Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Ingebjørg Dora Maxine Barthold

Privacy Leaks in Recommender Lists

Exploring Obfuscation Techniques to Preserve Privacy

Master's thesis in Datateknologi
Supervisor: Özlem Özgöbek
May 2023

NTNU
Norwegian University of
Science and Technology

Ingebjørg Dora Maxine Barthold

Privacy Leaks in Recommender Lists

Exploring Obfuscation Techniques to Preserve Privacy

Master's thesis in Datateknologi
Supervisor: Özlem Özgöbek
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Ingebjørg Barthold

Privacy Leaks in Recommender Lists:

Exploring Obfuscation Techniques to Preserve Privacy

Master's Thesis in Computer Science, June 2023
Supervisor: Özlem Özgöbek

Department of Computer Science
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Abstract

Recommender Systems have become an essential part of today's online services, providing entertaining content to each individual user of a service. However, these systems require a vast amount of user data, making them vulnerable to privacy attacks. The focus of this thesis is on the attack that manages to infer a user's gender based on a set of recommended movies for that given user. Even though earlier works have indeed focused on privacy in recommender systems, there is limited research on privacy-preserving techniques for recommender lists. The goal of this thesis is thus to experiment with obfuscation techniques, techniques that replace original items with new and "noisy" items, to prevent an adversary from being able to infer users' genders.

To do so, a set of obfuscation techniques discussed in earlier research, along with techniques traditionally used to introduce serendipity, are performed on recommender lists. These recommender lists are generated based on the MovieLens100K dataset. For each particular technique, the average gender leakage and recommender performance is measured.

The final contributions of this work include an awareness of the missing privacy preservation in Recommender Systems, along with the techniques that can be used to lower the accuracy of gender inference attacks. The results of the thesis reveal that the serendipity-introducing technique based on the concept of "K furthest neighbors" is able to lower inference performance while simultaneously preserving some degree of personalization. Moreover, the results also show that an increasing degree of obfuscation that decreases the degree of personalization does not necessarily correspond to better privacy preservation.

Sammendrag

Flesteparten av dagens digitale tjenester benytter en eller annen form for et anbefalingssystem. En stor ulempe med disse anbefalingssystemene er at de baserer seg på store mengder med persondata, noe som gjør dem utsatte for personvernangrep. Denne masteroppgaven tar for seg scenarioet der en angriper klarer å finne en brukers kjønn, utelukkende basert på brukerens filmanbefalinger. Det finnes verk som tar for seg personvern i anbefalingssystemer, men blant disse verkene er det manglende fokus på personvern i selve anbefalingslistene. Målet med denne oppgaven er derfor å finne og eksperimentere med teknikker som bytter ut "items" i anbefalingslister for å se om disse byttene kan vanskeliggjøre kjønnsklassifisering.

Teknikkene som brukes for å endre på anbefalingslistene er basert på tidligere personvernarbeid i anbefalingssystemdomenet. I tillegg eksperimenterer arbeidet med en teknikk som er mer knyttet opp mot "tilfeldige funn" (kalt serendipity på engelsk). Denne teknikken baserer seg på å anbefale en bruker u "items" som er mislikt av brukere som er ulike bruker u, preferansemessig. Mer spesifisert kalles "tilfeldige funn"-teknikken for "k-Furthest Neighbor (kFN)". Anbefalingslistene som modifiseres er generert ved bruk av en filmdatabase ved navn MovieLens100K.

Selve resultatene fra masterarbeidet fås ved å undersøke hvor relevante de modifiserte anbefalingslistene er, samt hvor bra de skjuler brukerens kjønn (kun mann og kvinne er tatt hensyn til). Resultatene viser at strategien som baserer seg på "tilfeldige funn" er den som gjør det best generelt, både med tanke på angrepsbeskyttelse og relevanse i anbefalingene. Videre peker resultatene også på at en høyere grad av modifisering, der modifiseringen medfører mindre personalisering, ikke nødvendigvis resulterer i mer personvernvennlige anbefalingslister.

Masterens bidrag er dermed en utforskning av mangelen på personvernfokus i anbefalingslister, i kombinasjon med brukbare teknikker som kan danne grunnlaget for fremtidige tiltak.

Preface

This thesis concludes the work conducted for my Master's Thesis at the Norwegian University of Science and Technology (NTNU). My thesis, including the work behind, was made possible with the support and guidance provided by my supervisor Özlem Özgöbek, Associate Professor at NTNU's Department of Computer Science. Thank you Özlem, for sharing your opinions on both my work and the field, and for willingly supervising me remotely all year.

In addition to my supervisor, I would like to thank Manel Slokom for always taking her time to help me and give me the resources I needed - I am grateful for your encouragement and technical assistance. Next, I am grateful for the support from my friends and family, and especially Marius who have listened to and discussed my daily thesis updates.

Ingebjørg Barthold
Oslo, 16th May 2023

Contents

List of Figures

List of Tables

Acronyms

FM Factorization Machine.

kFN k-Furthest Neighbor.

kNN k-Nearest Neighbor.

