: *International Journal of Human–Computer Interaction* (2022), pages 1–24.

[74] Allen Lin, Jianling Wang, Ziwei Zhu, and James Caverlee. "Quantifying and mitigating popularity bias in conversational recommender systems". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pages 1238–1247.

[75] Siyi Liu and Yujia Zheng. "Long-tail session-based recommendation". In: *Proceedings of the 14th ACM Conference on Recommender Systems*. 2020, pages 509–514.

[76] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. "Content-based recommender systems: State of the art and trends". In: *Recommender systems handbook* (2011), pages 73–105.

[77] Alessandro B Melchiorre, Navid Rekabsaz, Emilia Parada-Cabaleiro, Stefan Brandl, Oleg Lesota, and Markus Schedl. "Investigating gender fairness of recommendation algorithms in the music domain". In: *Information Processing & Management* 58.5 (2021), page 102666.

[78] Robert K Merton. "The Matthew effect in science: The reward and communication systems of science are considered." In: *Science* 159.3810 (1968), pages 56–63.

[79] Shira Mitchell, Eric Potash, Solon Barocas, Alexander D'Amour, and Kristian Lum. "Algorithmic fairness: Choices, assumptions, and definitions". In: *Annual Review of Statistics and Its Application* 8 (2021), pages 141–163.

[80] Lillio Mok, Samuel F Way, Lucas Maystre, and Ashton Anderson. "The Dynamics of Exploration on Spotify". In: *Proceedings of the International AAAI Conference on Web and Social Media*. Volume 16. 2022, pages 663–674.

[81] Daniel Müllensiefen, Bruno Gingras, Jason Musil, and Lauren Stewart. "The musicality of non-musicians: An index for assessing musical sophistication in the general population". In: *PloS one* 9.2 (2014), e89642.

[82] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. "Exploring the filter bubble: the effect of using recommender systems on content diversity". In: *Proceedings of the 23rd international conference on World wide web*. 2014, pages 677–686.

[83] Xia Ning and George Karypis. "Slim: Sparse linear methods for top-n recommender systems". In: *2011 IEEE 11th international conference on data mining*. IEEE. 2011, pages 497–506.

[84] Ricardo S Oliveira, Caio Nóbrega, Leandro Balby Marinho, and Nazareno Andrade. "A Multi-objective Music Recommendation Approach for Aspect-Based Diversification." In: *ISMIR*. 2017, pages 414–420.

[85] Francois Pachet. "Knowledge management and musical metadata". In: *Idea Group* 12 (2005).

[86] Eli Pariser. *The filter bubble: What the Internet is hiding from you*. penguin UK, 2011.

[87] Yoon-Joo Park and Alexander Tuzhilin. "The long tail of recommender systems and how to leverage it". In: *Proceedings of the 2008 ACM conference on Recommender systems*. 2008, pages 11–18.

[88] Francesco Sanna Passino, Lucas Maystre, Dmitrii Moor, Ashton Anderson, and Mounia Lalmas. "Where To Next? A Dynamic Model of User Preferences." In: *WWW*. 2021, pages 3210–3220.

[89] Lorenzo Porcaro, Emilia Gómez, and Carlos Castillo. "Diversity in the Music Listening Experience: Insights from Focus Group Interviews". In: *ACM SIGIR Conference on Human Information Interaction and Retrieval*. 2022, pages 272–276.

[90] Pearl Pu, Li Chen, and Rong Hu. "A user-centric evaluation framework for recommender systems". In: *Proceedings of the fifth ACM conference on Recommender systems*. 2011, pages 157–164.

[91] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. "BPR: Bayesian personalized ranking from implicit feedback". In: *arXiv preprint arXiv:1205.2618* (2012).

[92] Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. "Revisiting the performance of ials on item recommendation benchmarks". In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pages 427–435.

[93] Paul Resnick and Hal R Varian. "Recommender systems". In: *Communications of the ACM* 40.3 (1997), pages 56–58.

[94] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook". In: *Recommender systems handbook*. Springer, 2010, pages 1–35.

[95] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender systems: Techniques, applications, and challenges". In: *Recommender Systems Handbook* (2021), pages 1–35.

[96] Elaine Rich. "User modeling via stereotypes". In: *Cognitive science* 3.4 (1979), pages 329–354.

[97] Kyle Robinson, Dan Brown, and Markus Schedl. "User Insights on Diversity in Music Recommendation Lists." In: *ISMIR*. 2020, pages 446–453.

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

[99] Markus Schedl, Stefan Brandl, Oleg Lesota, Emilia Parada-Cabaleiro, David Penz, and Navid Rekabsaz. “LFM-2B: a dataset of enriched music listening events for recommender systems research and fairness analysis”. In: *ACM SIGIR Conference on Human Information Interaction and Retrieval*. 2022, pages 337–341.

[100] Markus Schedl, Peter Knees, Brian McFee, and Dmitry Bogdanov. “Music recommendation systems: Techniques, use cases, and challenges”. In: *Recommender Systems Handbook*. Springer, 2021, pages 927–971.

[101] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. “Current challenges and visions in music recommender systems research”. In: *International Journal of Multimedia Information Retrieval* 7 (2018), pages 95–116.

[102] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. “Methods and metrics for cold-start recommendations”. In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. 2002, pages 253–260.

[103] Gunnar Schröder, Maik Thiele, and Wolfgang Lehner. “Setting goals and choosing metrics for recommender system evaluations”. In: *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*. Volume 23. 2011, page 53.

[104] Ayan Sinha, David F Gleich, and Karthik Ramani. “Deconvolving feedback loops in recommender systems”. In: *Advances in neural information processing systems* 29 (2016).

[105] Stacy L Smith, Karla Hernandez, and Katherine Pieper. “Inclusion in the recording studio? Gender and Race/Ethnicity of Artists, Songwriters & Producers across 1,00 Popular Songs from 2012-2021”. In: *Annenberg Inclusion Initiative* (2022).

[106] Nasim Sonboli, Jessie J Smith, Florencia Cabral Berenfus, Robin Burke, and Casey Fiesler. “Fairness and transparency in recommendation: The users’ perspective”. In: *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization*. 2021, pages 274–279.

[107] Harald Steck. “Calibrated recommendations”. In: *Proceedings of the 12th ACM conference on recommender systems*. 2018, pages 154–162.

[108] Gábor Takács and Domonkos Tikk. “Alternating least squares for personalized ranking”. In: *Proceedings of the sixth ACM conference on Recommender systems*. 2012, pages 83–90.

[109] Richard H Thaler and Cass R Sunstein. *Nudge: Improving decisions about health, wealth, and happiness*. Penguin, 2009.

[110] Dongjing Wang, Shuiguang Deng, and Guandong Xu. “Sequence-based context-aware music recommendation”. In: *Information Retrieval Journal* 21 (2018), pages 230–252.

[111] Yen-Yao Wang, Andy Luse, Anthony M Townsend, and Brian E Mennecke. “Understanding the moderating roles of types of recommender systems and products on customer behavioral intention to use recommender systems”. In: *Information Systems and e-Business Management* 13 (2015), pages 769–799.

[112] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. “A survey on the fairness of recommender systems”. In: *ACM Transactions on Information Systems* 41.3 (2023), pages 1–43.

[113] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. “A theoretical analysis of NDCG type ranking measures”. In: *Conference on learning theory*. PMLR. 2013, pages 25–54.

[114] Erika J Wolf, Kelly M Harrington, Shaunna L Clark, and Mark W Miller. “Sample size requirements for structural equation models: An evaluation of power, bias, and solution propriety”. In: *Educational and psychological measurement* 73.6 (2013), pages 913–934.

[115] Emre Yalcin and Alper Bilge. “Evaluating unfairness of popularity bias in recommender systems: A comprehensive user-centric analysis”. In: *Information Processing & Management* 59.6 (2022), page 103100.

[116] Emre Yalcin and Alper Bilge. “Popularity bias in personality perspective: An analysis of how personality traits expose individuals to the unfair recommendation”. In: *Concurrency and Computation: Practice and Experience* (2023), e7647.

[117] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. “Challenging the long tail recommendation”. In: *arXiv preprint arXiv:1205.6700* (2012).

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

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

[120] Guoshuai Zhao, Hao Fu, Ruihua Song, Tetsuya Sakai, Zhongxia Chen, Xing Xie, and Xueming Qian. “Personalized reason generation for explainable song recommendation”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.4 (2019), pages 1–21.

[121] Xiangyu Zhao, Zhendong Niu, and Wei Chen. “Opinion-based collaborative filtering to solve popularity bias in recommender systems”. In: *Database and Expert Systems Applications: 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26-29, 2013. Proceedings, Part II 24*. Springer. 2013, pages 426–433.

[122] Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. “Popularity-opportunity bias in collaborative filtering”. In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021, pages 85–93.

Appendix A

Algorithmic Evaluation

A.1 Algorithm Training

Algorithm	Factors	Iterations	Precision	Recall	MAP	NDCG
Popularity			0.1003	0.0050	0.0421	0.1071
Random			0.0005	0.0000	0.0001	0.0005
`RankALS`	16	5	0.0556	0.0035	0.0169	0.0541
`RankALS`	16	10	0.0660	0.0043	0.0208	0.0645
`RankALS`	16	15	0.0684	0.0044	0.0218	0.0669
`RankALS`	16	20	0.0694	0.0045	0.0222	0.0678
`RankALS`	16	25	0.0698	0.0046	0.0224	0.0683
`RankALS`	16	30	0.0701	0.0046	0.0225	0.0686
`RankALS`	32	5	0.0516	0.0036	0.0145	0.0510
`RankALS`	32	10	0.0682	0.0050	0.0209	0.0678
`RankALS`	32	15	0.0726	0.0055	0.0229	0.0725
`RankALS`	32	20	0.0739	0.0056	0.0235	0.0739
`RankALS`	32	25	0.0746	0.0057	0.0237	0.0745
`RankALS`	32	30	0.0749	0.0057	0.0238	0.0748
`RankALS`	64	5	0.0595	0.0045	0.0173	0.0601
`RankALS`	64	10	0.0794	0.0061	0.0252	0.0805
`RankALS`	64	15	0.0847	0.0067	0.0276	0.0859
`RankALS`	64	20	0.0867	0.0069	0.0284	0.0879
`RankALS`	64	25	0.0876	0.0071	0.0289	0.0888
`RankALS`	64	30	0.0882	0.0072	0.0291	0.0895
`RankALS`	128	5	0.0782	0.0058	0.0249	0.0804
`RankALS`	128	10	0.1062	0.0080	0.0376	0.1091
`RankALS`	128	20	0.1157	0.0091	0.0422	0.1188
`RankALS`	128	30	0.1199	0.0096	0.0443	0.1231

Table A.1: Performance Metrics for Recommendation Algorithms ($N = 25$)

Algorithm	Factors	Iterations	Mean	Median	Variance
Profiles			1246.2462	586.7665	4157377.1857
Popularity			14715.3600	13916.0000	3964984.4900
Random			76.7580	22.9100	71391.1462
`RankALS`	16	5	714.0656	641.8542	803547.1685
`RankALS`	16	10	669.1099	605.9578	676708.4344
`RankALS`	16	15	672.3719	610.3640	665682.6428
`RankALS`	16	20	675.4636	614.2027	660100.0710
`RankALS`	16	25	678.0218	617.0896	657888.8213
`RankALS`	16	30	679.7334	618.9180	660031.3565
`RankALS`	32	5	591.3215	541.7898	571973.7052
`RankALS`	32	10	585.7920	539.2923	515025.7687
`RankALS`	32	15	586.9938	540.3387	513690.8994
`RankALS`	32	20	587.2278	540.6709	512505.9949
`RankALS`	32	25	588.3590	541.4350	516350.7099
`RankALS`	32	30	588.9046	542.0339	516956.7071
`RankALS`	64	5	650.1144	605.6811	515165.7609
`RankALS`	64	10	716.2868	669.0393	567359.7737
`RankALS`	64	15	723.5864	672.8868	575109.4735
`RankALS`	64	20	721.6244	669.1680	572844.3716
`RankALS`	64	25	719.1256	665.6639	565824.9100
`RankALS`	64	30	715.9016	662.5137	559407.9468
`RankALS`	128	5	920.1371	868.6522	587605.2678
`RankALS`	128	10	1108.1338	1043.3267	782371.1020
`RankALS`	128	20	1141.0597	1069.8903	817573.6827
`RankALS`	128	30	1151.1276	1075.0083	816702.9953

Table A.2: Descriptive Popularity Metrics for Recommendation Algorithms ($N = 25$)

Algorithm	Factors	Iterations	Overall	Blockbuster-focused	Diverse	Niche
Popularity			14715.3600	14715.3600	14715.3600	14715.3600
Profiles			1246.2462	2450.3554	1168.0457	376.9976
Random			76.7580	76.8854	76.6314	77.0105
`RankALS`	16	5	714.0656	2119.9090	411.3499	216.3222
`RankALS`	16	10	669.1099	1873.0977	413.8330	230.9122
`RankALS`	16	15	672.3719	1864.5885	419.9010	237.5282
`RankALS`	16	20	675.4636	1867.5364	423.0002	240.7410
`RankALS`	16	25	678.0218	1874.1782	424.5488	242.2441
`RankALS`	16	30	679.7334	1879.0134	425.6427	242.6852
`RankALS`	32	5	591.3215	1582.8407	383.3385	223.7180
`RankALS`	32	10	585.7920	1472.9129	405.6645	239.0236
`RankALS`	32	15	586.9938	1456.4809	411.4893	243.9908
`RankALS`	32	20	587.2278	1448.9384	413.8844	245.5182
`RankALS`	32	25	588.3590	1450.2048	415.3992	245.3633
`RankALS`	32	30	588.9046	1451.8452	415.9323	244.8516
`RankALS`	64	5	650.1144	1499.3853	482.2465	304.4184
`RankALS`	64	10	716.2868	1587.2122	554.1331	331.7923
`RankALS`	64	15	723.5864	1571.4486	570.4945	334.9702
`RankALS`	64	20	721.6244	1550.4393	573.4283	337.3691
`RankALS`	64	25	719.1256	1534.3497	574.4865	337.7901
`RankALS`	64	30	715.9016	1521.0738	573.5529	337.7472
`RankALS`	128	5	920.1371	1826.8930	756.0361	505.6525
`RankALS`	128	10	1108.1338	2147.2844	942.8799	564.7076
`RankALS`	128	20	1141.0597	2154.5180	990.9849	577.7887
`RankALS`	128	30	1151.1276	2137.6048	1011.3664	583.8981

Table A.3: Group Average Popularity Metric for Recommendation Algorithms ($N = 25$)

Algorithm	Factors	Iterations	APTL	CTL	Agg-Div	Gini
Popularity			0.0000	0.0000	0.0000	1.0000
Profiles			0.1847	0.9364	0.9513	0.7350
Random			0.7644	0.7302	0.7303	0.4669
`RankALS`	16	5	0.0138	0.0076	0.1094	0.9672
`RankALS`	16	10	0.0097	0.0052	0.1054	0.9690
`RankALS`	16	15	0.0094	0.0048	0.1054	0.9689
`RankALS`	16	20	0.0094	0.0047	0.1056	0.9688
`RankALS`	16	25	0.0094	0.0048	0.1058	0.9687
`RankALS`	16	30	0.0095	0.0048	0.1059	0.9686
`RankALS`	32	5	0.0078	0.0062	0.1188	0.9631
`RankALS`	32	10	0.0078	0.0057	0.1196	0.9627
`RankALS`	32	15	0.0082	0.0058	0.1208	0.9620
`RankALS`	32	20	0.0083	0.0057	0.1210	0.9618
`RankALS`	32	25	0.0084	0.0058	0.1213	0.9617
`RankALS`	32	30	0.0085	0.0058	0.1214	0.9616
`RankALS`	64	5	0.0010	0.0012	0.1000	0.9696
`RankALS`	64	10	0.0019	0.0019	0.1045	0.9683
`RankALS`	64	15	0.0024	0.0023	0.1077	0.9670
`RankALS`	64	20	0.0026	0.0025	0.1090	0.9665
`RankALS`	64	25	0.0028	0.0025	0.1096	0.9662
`RankALS`	64	30	0.0028	0.0026	0.1100	0.9661
`RankALS`	128	5	0.0002	0.0003	0.0733	0.9790
`RankALS`	128	10	0.0007	0.0007	0.0783	0.9782
`RankALS`	128	20	0.0009	0.0010	0.0815	0.9773
`RankALS`	128	30	0.0011	0.0011	0.0831	0.9770

Table A.4: Exposure-Related Popularity Metrics for Recommendation Algorithms ($N = 25$)

Algorithm	Factors	Iterations	Overall	Blockbuster-focused	Diverse	Niche
Popularity Profiles			0.5920	0.3291	0.5828	0.8642
Random			0.3626	0.5396	0.3501	0.1979
`RankALS`	16	5	0.2316	0.2651	0.2325	0.1972
`RankALS`	16	10	0.2398	0.2751	0.2358	0.2086
`RankALS`	16	15	0.2405	0.2758	0.2357	0.2100
`RankALS`	16	20	0.2406	0.2757	0.2356	0.2105
`RankALS`	16	25	0.2405	0.2755	0.2355	0.2104
`RankALS`	16	30	0.2403	0.2752	0.2355	0.2103
`RankALS`	32	5	0.2514	0.2996	0.2378	0.2168
`RankALS`	32	10	0.2510	0.3018	0.2366	0.2145
`RankALS`	32	15	0.2489	0.2994	0.2357	0.2118
`RankALS`	32	20	0.2481	0.2982	0.2353	0.2109
`RankALS`	32	25	0.2477	0.2977	0.2349	0.2105
`RankALS`	32	30	0.2473	0.2968	0.2347	0.2103
`RankALS`	64	5	0.2647	0.3074	0.2417	0.2450
`RankALS`	64	10	0.2588	0.2978	0.2398	0.2389
`RankALS`	64	15	0.2550	0.2916	0.2382	0.2352
`RankALS`	64	20	0.2533	0.2891	0.2372	0.2337
`RankALS`	64	25	0.2523	0.2874	0.2367	0.2328
`RankALS`	64	30	0.2517	0.2864	0.2363	0.2323
`RankALS`	128	5	0.2472	0.2608	0.2310	0.2497
`RankALS`	128	10	0.2093	0.1923	0.1963	0.2394
`RankALS`	128	20	0.1876	0.1575	0.1728	0.2325
`RankALS`	128	30	0.1765	0.1403	0.1602	0.2289