LogReg Logistic Regression.

NB Naive Bayes.

PII Personal Identifiable Information.

QID Quasi-Identifier.

SA Sensitive Attribute.

SEntropy Shannon Entropy.

1 Introduction

This introductory chapter describes the background of the research conducted, along with the motivation for this master thesis. Furthermore, it introduces the goals and research questions, followed by the research methodology. Lastly, the introduction chapter contains an overview of the thesis structure.

1.1 Background and Motivation

Most, if not all, of the large online services in use today are either based on or influenced by recommender systems. A recommender system is a tool helping customers or users navigate among the abundance of items in an online service's item catalog. This catalog may contain items such as books, movies, and grocery items, or social media posts such as videos on TikToks. In addition to being a great help to the users of a system, recommender systems are useful tools for the services that implement them. For example, if an online video-sharing platform has a good recommender algorithm, which shows its users videos they find interesting or entertaining, the users keep coming back for more. On the other hand, if the platform fails to provide the users with the content they want, they will move on to a different platform. In other words, recommender systems may strengthen or weaken the position of a service in the market.

Another example of why recommender systems are important for online services is that they also help match niche items with interested users (Silveira et al., 2019). Before personalization was introduced into online systems, it was seldom profitable to promote niche items. However, with personalized systems that may present different items to different users, niche items can also be promoted. As a result, personalized systems and recommender systems increase the proportion of items that are publicized along with the number of users being shown items that they actually enjoy.

Despite being a great tool for finding relevant and interesting items, concerns regarding the privacy aspect of recommender systems have been raised in recent years. With the growing user interest in digital services, these services are able to collect an increasing amount of data. The vast collection of users' personal information and historical behavior is what makes data leakage a vulnerability. There exist regulations trying to affect the recommender systems in a privacy-preserving way, such as GDPR [1] and CCPA [2]. Some general principles that can be taken out of the privacy regulations are data minimization and the right of a user to control the storage of personal data. Today, service providers comply to the regulations by making the users agree to their terms before accessing the

[1] https://gdpr-info.eu/
[2] https://oag.ca.gov/privacy/ccpa

service. Eventually, what this results in, is the concept of "all-or-nothing" (Chen et al., 2022), dividing potential users into two groups. The first one is the "yes-group" that allows all (legal) data collection and thus are provided with good personalization. The other group is the "no-group", which either gets no personalization at all or is denied access to the service. Anyhow, some users want a certain degree of privacy *and* personalization, which suggests a need for more privacy-aware services.

For recommender systems, it has been proven that the recommender lists leak sensitive information (Slokom et al., 2022). It is possible to acquire information about a user's gender, location, or age based on that same user's recommended items. An attack that infers user information based on recommender lists is a type of attribute inference attack (Ge et al., 2022). Even though the recommender list leakage problem is identified, research on how to mitigate the problem is still lacking. Yet, there exists research on a similar leakage problem in recommender systems: the leakage of personal information from users' historical ratings. These historical ratings are often stored in so-called user-item matrices. To protect these matrices from attribute inference attacks, obfuscation has proven to be an effective method. Therefore, this research focuses on the possibility of using similar obfuscation techniques to mitigate attribute inference attacks in recommender lists.

1.2 Goals and Research Questions

In short, the main goal of this thesis can be summed up as follows:

Goal Explore techniques for developing more privacy-preserving recommender lists (recommender system output).

This goal is very general, difficult to measure, and without any definition of done. Therefore, the following research questions are defined as smaller, more specific, and measurable steps toward achieving the main goal:

Research question 1 To what extent is it possible to obfuscate recommender lists in order to lower the accuracy of attribute inference attacks by using techniques proven effective in the obfuscation of user-item matrices?

Research question 2 In terms of recommendation performance, how do the obfuscated recommendation lists perform compared to the originals? Moreover, are there considerable differences between the explored obfuscation techniques?

1.3 Research Method

To answer the presented research questions, a set of experiments were conducted. The experiments consist of different strategies that modify the output of a recommender system. Each of the experiments concerns a given replacement strategy that replaces items in the original recommender list with new items.

The implementation of the work can be found on GitHub.

1.4 Contributions

The main contributions of this thesis can be summed up into the following:

1. This work has proved that serendipity techniques may be effective in privacy-preservation of user attributes.
2. The work has also shown that an obfuscation that results in lower personalization (measured with offline metrics) does not necessarily lead to protection against inference attacks.
3. This thesis has discussed that there is a need for further research into protection against attribute inference attacks on recommender lists.

1.5 Thesis Structure

This section provides an overview and description of all the chapters in this thesis.

Chapter 1 - Introduction Introduces the problem in the field along with how and why the work presented tries to solve it.

Chapter 2 - Background Theory Introducing and describing well-establish theories and techniques relevant to this master thesis.

Chapter 3 - Related Works An overview of recent relevant works published in the field of privacy in recommender systems.

Chapter 4 - Datasets An introduction and description of the dataset used in the experiments.