Table A.5: User Popularity Deviation Metric for Recommendation Algorithms ($N = 25$)

Algorithm	Factors	Iterations	Overall	Blockbuster-focused	Diverse	Niche
Popularity Profiles			10.6090	5.0054	11.5983	38.0330
Random			-0.9394	-0.9686	-0.9344	-0.7957
RankALS	16	5	-0.4367	-0.1349	-0.6478	-0.4262
RankALS	16	10	-0.4721	-0.2356	-0.6457	-0.3875
RankALS	16	15	-0.4696	-0.2391	-0.6405	-0.3699
RankALS	16	20	-0.4671	-0.2379	-0.6379	-0.3614
RankALS	16	25	-0.4651	-0.2351	-0.6365	-0.3574
RankALS	16	30	-0.4638	-0.2332	-0.6356	-0.3563
RankALS	32	5	-0.5335	-0.3540	-0.6718	-0.4066
RankALS	32	10	-0.5379	-0.3989	-0.6527	-0.3660
RankALS	32	15	-0.5369	-0.4056	-0.6477	-0.3528
RankALS	32	20	-0.5367	-0.4087	-0.6457	-0.3488
RankALS	32	25	-0.5358	-0.4082	-0.6444	-0.3492
RankALS	32	30	-0.5354	-0.4075	-0.6439	-0.3505
RankALS	64	5	-0.4871	-0.3881	-0.5871	-0.1925
RankALS	64	10	-0.4349	-0.3523	-0.5256	-0.1199
RankALS	64	15	-0.4292	-0.3587	-0.5116	-0.1115
RankALS	64	20	-0.4307	-0.3673	-0.5091	-0.1051
RankALS	64	25	-0.4327	-0.3738	-0.5082	-0.1040
RankALS	64	30	-0.4352	-0.3792	-0.5090	-0.1041
RankALS	128	5	-0.2741	-0.2544	-0.3527	0.3413
RankALS	128	10	-0.1258	-0.1237	-0.1928	0.4979
RankALS	128	20	-0.0998	-0.1207	-0.1516	0.5326
RankALS	128	30	-0.0919	-0.1276	-0.1341	0.5488

Table A.6: Popularity Lift Metric for Recommendation Algorithms ($N = 25$)

A.2 Mitigation Stratgies

Algorithm	λ/p	Precision	Recall	MAP	NDCG
RankALS		0.1199	0.0096	0.0443	0.1231
Popularity		0.1003	0.0050	0.0421	0.1071
Random		0.0005	0.0000	0.0001	0.0005
FA*IR	0.02	0.1171	0.0090	0.0454	0.1189
FA*IR	0.10	0.1165	0.0091	0.0451	0.1184
FA*IR	0.20	0.1133	0.0087	0.0438	0.1159
FA*IR	0.30	0.1136	0.0090	0.0434	0.1160
FA*IR	0.40	0.1102	0.0085	0.0421	0.1135
FA*IR	0.50	0.1102	0.0085	0.0423	0.1137
FA*IR	0.60	0.1090	0.0085	0.0416	0.1127
FA*IR	0.70	0.1075	0.0084	0.0407	0.1116
FA*IR	0.80	0.1069	0.0082	0.0404	0.1111
FA*IR	0.90	0.1050	0.0089	0.0387	0.1094
FA*IR	0.98	0.0936	0.0077	0.0294	0.0919
XQ	0.00	0.1171	0.0090	0.0454	0.1189
XQ	0.10	0.1171	0.0090	0.0454	0.1189
XQ	0.20	0.1171	0.0090	0.0454	0.1189
XQ	0.30	0.1176	0.0091	0.0455	0.1193
XQ	0.40	0.1181	0.0091	0.0456	0.1197
XQ	0.50	0.1186	0.0092	0.0457	0.1201
XQ	0.60	0.1186	0.0092	0.0458	0.1202
XQ	0.70	0.1192	0.0095	0.0459	0.1208
XQ	0.80	0.1187	0.0094	0.0457	0.1202
XQ	0.90	0.1165	0.0093	0.0437	0.1173
XQ	1.00	0.1150	0.0092	0.0422	0.1151
CP	0.00	0.1171	0.0090	0.0454	0.1189
CP	0.10	0.1170	0.0090	0.0457	0.1193
CP	0.20	0.1168	0.0090	0.0458	0.1195
CP	0.30	0.1170	0.0091	0.0460	0.1198
CP	0.40	0.1170	0.0091	0.0460	0.1201
CP	0.50	0.1171	0.0091	0.0461	0.1205
CP	0.60	0.1178	0.0091	0.0461	0.1207
CP	0.70	0.1181	0.0091	0.0462	0.1209
CP	0.80	0.1194	0.0092	0.0464	0.1217
CP	0.90	0.1211	0.0092	0.0469	0.1230
CP	1.00	0.1181	0.0086	0.0443	0.1202

Table A.7: Performance Metrics for Mitigation Algorithms ($N = 5000, k = 25$)

Algorithm	λ/p	Mean	Median	Variance
Profiles		1246.2462	586.7665	4157377.1857
RankALS		1151.1276	1075.0083	816702.9953
Popularity		14715.3600	13916.0000	3964984.4900
Random		76.7580	22.9100	71391.1462
FA*IR	0.02	1288.9512	1186.9200	951894.4462
FA*IR	0.10	1264.1758	1173.5760	965352.9655
FA*IR	0.20	1241.4909	1158.3360	974343.2370
FA*IR	0.30	1221.8915	1138.7440	982521.8242
FA*IR	0.40	1187.0533	1112.7360	983520.1415
FA*IR	0.50	1187.0533	1112.7360	983520.1415
FA*IR	0.60	1170.7021	1096.9800	986656.9888
FA*IR	0.70	1155.2611	1084.0280	987934.3145
FA*IR	0.80	1140.0179	1066.7960	984868.5810
FA*IR	0.90	1100.2645	999.3360	985924.2526
FA*IR	0.98	961.9483	865.9920	914140.8436
XQ	0.00	1288.9512	1186.9200	951894.4462
XQ	0.10	1288.0240	1185.9120	952419.3422
XQ	0.20	1287.6909	1185.4760	952578.0318
XQ	0.30	1286.2477	1184.2840	953052.4104
XQ	0.40	1283.1286	1183.0080	952678.9437
XQ	0.50	1278.4954	1178.1720	952998.9860
XQ	0.60	1274.1877	1175.3040	954419.0510
XQ	0.70	1265.7973	1168.8440	955213.0707
XQ	0.80	1252.5776	1159.1680	961842.9231
XQ	0.90	1232.1234	1142.5560	971820.9426
XQ	1.00	1210.7534	1124.8440	978508.8103
CP	0.00	1288.9512	1186.9200	951894.4462
CP	0.10	1288.9266	1189.4880	960849.2414
CP	0.20	1290.3842	1189.5160	962807.6931
CP	0.30	1279.3554	1178.8400	995392.2280
CP	0.40	1281.1480	1180.3280	1005855.8967
CP	0.50	1286.7789	1184.7760	1028354.3617
CP	0.60	1285.4755	1186.0600	1024565.3071
CP	0.70	1282.8040	1184.1960	1080043.9288
CP	0.80	1283.1298	1177.0200	1146168.4466
CP	0.90	1276.6976	1171.0120	1346400.0220
CP	1.00	1235.1005	1098.2480	1553313.2382

Table A.8: Descriptive Popularity Metrics for Mitigation Algorithms ($N = 5000, k = 25$)