Chapter 5 - Method and Experiment This chapter explains the experiments of the thesis, along with the tools and methods used to conduct them.

Chapter 6 - Results Here are the results of the experiments presented.

Chapter 7 - Evaluation and Discussion A chapter for looking at the results in more detail, allowing both evaluation and discussion of them.

Chapter 8 - Conclusion and Future Work Concludes the findings and contributions of this master thesis along with introducing further possibilities for the field.

2 Background Theory

This chapter attempts to introduce relevant concepts for this thesis. The chapter is heavily based on Barthold (2022), a report from the NTNU Specialization Project in Computer Science. First, some general theory about recommender systems is presented, before privacy-related theories and techniques are introduced.

2.1 Recommender Systems

A recommender system is, as the name suggests, a system that generates recommendations for a user. A visual example of the user side of a recommender system implementation can be seen by visiting the video-sharing platform YouTube[1], as in Figure 2.1. Based on a user's previous clicks, searches, and likes, YouTube provides a list of videos that are likely to be found interesting by the user. In short, a good recommender system is a system that provides users with entertaining items that they did not know they were looking for.

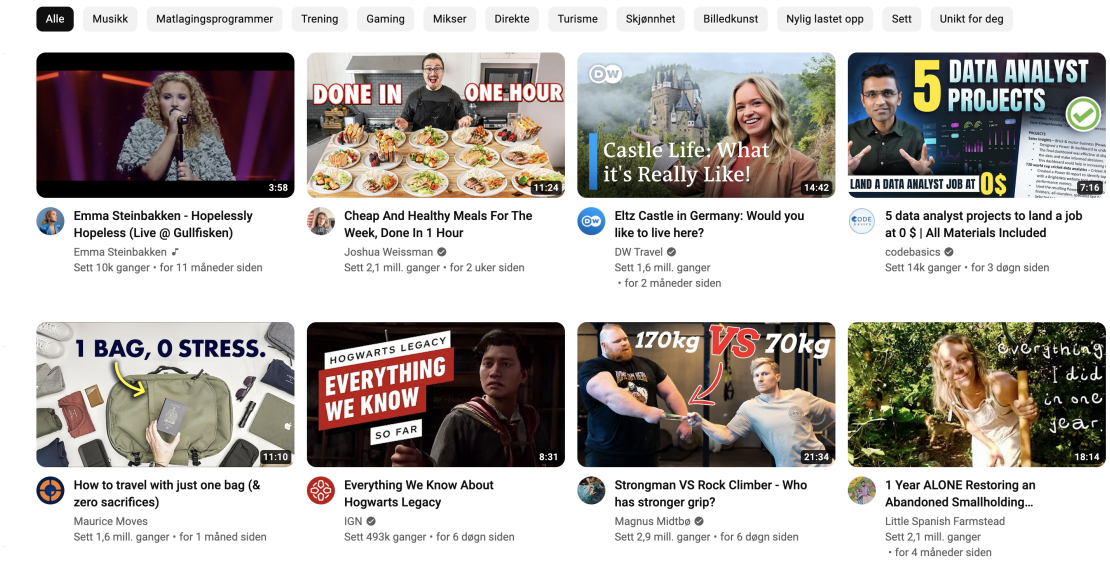


Figure 2.1: A recommendation list from YouTube.

The general behavior of recommender systems is rooted in the concept of a feedback

[1] https://www.youtube.com/

loop. The feedback loop has two components: the recorded user interactions (user ratings and activity), and the recommended items. Both elements are visualized in Figure 2.2. As the figure further illustrates, the user interactions lead to a set of recommended items that, in turn, give the user more items to interact with. Most likely, each new iteration of recorded interactions is based on the previously presented recommender list, hence the loop. The main goal of the recommender is to predict ratings for items the user has not seen yet and use that to decide which items to present next.

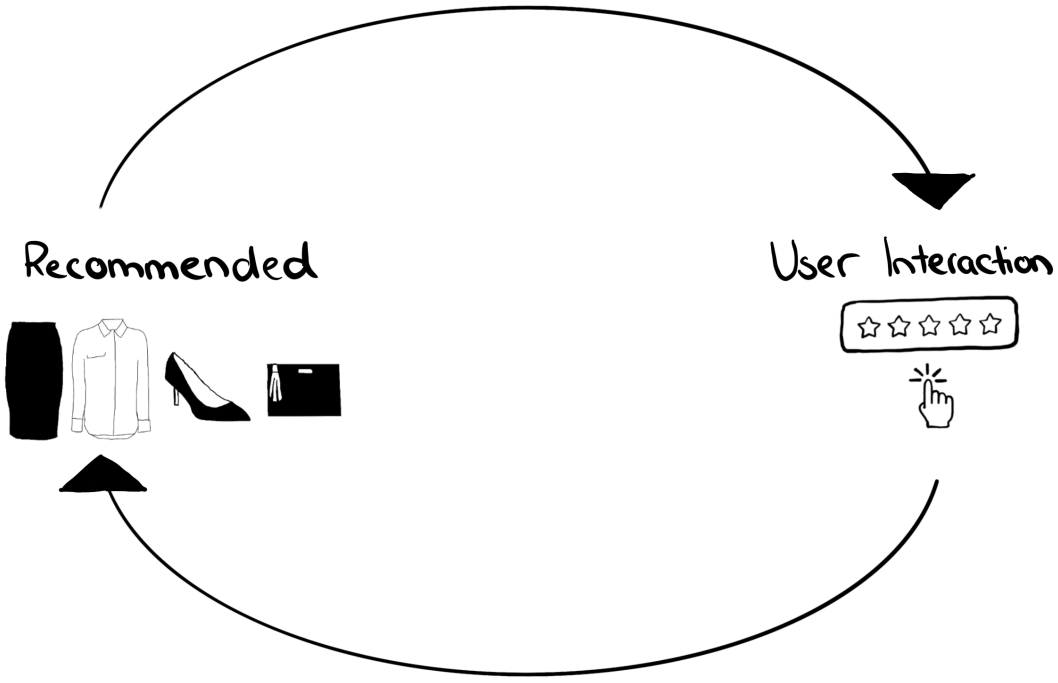


Figure 2.2: The life cycle of the feedback loop in recommender systems. Each list of recommended items leads to a new set of user interactions, and each set of user interactions leads to a new set of recommendations.

Modern recommender systems collect and process various data, from online behaviors such as browsing history and explicit ratings to private information such as addresses and age. What information a recommender system prioritizes, is different from system to system. We can sort the types of data the recommender systems require for optimal performance into three groups: user information, item information, and context information. The first group, user information, includes preferences, personal information, and historical behaviors in the application. This user information can be further partitioned into Personal Identifiable Information (PII), Quasi-Identifiers (QIDs), and Sensitive Attributes (SAs), depending on the sensitivity and traceability of the data (Saleem et al., 2021). Table 2.1 explains the named partitions in more detail.

Table 2.1: Terms describing different user attributes

Name	Abbreviation	Description
Personal Identifiable Information	PII	Attributes that uniquely identify a user without being accompanied by other attributes (e.g. email, phone number, ID number)
Quasi-Identifier	QID	Attributes that with the help of some auxiliary information (e.g other quasi-identifiers) can identify a user
Sensitive Attribute	SA	Attributes that the user wants to hide or that are vulnerable to unfair treatment, such as gender and political orientation.

The item information, which makes up the second group, describes the individual items in the catalog. This can be explicitly defined attributes in the form of metadata, or it may be information extracted from the item itself. E.g. for written texts, information about each text can be automatically extracted tags that describe the theme and content of that document.

Context information consists of extra information about the user's action. Examples of such information are time, place, and location. Context information is important because such factors may affect the users' preferences. For instance, users may want to watch Christmas movies because it is December, or listen to "focus music" because they are at the university.

To predict the likelihood of whether a user will like an item or not, different strategies have been proposed. Three popular such strategies, as presented in Aggarwal et al. (2016), are Collaborative Filtering, Content-Based Filtering, and Hybrid Systems.

2.1.1 Collaborative Filtering

Collaborative filtering focuses on the idea that similar users unveil similar rating patterns - as illustrated in Figure 2.3. The calculations of similarity and predicted ratings are performed by looking at previous ratings for all the users in the system. Users that have rated the same items in a similar manner are said to be similar. The focus of collaborative filtering can also be translated to hold for item similarity because also similar items unveil similar rating patterns.

The recorded ratings are traditionally stored in what is called a "user-item matrix". Each "coordinate" ($userID$, $itemID$) in the matrix corresponds to the rating of $itemID$ given by $userID$. User-item matrices may either be filled with implicit or explicit ratings. Implicit ratings are traditionally presented as binary values where 1 denotes "interacted with", e.g. "clicked on" or "watched", and 0 means "not interacted with". On the other hand, explicit ratings are values describing how much a user liked an item. The latter of the rating methods is more informative, but today, most of the recorded data are implicit