Algorithm	λ/p	Overall	Blockbuster-focused	Diverse	Niche
Profiles		1246.2462	2450.3554	1168.0457	376.9976
`RankALS`		1151.1276	2137.6048	1011.3664	583.8981
Popularity		14715.3600	14715.3600	14715.3600	14715.3600
Random		76.7580	76.8854	76.6314	77.0105
`CP`	0.00	1288.9512	2431.6911	1103.6949	574.5758
`CP`	0.10	1288.9266	2439.5385	1102.2316	570.1308
`CP`	0.20	1290.3842	2441.5244	1104.0351	569.9275
`CP`	0.30	1279.3554	2428.9393	1090.1349	569.5033
`CP`	0.40	1281.1480	2433.9156	1091.0692	570.3608
`CP`	0.50	1286.7789	2439.2170	1098.6205	570.4408
`CP`	0.60	1285.4755	2439.6504	1097.1124	567.8150
`CP`	0.70	1282.8040	2433.2785	1095.3105	566.6250
`CP`	0.80	1283.1298	2441.9052	1092.6219	566.9067
`CP`	0.90	1276.6976	2433.0726	1088.7465	555.2917
`CP`	1.00	1235.1005	2419.9896	1033.9735	522.2417
`FA*IR`	0.02	1288.9512	2431.6911	1103.6949	574.5758
`FA*IR`	0.10	1264.1758	2417.2778	1073.2854	555.5150
`FA*IR`	0.20	1241.4909	2406.1578	1045.3378	536.0458
`FA*IR`	0.30	1221.8915	2397.2652	1022.0557	515.7567
`FA*IR`	0.40	1187.0533	2386.2437	980.8927	473.6258
`FA*IR`	0.50	1187.0533	2386.2437	980.8927	473.6258
`FA*IR`	0.60	1170.7021	2381.5585	961.6497	453.0667
`FA*IR`	0.70	1155.2611	2375.3304	944.5776	432.2908
`FA*IR`	0.80	1140.0179	2371.5267	926.7886	412.0275
`FA*IR`	0.90	1100.2645	2361.0378	881.4205	356.6633
`FA*IR`	0.98	961.9483	2334.8711	729.5635	133.9300
XQ	0.00	1288.9512	2431.6911	1103.6949	574.5758
XQ	0.10	1288.0240	2431.6911	1103.6949	569.7467
XQ	0.20	1287.6909	2431.6911	1103.5976	568.3117
XQ	0.30	1286.2477	2431.5000	1103.3378	561.8108
XQ	0.40	1283.1286	2430.6215	1100.4059	555.5942
XQ	0.50	1278.4954	2429.6385	1096.3365	545.1158
XQ	0.60	1274.1877	2429.1022	1092.4370	535.3067
XQ	0.70	1265.7973	2427.4378	1083.7581	520.2392
XQ	0.80	1252.5776	2423.2363	1067.1995	507.1692
XQ	0.90	1232.1234	2415.0452	1041.7257	488.3958
XQ	1.00	1210.7534	2403.5993	1015.4089	471.1142

Table A.9: Group Average Popularity Metric for Mitigation Algorithms ($N = 5000, k = 25$)

Algorithm	λ/p	APTL	CTL	Agg-Div	Gini
Profiles		0.1847	0.9364	0.9513	0.7350
RankALS		0.0011	0.0011	0.0831	0.9770
Popularity		0.0000	0.0000	0.0000	1.0000
Random		0.7644	0.7302	0.7303	0.4669
FA*IR	0.02	0.0003	0.0000	0.0025	0.9977
FA*IR	0.10	0.0331	0.0001	0.0025	0.9977
FA*IR	0.20	0.0630	0.0002	0.0025	0.9977
FA*IR	0.30	0.0898	0.0003	0.0025	0.9977
FA*IR	0.40	0.1381	0.0005	0.0025	0.9977
FA*IR	0.50	0.1381	0.0005	0.0025	0.9977
FA*IR	0.60	0.1605	0.0006	0.0026	0.9976
FA*IR	0.70	0.1826	0.0006	0.0026	0.9976
FA*IR	0.80	0.2042	0.0007	0.0026	0.9976
FA*IR	0.90	0.2653	0.0009	0.0026	0.9976
FA*IR	0.98	0.4758	0.0017	0.0026	0.9976
XQ	0.00	0.0003	0.0000	0.0025	0.9977
XQ	0.10	0.0019	0.0000	0.0025	0.9977
XQ	0.20	0.0035	0.0000	0.0025	0.9977
XQ	0.30	0.0074	0.0000	0.0025	0.9977
XQ	0.40	0.0122	0.0000	0.0025	0.9977
XQ	0.50	0.0181	0.0001	0.0025	0.9977
XQ	0.60	0.0251	0.0001	0.0025	0.9977
XQ	0.70	0.0368	0.0001	0.0025	0.9977
XQ	0.80	0.0544	0.0002	0.0025	0.9977
XQ	0.90	0.0829	0.0003	0.0025	0.9977
XQ	1.00	0.1118	0.0004	0.0025	0.9977
CP	0.00	0.0003	0.0000	0.0025	0.9977
CP	0.10	0.0014	0.0000	0.0025	0.9977
CP	0.20	0.0016	0.0000	0.0025	0.9977
CP	0.30	0.0018	0.0000	0.0025	0.9977
CP	0.40	0.0018	0.0000	0.0025	0.9977
CP	0.50	0.0029	0.0000	0.0025	0.9978
CP	0.60	0.0038	0.0000	0.0025	0.9977
CP	0.70	0.0066	0.0000	0.0025	0.9978
CP	0.80	0.0123	0.0000	0.0025	0.9978
CP	0.90	0.0310	0.0001	0.0025	0.9978
CP	1.00	0.1032	0.0004	0.0025	0.9978

Table A.10: Exposure-Related Popularity Metrics for Mitigation Algorithms ($N = 5000, k = 25$)

Algorithm	λ/p	Overall	Blockbuster-focused	Diverse	Niche
RankALS		0.1765	0.1403	0.1602	0.2289
Popularity		0.5920	0.3291	0.5828	0.8642
Random		0.3626	0.5396	0.3501	0.1979
FA*IR	0.02	0.1703	0.1357	0.1710	0.2042
FA*IR	0.10	0.1250	0.1145	0.1207	0.1400
FA*IR	0.20	0.1065	0.1090	0.1020	0.1085
FA*IR	0.30	0.0950	0.1065	0.0914	0.0871
FA*IR	0.40	0.0835	0.1036	0.0842	0.0626
FA*IR	0.50	0.0835	0.1036	0.0842	0.0626
FA*IR	0.60	0.0813	0.1030	0.0843	0.0567
FA*IR	0.70	0.0805	0.1026	0.0858	0.0532
FA*IR	0.80	0.0818	0.1032	0.0880	0.0543
FA*IR	0.90	0.0900	0.1051	0.1004	0.0645
FA*IR	0.98	0.2218	0.1213	0.2249	0.3191
XQ	0.00	0.1703	0.1357	0.1710	0.2042
XQ	0.10	0.1688	0.1357	0.1710	0.1998
XQ	0.20	0.1643	0.1357	0.1704	0.1868
XQ	0.30	0.1554	0.1337	0.1698	0.1626
XQ	0.40	0.1480	0.1315	0.1656	0.1468
XQ	0.50	0.1404	0.1301	0.1610	0.1301
XQ	0.60	0.1305	0.1280	0.1559	0.1078
XQ	0.70	0.1173	0.1272	0.1451	0.0796
XQ	0.80	0.1011	0.1208	0.1245	0.0581
XQ	0.90	0.0836	0.1114	0.0979	0.0414
XQ	1.00	0.0750	0.1059	0.0846	0.0345
CP	0.00	0.1703	0.1357	0.1710	0.2042
CP	0.10	0.1679	0.1341	0.1690	0.2007
CP	0.20	0.1655	0.1306	0.1672	0.1985
CP	0.30	0.1625	0.1259	0.1631	0.1985
CP	0.40	0.1593	0.1228	0.1570	0.1982
CP	0.50	0.1505	0.1147	0.1508	0.1860
CP	0.60	0.1425	0.1051	0.1430	0.1795
CP	0.70	0.1316	0.0973	0.1325	0.1650
CP	0.80	0.1113	0.0814	0.1176	0.1350
CP	0.90	0.0761	0.0590	0.0813	0.0880
CP	1.00	0.0312	0.0353	0.0329	0.0253

Table A.11: User Popularity Deviation Metric for Mitigation Algorithms ($N = 5000, k = 25$)

Algorithm	λ/p	Overall	Blockbuster-focused	Diverse	Niche
RankALS		-0.0919	-0.1276	-0.1341	0.5488
Popularity		10.6090	5.0054	11.5983	38.0330
Random		-0.9394	-0.9686	-0.9344	-0.7957
FA*IR	0.02	0.0058	0.0193	-0.0569	0.4689
FA*IR	0.10	-0.0135	0.0132	-0.0829	0.4202
FA*IR	0.20	-0.0312	0.0086	-0.1068	0.3704
FA*IR	0.30	-0.0465	0.0048	-0.1267	0.3186
FA*IR	0.40	-0.0737	0.0002	-0.1618	0.2109
FA*IR	0.50	-0.0737	0.0002	-0.1618	0.2109
FA*IR	0.60	-0.0865	-0.0017	-0.1783	0.1583
FA*IR	0.70	-0.0985	-0.0043	-0.1929	0.1052
FA*IR	0.80	-0.1104	-0.0059	-0.2081	0.0534
FA*IR	0.90	-0.1414	-0.0103	-0.2468	-0.0882
FA*IR	0.98	-0.2494	-0.0213	-0.3766	-0.6576
XQ	0.00	0.0058	0.0193	-0.0569	0.4689
XQ	0.10	0.0051	0.0193	-0.0569	0.4566
XQ	0.20	0.0048	0.0193	-0.0570	0.4529
XQ	0.30	0.0037	0.0192	-0.0572	0.4363
XQ	0.40	0.0013	0.0188	-0.0597	0.4204
XQ	0.50	-0.0024	0.0184	-0.0632	0.3936
XQ	0.60	-0.0057	0.0182	-0.0665	0.3686
XQ	0.70	-0.0123	0.0175	-0.0739	0.3300
XQ	0.80	-0.0226	0.0157	-0.0881	0.2966
XQ	0.90	-0.0385	0.0123	-0.1099	0.2486
XQ	1.00	-0.0552	0.0075	-0.1323	0.2044
CP	0.00	0.0058	0.0193	-0.0569	0.4689
CP	0.10	0.0058	0.0226	-0.0582	0.4576
CP	0.20	0.0069	0.0234	-0.0566	0.4571
CP	0.30	-0.0017	0.0181	-0.0685	0.4560
CP	0.40	-0.0003	0.0202	-0.0677	0.4582
CP	0.50	0.0041	0.0224	-0.0612	0.4584
CP	0.60	0.0031	0.0226	-0.0625	0.4517
CP	0.70	0.0010	0.0199	-0.0641	0.4486
CP	0.80	0.0013	0.0236	-0.0664	0.4493
CP	0.90	-0.0038	0.0199	-0.0697	0.4196
CP	1.00	-0.0362	0.0144	-0.1165	0.3351

Table A.12: Popularity Lift Metric for Mitigation Algorithms ($N = 5000, k = 25$)

Appendix B

Questionnaires

In the following, we will list the questionnaires that are used in the study. Most questionnaires have to be asked on a 7-point agreement scale, where 1 indicates ”completely disagree” and 7 indicates ”completely agree”[1]. If an item is measured on a different scale, the options are shown in brackets below the item. Items typically contribute positively to the score. If an item is negatively coded, it is marked with a * symbol in the ”R” column.

B.1 Pre-Questionnaires

Metric	Source	Item	R
Personal Characteristics (PC)			
Demographics		Age (free-text comment)	
		Gender (Female; Male; Non-binary / third gender; Prefer not to say)	
Musical Sophistication	[81]	I spend a lot of my free time doing music-related activities	
		I enjoy writing about music, for example on blogs and forums	
		If somebody starts singing a song I don’t know, I can usually join in	
		I can sing or play music from memory	
		I am able to hit the right notes when I sing along with a recording	
		I can compare and discuss differences between two performances or versions of the same piece of music	
		I have never been complimented for my talents as a musical performer	*
		I often read or search the internet for things related to music	
		I am not able to sing in harmony when somebody is singing a familiar tune	*
		I am able to identify what is special about a given musical piece	
		When I sing, I have no idea whether I’m in tune or not	*

[1]7: Completely Agree, 6: Strongly Agree, 5: Agree, 4: Neither Agree Nor Disagree, 3: Disagree, 2: Strongly Disagree, 1: Completely Disagree

		Music is kind of an addiction for me - I couldn't live without it	
		I don't like singing in public because I'm afraid that I would sing wrong notes	*
		I would not consider myself a musician	*
		After hearing a new song two or three times, I can usually sing it by myself	
		I engaged in regular, daily practice of a musical instrument (including voice) for __ years (0; 1; 2; 3; 4-5; 6-9; 10 or more)	
		At the peak of my interest, I practiced __ hours per day on my primary instrument (0; 0.5; 1; 1.5; 2; 3-4; 5 or more)	
		I can play __ musical instruments (0; 1; 2; 3; 4; 5; 6 or more)	
Active (Musical) Engagement	[81]	I spend a lot of my free time doing music-related activities	
		I enjoy writing about music, for example on blogs and forums	
		I'm intrigued by musical styles I'm not familiar with and want to find out more	
		I often read or search the internet for things related to music	
		I don't spend much of my disposable income on music	*
		Music is kind of an addiction for me - I couldn't live without it	
		I keep track of new music that I come across (e.g. new artists or recordings)	
		I have attended _ live music events as an audience member in the past twelve months (0; 1; 2; 3; 4-6; 7-10; 11 or more)	
		I listen attentively to music for _ per day (0-15; 15-30; 30-60; 60-90; 2hrs; 2-3hrs; 4hrs or more)	

Table B.1: Pre-Questionnaires

B.2 Post-Questionnaires

Metric	Source	Item	R
Subjective System Aspects (SSA)			
Perceived popularity	[43]	The recommendations consist of popular tracks	

		The recommendations consist of tracks that are currently played a lot by the media	
		I do not think that the recommended tracks are very well known	*
		I believe that the majority of music listeners are familiar with the tracks in the recommendation	
		The recommended tracks have been receiving a lot of attention by the media	
Discovery	[43]	The recommendations broaden my taste	
		The recommendations deepen my taste	
		The recommendations allow me to discover new tracks to listen to	
		The recommendations allow me to refine my taste	
		The recommendations give me a new perspective on my musical taste	
Perceived fairness	[39]	The playlist has a good balance between popular and less popular items	
		The playlist would be more balanced if *more* popular items were included	*
		The playlist would be more balanced if *less* popular items were included	*
Familiarity	[43]	I am familiar with the recommended tracks	
		I do not know the tracks from the list	*
		I already listen to the tracks that are recommended	
		I have heard the recommended tracks before	
		The recommended tracks are already in my own playlists	
Perceived Recommendation Quality	[60]	I liked the items recommended by the system	
		The recommended songs fitted my preference	
		The recommended songs were well-chosen	
		The recommended songs were relevant	
		The system recommended too many bad songs	*
		I didn't like any of the recommended songs	*
		The songs I selected were "the best among the worst"	*
Experience (EXP)			
Recommendation satisfaction	[39, 43]	I am satisfied with the list of recommended tracks	
		In most ways the recommended tracks are close to ideal	
		The list of tracks recommendations meet my exact needs	

		I would give the recommended tracks a high rating	
		The list of tracks shows too many bad items	*
		The list of tracks is attractive	
		The list of recommendations matches my preference	
Choice satisfaction	[60]	I like the items I've chosen	
		I was excited about my chosen items	
		I enjoyed listening to my chosen items	
		The items I listened to were a waste of my time	*
		The chosen songs fit my preference	
		I know several songs that are better than the ones I selected	*
		Some of my chosen songs could become part of my favorites	
		I would recommend some of the chosen songs to others/friends	
Perceived system effectiveness	[60]	I would recommend the systems to others	
		The system is useless	*
		The system makes me more aware of my choice options	
		I make better choices with the system	
		I can find better songs without the help of the system	*
		I can find better songs using the recommender system	
		The system showed useful items	
Behavioural Intentions (BI)			
Use Intention	[90]	I will use this recommender again	
		I will use this recommender frequently	
		I will tell my friends about this recommender	
Choice Listening Intention	[90]	I will listen to the songs that I have chosen again in the future	
		I will add the songs that I have chosen to my playlist	
Openness to similar recommendations	[14]	I would like to be recommended more songs similar to the one I were recommended	
		I would like to receive more music recommendations that are similar in genre and style to the ones I just listened to.	
		I am interested in discovering more music from the same artists or bands that were recommended to me.	

	I would be likely to listen to more music from the same genre or style that was recommended to me.
	I would appreciate it if the music recommender system suggests more songs that are similar in mood and tone to the ones I just listened to.

Table B.2: Post-Questionnaires

Appendix C

User Study Evaluation

C.1 Pragmatic Evaluation

C.1.1 Algorithmic Effects

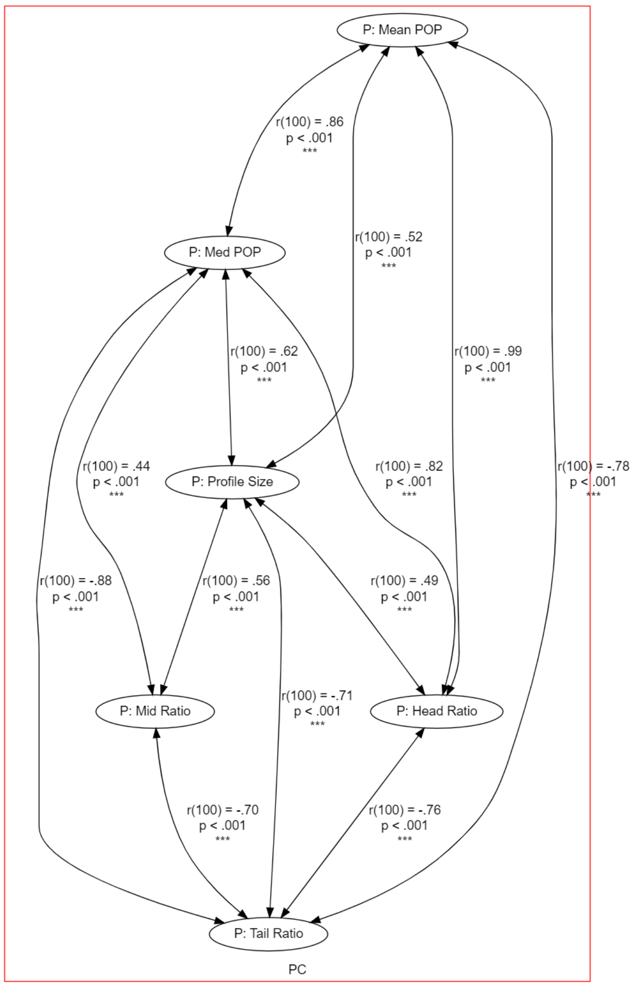

P: Mean POP
P: Med POP
P: Profile Size
P: Mid Ratio
P: Head Ratio
P: Tail Ratio
r(100) = .86 p < .001 ***
r(100) = .52 p < .001 ***
r(100) = .62 p < .001 ***
r(100) = .99 p < .001 ***
r(100) = .44 p < .001 ***
r(100) = .82 p < .001 ***
r(100) = -.78 p < .001 ***
r(100) = -.88 p < .001 ***
r(100) = .56 p < .001 ***
r(100) = .49 p < .001 ***
r(100) = -.71 p < .001 ***
r(100) = -.70 p < .001 ***
r(100) = -.76 p < .001 ***
PC