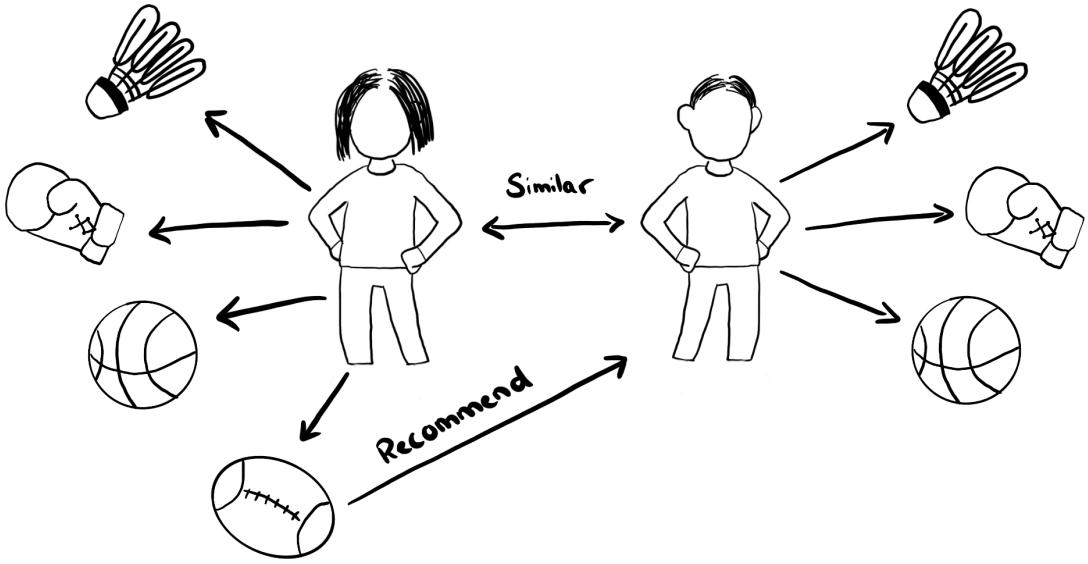


Figure 2.3: An illustration of the idea that users are recommended items that are liked by similar users.

ratings.

Another characteristic of today's recorded data is that the distribution of ratings among items forms the shape of a long tail when presented on a graph based on the number of times each item is rated. This long tail shape is shown in Figure 2.4. The figure shows that few items are rated frequently, and the rest, which consists of the majority of items, are rarely rated. A result of the long tail property is that popular items (head-items) are recommended more often than non-popular items (tail-items).

One of the advantages of collaborative filtering is that it is transferable to all domains, as it is domain free (Koren et al., 2009).

2.1.2 Content-Based Filtering

Content-based filtering compares characteristics (attributes, keywords,..) of items a user has liked in the past with the characteristics of unseen items. This method is particularly effective if the data and metadata, along with user preferences, are structured or easy to retrieve. One of the strengths of content-based filtering is that cold-start items - meaning new items that are not yet interacted with - can be recommended even though no historical ratings exist. Also, new users can get relevant recommendations from content-based systems, as long as they provide some kind of information regarding their interests. In Figure 2.5, the general idea of content-based filtering is illustrated. Here, the user has previously liked dystopian books, and thus another popular dystopia is recommended to the user.

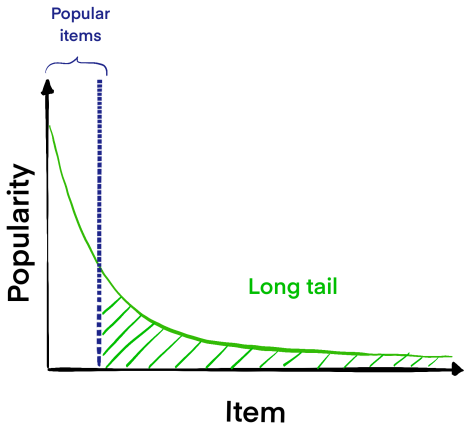


Figure 2.4: The long tail property.

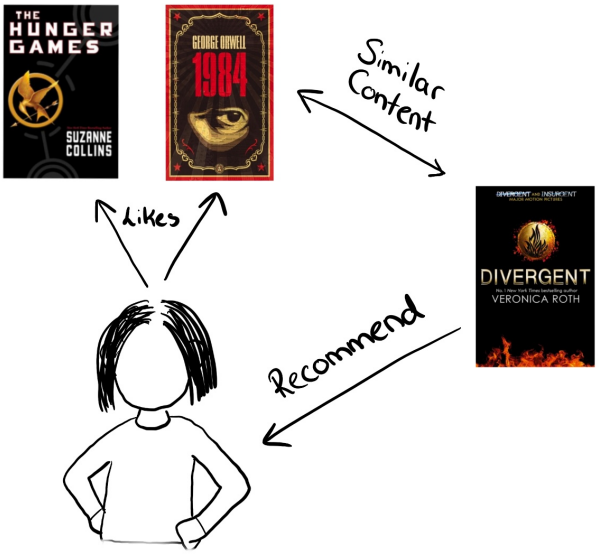


Figure 2.5: An illustration of how content-based filtering works.

2.1.3 Hybrid Methods

Hybrid systems combine characteristics from multiple recommender systems to build on their different strengths. There are various ways in which the systems are used together. Firstly, one can combine the existing recommender systems as they are and produce a single output. This method is called an ensemble design. A monolithic system, on the other hand, combines the different systems into one unified system, where there

is no clear distinction between the original systems. This method often requires some modification of the existing systems. Lastly, there is a method called mixed systems. This method simply presents recommendations from different systems side by side. For further explanations of the hybrid methods, see Aggarwal et al. (2016).

2.2 Factorization Machines

The preceding section on recommendation systems, section 2.1, introduced recommender systems and different recommender models in general. In contrast, this section elaborates on a specific type of recommender called Factorization Machines (FMs), introduced in Rendle (2010). Compared to other classification and regression models, such as Support Vector Machines and Matrix Factorization, FMs are effective in situations with very sparse data and they have linear complexity. These two strengths make Factorization Machines a good fit for recommender systems.

Factorization Machines are supervised learning models that, for the 2-way FMs, are described by the model equation Equation 2.1. The designation "2-way FM" means that the factorization machine is able to capture single and pairwise interactions between the variables. $\hat{y}(x)$ is the function that predicts the rating behavior of a given user for a given item, where the input x contains real-valued feature vectors. In addition to the traditionally contained information - user, item, and rating -, the input vector of FMs may hold arbitrary auxiliary features such as last interaction or gender. These auxiliary features allow for the inclusion of information relevant to the interactions.

The parameters in Equation 2.1 are described as the following: $w_0 \in \mathbb{R}$ is the global bias, $w_i \in \mathbb{R}^n$ is the strength of the i-th variable and $\mathbf{V} \in \mathbb{R}^{n \times k}$ contains the rows v_i describing the i-th variable with k factors. All of these parameters are to be estimated based on the training samples during the training of the model.

$$\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_i, v_j > x_i x_j \tag{2.1}$$

In terms of applicability, FMs are a good fit for different prediction tasks. In the context of recommender systems, these prediction tasks generally include regression tasks that predict the rating of an item, binary classification to predict if an item is relevant or not, and ranking tasks where the goal is to order the vectors $\mathbf{x}$ based on the score of $\hat{y}(x)$.

2.3 Privacy in Recommender Systems

The collection and processing of user data, both personal and behavioral, lead to privacy concerns. As mentioned in the introduction, this increasing collection has resulted in rules and laws. Consequently, there has been interest in fair and ethical recommender systems in recent years. With regards to privacy, this means a focus on the concepts of threat awareness, privacy protection, and authorized access to private information

(Ge et al., 2022). These concepts coincide with the definition of information privacy as formulated in Kang (1997).

> Information privacy is "an individual's claim to control the terms under which personal information – information identifiable to the individual – is acquired, disclosed or used." (Kang, 1997)

This section will focus on the privacy threats of recommender systems and how to mitigate them. Moreover, the attack named attribute inference attack is described in detail, as this is the attack this master thesis tries to prevent.

2.3.1 Privacy Threats

This subsection is inspired by threats identified and described in "A Survey on Trustworthy Recommender Systems" (Ge et al., 2022) and "Latest trends of security and privacy in recommender systems: A comprehensive review and future perspectives" (Himeur et al., 2022).

The main concept that is discussed in the context of privacy threats is privacy breaches. This general term concerns information that is accessed by someone who does not have permission to access that information. Without permission, access to the information is achieved by breaking into the systems with the help of hacking and monitoring. Generally, this threat contains both a security breach - when a protected computer is entered - and private information disclosure - when the user data is exposed. A further privacy threat related to data access is missing access control. Missing access control describes a lack of policies that ensure limitations on what the users of a system can access. In the field of computer science, this is tightly connected to the principle of least privilege, where a user (in this case) should only be able to access the information and resources necessary for him/her to carry out his/hers legitimate purpose.

For recommender systems and personalized systems, disclosing data to third parties or the public is not new but can be seen as a threat. Examples, where data sharing might occur, is for research purpose, targeted advertising, or, as a more specified example, when patient data is being shared between health researchers. In most of these cases, it can be assumed that the data is anonymized, meaning that the PII (see Table 2.1) are removed and anonymization techniques - as described in subsection 2.3.2 - are applied. However, there are multiple attacks that can de-anonymize user identities. For example, linkage attacks concatenate different datasets, where at least one of the datasets is not anonymized, such that one can disclose the identity of each user. Another attack is the homogeneity attack which can disclose sensitive attributes when there is homogeneity in the dataset. Often, this attack requires some kind of background information about the target user, such as age or city of residence. In some situations, data disclosure for legitimate purposes may also be a reason for data sharing, and in this case, the data could be un-anonymized. Although this data disclosure is lawful, the sharing is usually done even though there is no given consent from the users. This is a trade-off between law enforcement and privacy.

As mentioned, a reason for sharing data from personalized systems may be the selling of data for the purpose of advertising. Targeted advertising is a lucrative business, where advertisers utilize characteristics or user preferences to target different user groups in their advertisement campaigns. If the user is not formally informed about- and has accepted the selling of his or her data, this is a violation of the information privacy noted earlier in this section. One of the risks of targeted advertising, which also applies to recommender systems in general, is that it might alter a user's preferences. A harmful example of this lack of autonomy is targeted advertising for elections, where the advertisements affect the users voting behavior.

An inference attack is an attack that fits into the "false privacy" category. This attack is able to infer sensitive attributes for users, only by using public information. One could therefore think that one is protected when only public and insensitive information is shared. However, the inference attacks may also predict your private attributes with high probability, thereby the "false privacy". Even in systems that do not store or collect private information, private information could be accurately predicted. There are different types of inference attacks, but one of them - namely, attribute inference attack - is described more in detail in section 2.4.