Figure C.1: Effects between factors in the PC

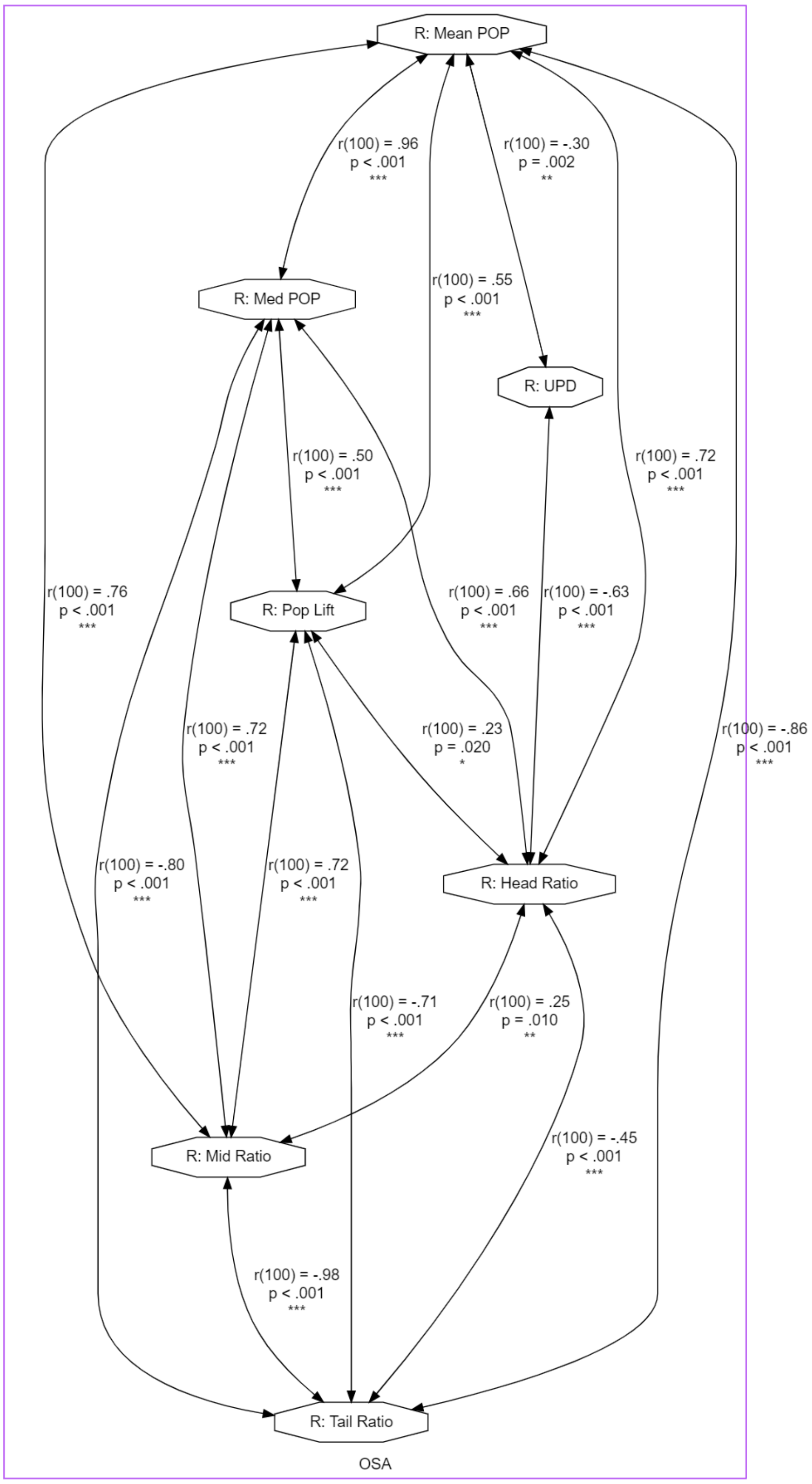

R: Mean POP
R: Med POP
R: UPD
R: Pop Lift
R: Head Ratio
R: Mid Ratio
R: Tail Ratio
OSA
r(100) = .96 p < .001 ***
r(100) = -.30 p = .002 **
r(100) = .55 p < .001 ***
r(100) = .50 p < .001 ***
r(100) = .72 p < .001 ***
r(100) = .76 p < .001 ***
r(100) = .66 p < .001 ***
r(100) = -.63 p < .001 ***
r(100) = .72 p < .001 ***
r(100) = .23 p = .020 *
r(100) = -.86 p < .001 ***
r(100) = -.80 p < .001 ***
r(100) = .72 p < .001 ***
r(100) = -.71 p < .001 ***
r(100) = .25 p = .010 **
r(100) = -.45 p < .001 ***
r(100) = -.98 p < .001 ***

Figure C.2: Effects between factors in the OSA

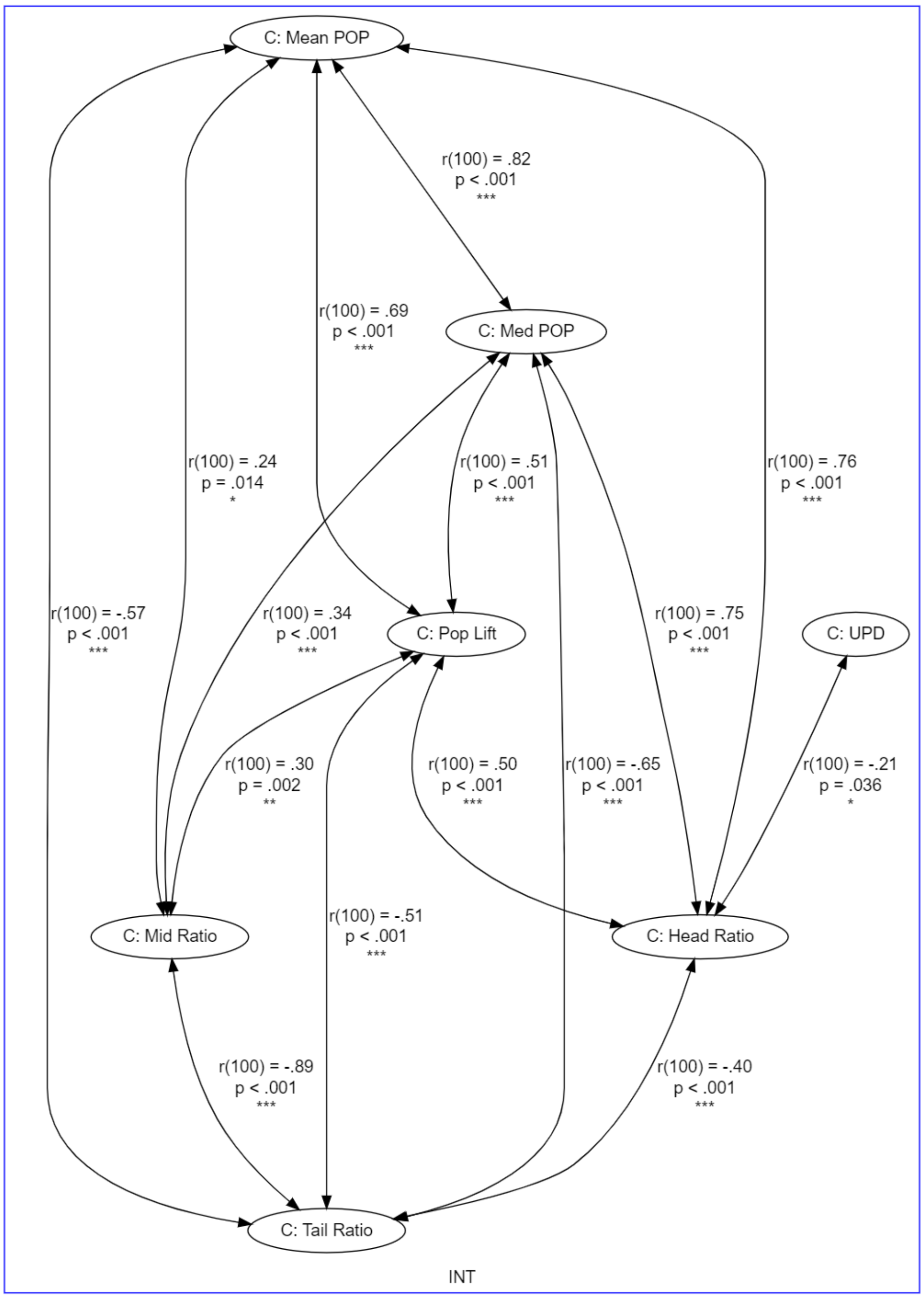
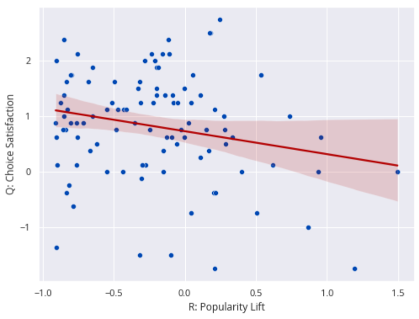
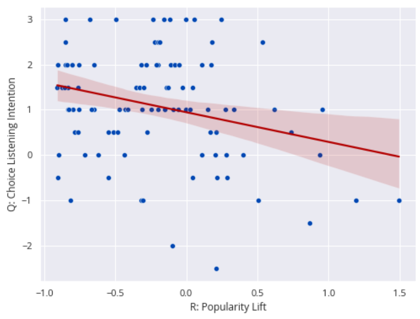