2.3.2 Privacy Preserving Techniques

Himeur et al. (2022) lists three approaches to mitigate the discussed privacy threats in recommender systems. The first of these approaches is architecture-based protection, which aims to protect against privacy breaches and data theft. An example of how the architecture can be used to protect against data theft is by creating the system as a federated learning system. With federated learning, the user data is stored over multiple different devices, such as locally on each of the users' devices, without the raw data being transferred to the central server. Instead, the distributed devices send model weights to the central server that eventually updates the global model(Yang et al., 2019).

Other proposed approaches to make recommender systems better at preserving privacy are by introducing laws and regulations, or by using algorithmic privacy-preserving techniques. Regulations and algorithmic privacy-preserving techniques are presented more in detail in this subsection.

Rules and Regulations

To conduct and administer the advancements of services such as recommender systems, regulations have been proposed. GDPR, mentioned in the introductory chapter, is one of these sets of regulations. GDPR stands for General Data Protection Regulation and aims to increase the right of self-determination by giving users control and rights over their personal data. Although being made for the protection of EU citizens, the GDPR has been a model for similar protection regulations all around the world. Further guidelines, also proposed by the EU, is the Digital Service Act, which among other things, directly regards recommender systems by requiring more transparent systems. Moreover, it demands that larger platforms will allow users to choose whether or not

to have recommendations based on profiling, along with restricting targeted advertising based on users' special characteristics and for children in general.

Anonymization

Anonymization is the technique of hiding a user's identity in its user data. This can be described as de-identification or pseudonymization. A naive solution is to simply remove PIIs such as email or national ID numbers. Nowadays, this removal is combined with more mathematical and secure anonymization procedures, such as k-anonymity. k-anonymity as a privacy measure was defined by Samarati and Sweeney (1998) as the concept where an attempt to use QIDs (see Table 2.1 for definition) to identify an individual should result in at least k entries. Further anonymization techniques have been proposed since the introduction of k-anonymity in 1998, as a result of discovered loopholes in the ever-improving anonymization techniques. For example, l-diversity was introduced after the detection of missing attribute disclosure protection in k-anonymity Machanavajjhala et al. (2007).

Differential Privacy

Differential Privacy was created for the purpose of protecting the privacy of individuals in a statistical database, in addition to being able to learn general properties about the population as a whole Dwork (2006). The essence of differential privacy is that the statistical outcome of a given query should be unaffected by the presence or absence of an individual entry. In the recommender system environment, differential privacy protects against membership inference and is thus used as a protection mechanism. Differential privacy is achieved by mathematically applying noise to the original data.

Cryptography

Cryptography entails the concept of "secret writing", where the secret writing is a result of a transformation of the original text given a certain secret. Cryptography is generally more related to security than privacy, but if user data is secured such that only confidential users can access it, that may lead to the protection of user privacy. Homomorphic encryption is one of the most applicable encryption systems for recommender systems, as it allows for mathematical operations on the encrypted data. In other words, data can be processed while being unreadable to adversaries, while the general user of a system is unaffected by the encryption. Even though the concept of cryptographic recommender systems is good, it is resource extensive and may therefore be slow and energy-demanding.

Obfuscation

Obfuscation is the concept of adding noise to original data with the purpose of hiding user preferences. "Obfuscation" describes creating something that is obscure and indistinct. Strategies used when performing obfuscation often include a combination of data randomization, insertion, and removal. In recommender systems, obfuscation is used to

model noise based on existing signals to make the available data less valuable for an attacker. Obfuscation is described in more detail in section 2.5.

2.4 Attribute Inference Attacks

One attack of importance in this work is the attribute inference attack. Attribute inference attack describes the attack where an adversary uses publicly available data to infer information about users of a system, e.g. sensitive attributes. Research has found that this publicly available data can be everything from rating behavior (Bhagat et al., 2013) and recommendation lists (Finjord, 2021) to social relations (Dey et al., 2012) and writing style (Rao et al., 2010). To perform the attack, the attacker first needs to train its classifier. This can be done by using sensitive attributes that some people choose to make public, in this case, gender, along with the data that is made public by all users. Eventually, the attacker will be able to predict the gender of users that would rather not share it. In other words, attribute inference attacks can be summed up as a method of using some users' public attributes to derive the private attributes of other users.

Attribute inference attacks can be grouped into two general types (Gong and Liu, 2018). The first type utilizes social links to infer information because the publicly available attributes of the people you are connected to often correspond to similar attributes for you. As an example, presented by Gong and Liu (2018), if more than half of the friends of a user u major in computer science at a certain university, the probability that that user u also majors in computer science at the same university is high. The other type of attribute inference attacks are attacks that base their predictions on your behavior. Similarities between users are based on similarities in their user interactions, resulting in the prediction that if you act similarly to another user, it is likely that the two of you share attributes. Interestingly enough, this is exactly the assumption that recommender systems are built upon: it is likely that users who behave similarly will like the same items.