C: Mean POP
C: Med POP
C: Pop Lift
C: UPD
C: Mid Ratio
C: Head Ratio
C: Tail Ratio
r(100) = .82 p < .001 ***
r(100) = .69 p < .001 ***
r(100) = .24 p = .014 *
r(100) = .51 p < .001 ***
r(100) = .76 p < .001 ***
r(100) = -.57 p < .001 ***
r(100) = .34 p < .001 ***
r(100) = .75 p < .001 ***
r(100) = .30 p = .002 **
r(100) = .50 p < .001 ***
r(100) = -.65 p < .001 ***
r(100) = -.21 p = .036 *
r(100) = -.51 p < .001 ***
r(100) = -.89 p < .001 ***
r(100) = -.40 p < .001 ***
INT

Figure C.3: Effects between factors in the INT

C.1.2 Popularity Lift's Impact on EXP and BI



(a) *R: Popularity Lift* vs *Q: Choice Satisfaction*



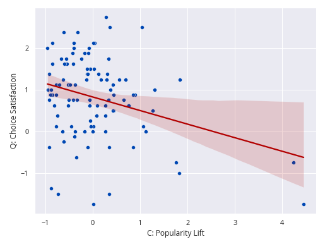
(b) *R: Popularity Lift* vs *Q: Choice Listening Intention*

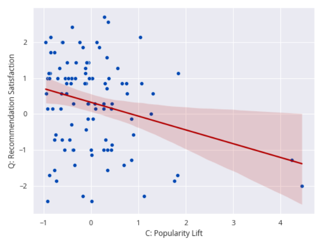
Figure C.4: Scatter Plots highlighting the relationships between *C: Popularity Lift* and EXP and BI questionnaire scores with regression line.

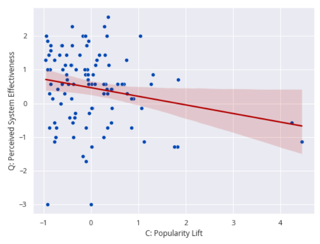
C.1.3 Personal Characteristics

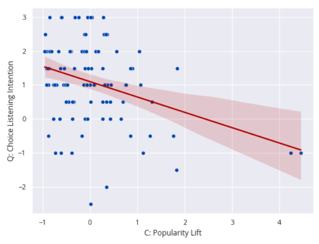
Personal Characteristic	Factor	Effect Size ($r(100)$)	p	$p < .05$
P: Mean POP	Q: Perc. Rec. Quality	.20	.040	*
P: Mean POP	Q: Choice Satisfaction	.21	.036	*
P Mean POP	Q: Perc. System Effectiveness	.21	.037	*
P: Mean POP	Q: Choice Listening Intention	.22	.026	*
P: Head Ratio	Q: Perc. Rec. Quality	.22	.029	*
P: Head Ratio	Q: Choice Satisfaction	.23	.018	*
P: Head Ratio	Q: Perc. System Effectiveness	.22	.025	*
P: Head Ratio	Q: Choice Listening Intention	.24	.013	*
P: Tail Ratio	Q: Perc. Rec. Quality	-.25	.010	*
P: Tail Ratio	Q: Rec. Satisfaction	-.22	.028	*
P: Tail Ratio	Q: Choice Satisfaction	-.20	.039	*
P: Tail Ratio	Q: Choice Listening Intention	-.25	.012	*
Q: Musical Sophistication	C: Med POP	-.20	.049	*
Q: Musical Sophistication	C: Head Ratio	-.24	.014	*
Q: Musical Sophistication	Q: Use Intention	.24	.015	*
Q: Musical Engagement	Time Spent	-.22	.023	*
Q: Musical Engagement	C: Mean POP	-.20	.048	*
Q: Musical Engagement	C: Med POP	-.31	.001	**
Q: Musical Engagement	C: Head Ratio	-.30	.002	**
Q: Musical Engagement	Q: Perc. Popularity	-.24	.014	*

Table C.1: Correlations between Personal Characteristics and other factors without algorithmic effects


(a) *C: Popularity Lift* vs *Q: Choice Satisfaction*


(b) *C: Popularity Lift* vs *Q: Recommendation Satisfaction*


(c) *C: Popularity Lift* vs *Q: Perceived System Effectiveness*


(d) *C: Popularity Lift* vs *Q: Choice Listening Intention*

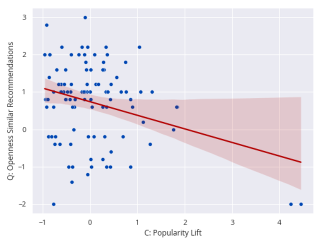

(e) *C: Popularity Lift* vs *Q: Openness to Similar Recommendations*

Figure C.5: Scatter Plots highlighting the relationships between *C: Popularity Lift* and EXP and BI questionnaire scores with regression line.

C.1.4 Interaction Factors

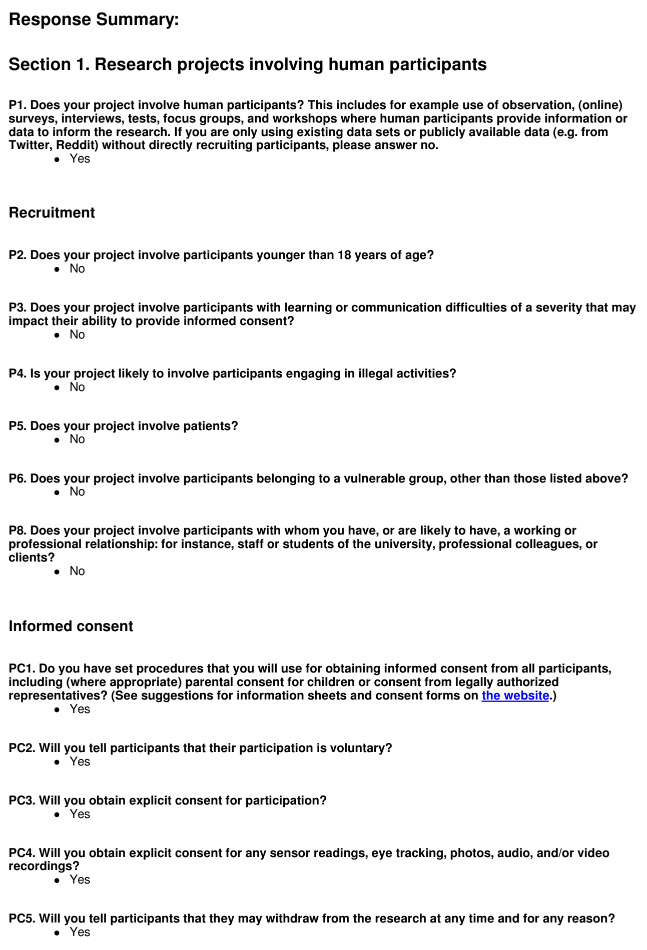
Interaction Factor	Factor	Effect Size ($r(100)$)	p	$p < .05$
C: Mid Ratio	Q: Perc. Popularity	0.30	0.003	**
C: Tail Ratio	Q: Perc. Popularity	-0.33	0.001	***
C: Median Interactions	Q: Familiarity	0.24	0.016	*
C: Mid Ratio	Q: Familiarity	0.27	0.006	**
C: Tail Ratio	Q: Familiarity	-0.32	0.001	**
C: Pop Lift	Q: Discovery	-0.26	0.007	**
C: Pop Lift	Q: Perc. Rec. Quality	-0.30	0.002	**
C: Pop Lift	Q: Recommendation Satisfaction	-0.27	0.006	**
C: Pop Lift	Q: Choice Satisfaction	-0.30	0.002	**
C: Pop Lift	Q: Perc. System Effectiveness	-0.21	0.036	*
C: Pop Lift	Q: Openness Sim. Rec.	-0.30	0.002	**
C: Pop Lift	Q: Choice Listening Intention	-0.34	0.001	***

Table C.2: Correlations between Personal Characteristics and other factors without algorithmic effects

Appendix D

Ethics and Privacy Quick Scan

After the literature research and before starting the research project, an "Ethics and Privacy Quick Scan" was filled in to assess potential privacy-related and ethical risks caused by the research. This quick scan did not predict any potential risks. The entire document can be found below.



Response Summary:

Section 1. Research projects involving human participants

P1. Does your project involve human participants? This includes for example use of observation, (online) surveys, interviews, tests, focus groups, and workshops where human participants provide information or data to inform the research. If you are only using existing data sets or publicly available data (e.g. from Twitter, Reddit) without directly recruiting participants, please answer no.
- Yes

Recruitment

P2. Does your project involve participants younger than 18 years of age?
- No

P3. Does your project involve participants with learning or communication difficulties of a severity that may impact their ability to provide informed consent?
- No

P4. Is your project likely to involve participants engaging in illegal activities?
- No

P5. Does your project involve patients?
- No

P6. Does your project involve participants belonging to a vulnerable group, other than those listed above?
- No

P8. Does your project involve participants with whom you have, or are likely to have, a working or professional relationship: for instance, staff or students of the university, professional colleagues, or clients?
- No

Informed consent

PC1. Do you have set procedures that you will use for obtaining informed consent from all participants, including (where appropriate) parental consent for children or consent from legally authorized representatives? (See suggestions for information sheets and consent forms on the website.)
- Yes

PC2. Will you tell participants that their participation is voluntary?
- Yes

PC3. Will you obtain explicit consent for participation?
- Yes

PC4. Will you obtain explicit consent for any sensor readings, eye tracking, photos, audio, and/or video recordings?
- Yes

PC5. Will you tell participants that they may withdraw from the research at any time and for any reason?
- Yes

PC6. Will you give potential participants time to consider participation?
- Yes

PC7. Will you provide participants with an opportunity to ask questions about the research before consenting to take part (e.g. by providing your contact details)?
- Yes

PC8. Does your project involve concealment or deliberate misleading of participants?
- No

Section 2. Data protection, handling, and storage

The General Data Protection Regulation imposes several obligations for the use of **personal data** (defined as any information relating to an identified or identifiable living person) or including the use of personal data in research.

D1. Are you gathering or using personal data (defined as any information relating to an identified or identifiable living person)?
- Yes

High-risk data

DR1. Will you process personal data that would jeopardize the physical health or safety of individuals in the event of a personal data breach?
- No

DR2. Will you combine, compare, or match personal data obtained from multiple sources, in a way that exceeds the reasonable expectations of the people whose data it is?
- No

DR3. Will you use any personal data of children or vulnerable individuals for marketing, profiling, automated decision-making, or to offer online services to them?
- No

DR4. Will you profile individuals on a large scale?
- No

DR5. Will you systematically monitor individuals in a publicly accessible area on a large scale (or use the data of such monitoring)?
- No

DR6. Will you use special category personal data, criminal offense personal data, or other sensitive personal data on a large scale?
- No

DR7. Will you determine an individual's access to a product, service, opportunity, or benefit based on an automated decision or special category personal data?
- No

DR8. Will you systematically and extensively monitor or profile individuals, with significant effects on them?
- No

DR9. Will you use innovative technology to process sensitive personal data?
- No

Data minimization

DM1. Will you collect only personal data that is strictly necessary for the research?
- Yes

DM4. Will you anonymize the data wherever possible?
- Yes

DM5. Will you pseudonymize the data if you are not able to anonymize it, replacing personal details with an identifier, and keeping the key separate from the data set?
- Yes

Using collaborators or contractors that process personal data securely

DC1. Will any organization external to Utrecht University be involved in processing personal data (e.g. for transcription, data analysis, data storage)?
- No

International personal data transfers

DI1. Will any personal data be transferred to another country (including to research collaborators in a joint project)?
- No

Fair use of personal data to recruit participants

DF1. Is personal data used to recruit participants?
- No

Participants' data rights and privacy information

DP1. Will participants be provided with privacy information? (Recommended is to use as part of the information sheet: For details of our legal basis for using personal data and the rights you have over your data please see the University's privacy information at www.uu.nl/en/organisation/privacy.)
- Yes

DP2. Will participants be aware of what their data is being used for?
- Yes

DP3. Can participants request that their personal data be deleted?
- Yes

DP4. Can participants request that their personal data be rectified (in case it is incorrect)?
- Yes

DP5. Can participants request access to their personal data?
- Yes

DP6. Can participants request that personal data processing is restricted?
- Yes

DP7. Will participants be subjected to automated decision-making based on their personal data with an impact on them beyond the research study to which they consented?

- No

DP8. Will participants be aware of how long their data is being kept for, who it is being shared with, and any safeguards that apply in case of international sharing?

- Yes

DP9. If data is provided by a third party, are people whose data is in the data set provided with (1) the privacy information and (2) what categories of data you will use?

- Not applicable

Using data that you have not gathered directly from participants

DE1. Will you use any personal data that you have not gathered directly from participants (such as data from an existing data set, data gathered for you by a third party, data scraped from the internet)?

- No

Secure data storage

DS1. Will any data be stored (temporarily or permanently) anywhere other than on password-protected University authorized computers or servers?

- Yes

DS2. Does this only involve data stored temporarily during a session with participants (e.g. data stored on a video/audio recorder/sensing device), which is immediately transferred (directly or with the use of an encrypted and password-protected data-carrier (such as a USB stick)) to a password-protected University authorized computer or server, and deleted from the data capture and data-carrier device immediately after transfer?

- Yes

DS4. Excluding (1) any international data transfers mentioned above and (2) any sharing of data with collaborators and contractors, will any personal data be stored, collected, or accessed from outside the EU?

- No

Section 3. Research that may cause harm

Research may cause harm to participants, researchers, the university, or society. This includes when technology has dual-use, and you investigate an innocent use, but your results could be used by others in a harmful way. If you are unsure regarding possible harm to the university or society, please discuss your concerns with the Research Support Office.

H1. Does your project give rise to a realistic risk to the national security of any country?

- No

H2. Does your project give rise to a realistic risk of aiding human rights abuses in any country?

- No

H3. Does your project (and its data) give rise to a realistic risk of damaging the University's reputation? (E.g., bad press coverage, public protest.)

- No

H4. Does your project (and in particular its data) give rise to an increased risk of attack (cyber- or otherwise) against the University? (E.g., from pressure groups.)

- No

H5. Is the data likely to contain material that is indecent, offensive, defamatory, threatening, discriminatory, or extremist?

- No

H6. Does your project give rise to a realistic risk of harm to the researchers?

- No

H7. Is there a realistic risk of any participant experiencing physical or psychological harm or discomfort?

- No

H8. Is there a realistic risk of any participant experiencing a detriment to their interests as a result of participation?

- No

H9. Is there a realistic risk of other types of negative externalities?

- No

Section 4. Conflicts of interest

C1. Is there any potential conflict of interest (e.g. between research funder and researchers or participants and researchers) that may potentially affect the research outcome or the dissemination of research findings?

- No

C2. Is there a direct hierarchical relationship between researchers and participants?

- No

Section 5. Your information.

This last section collects data about you and your project so that we can register that you completed the Ethics and Privacy Quick Scan, sent you (and your supervisor/course coordinator) a summary of what you filled out, and follow up where a fuller ethics review and/or privacy assessment is needed. For details of our legal basis for using personal data and the rights you have over your data please see the University's privacy information. Please see the guidance on the ICS Ethics and Privacy website on what happens on submission.

Z0. Which is your main department?

- Information and Computing Science

Z1. Your full name:

Robin Ungruh

Z2. Your email address:

r.ungruh@students.uu.nl

Z3. In what context will you conduct this research?

- As a student for my master thesis, supervised by::
  Hanna Hauptmann

Z5. Master programme for which you are doing the thesis

- Human-Computer Interaction

Z6. Email of the course coordinator or supervisor (so that we can inform them that you filled this out and provide them with a summary):

h.j.hauptmann@uu.nl

Z7. Email of the moderator (as provided by the coordinator of your thesis project):
graduation.hci@uu.nl

Z8. Title of the research project/study for which you filled out this Quick Scan:
Addressing the Long Tail: Investigating the Impact of Fair Music Recommendations on Listening Behaviour

Z9. Summary of what you intend to investigate and how you will investigate this (200 words max):
I will investigate the impact of modified personalized music recommendations on the user. For this purpose, I will develop a music recommender system. The users will interact with the recommendation system in an online study. Participants will be informed about the aim of the study and that their participation is voluntary. I will gather data about the users listening history from their Spotify listening history by using the Spotify API. Users can log in, and their previous interaction data can be retrieved. Based on this, they will receive recommendations and a task that asks them to explore the recommendation lists. Finally, they will be asked to fill in various questionnaires to investigate their perception of the recommendations. Some interaction data will be stored in a log file during the study. All data that could be used to identify the participant, like their Spotify ID, will not be stored, removed immediately or anonymized. Finally, some users who are willing to participate will also be interviewed to gain more insights into their experiences during the conduction of the study. During this step, I will make audio recordings, which will be transcribed.

Z10. In case you encountered warnings in the survey, does supervisor already have ethical approval for a research line that fully covers your project?

- Not applicable

Scoring

- Privacy: 0
- Ethics: 0