2.5 Obfuscation Techniques and Synthetic Data

In the field of computer science, obfuscation is a term used interchangeably with anonymization, tokenization, encryption, and masking techniques. However, in the context of this thesis, obfuscation concerns the task of controlled data substitution and noise insertion. This is in line with the definition presented by Brunton and Nissenbaum (2015): "Obfuscation is the deliberate addition of ambiguous, confusing, or misleading information to interfere with surveillance and data collection." Obfuscation is a technique that can be used to make it more difficult to infer private attributes because it presents real user data along with misleading user data.

What an obfuscation essentially does is introduce synthetic data into the original data, either by insertions or replacements. If we look at an example relevant to recommender systems, we can examine the user preferences before and after an obfuscation, as seen in Figure 2.6 and Figure 2.7, respectively. In the first figure, the historical user activity

only contains movies that are so-called "chick flicks" - movies typically liked by teenage girls. In the second figure, however, the watch history is infected with a more diverse set of movies, where it is less obvious that the user account belongs to a female. Nonetheless, the second figure does still contain both romantic movies and comedies popular around the same time period, because the recommender's main goal is still to provide the user with relevant items.

Figure 2.6: Watch history from a typical female user before obfuscation.

Some general characteristics that should be fulfilled when doing a user-oriented obfuscation, as presented in Slokom et al. (2021), are *understandable*, *unobtrusive*, and *useful*. The term *understandable* means that it should be possible to understand why there is a need for synthetic data and why the specific set of items have been added. As for *unobtrusive*, the user should not perceive the obfuscation as problematic, disturbing, or limiting. For example in a recommender system for movies, children should not be recommended movies created for adults only. Another example of a problematic situation that is against the concept of unobtrusiveness is if users get recommended something they find totally unethical, e.g. movies that romanticize animal abuse. Lastly, the obfuscation should be *useful* and not be in the way of relevant recommendations.

The task at hand when obfuscating data is to balance the trade-off between privacy and utility - how close to the original data can the synthetic data be without compromising the original data? And furthermore, how does one choose what entries to keep, remove and insert? There are different proposed strategies at hand. Following in this section, obfuscation by replacement is presented. Firstly strategies for selecting items to be removed are presented before some strategies for adding items are introduced.

Figure 2.7: A user profile that is obfuscated to be less gender indicative.

2.5.1 Removal Strategies

In general, the more items we remove, the more difficult it becomes to infer correct user data. However, by removing items, it is likely that the user experienced utility decreases too. Table 2.2 presents a collection of different selection strategies.

The first and simplest selection is the random selection, which samples a random set of items that make up the x items that are to be removed. This strategy does not implement any special methods to maximize the increase in privacy or minimize a decrease in personalization. However, it is a naive way to bewilder a potential attacker. The importance-based strategy, on the other hand, considers the fact that it is likely that removing items decreases user utility. By selecting items that are less important to the user, we can still keep the items that we are almost certain the user will like. The importance-based strategy can be implemented in different ways, where the simplest idea is based on the fact that recommender lists are sorted from best to worst, meaning that if we select the lower items in the list first, we are likely to keep the most personalized items. The last strategy in Table 2.2 is the class-based strategy. This strategy is based on the fact that some items are more indicative of a class than others. If we select the most class-indicative items, the idea is that the inference performance will decrease.

2.5.2 Insertion Strategies

Insertion strategies concern the selection of items from the catalog to be inserted into the original set of items. This is the synthetic data. One thing to note is that the synthesized data can also be or become indicative of a class. For example, by naively adding the same

Table 2.2: Removal strategies for user profiles.

Strategy	Description
Random	Items to be removed are selected at random.
Importance Based	The least important items are removed first. These are typically the items that relate the least to a user's preferences.
Class-Based	The items that indicate a given class the most are selected first. E.g. if we want to hide the gender of a female, the most female-indicative items are removed first.

male indicative item over and over in a female user profile, this inserted male indicative item may become indicative of a female. Moreover, if this naive obfuscation is performed, it is also possible to identify that the data is obfuscated based on unnatural values and occurrences in the data. Examples of insertion strategies are presented in Table 2.3.

In the table, the term classes are used. Each attribute we want to protect has two or more different classes, e.g. the attribute age may contain classes such as child, teenager, adult, and elder. The point of many of the insertion strategies presented in the table is to insert an item from a different class than which the user belongs to. Therefore, there is a need for classification methods that categorizes all users and items accordingly. If one were to hide more than one attribute, the insertion items chosen have to be from a class different from all the target classes for all the attributes to be protected. The class-based strategies presented in Table 2.3 are the majority strategy, random class-based, greedy, and sample. The two other presented strategies, popularity, and random, are selections based on the whole item catalog, without being restricted by specific classes.

Table 2.3: Selection strategies for insertion of items in user profiles.

Strategy	Description
Popularity	Choosing items that are rated by the largest number of users.
Random	Choosing a random item from the whole catalog.
Majority	Choosing items that are rated by the largest number of users in a specific class.
Random Class Based	Choosing a random item from a different class than the class of the user.
Greedy	Choosing the items that correlate best to a different class than the class of the user.
Sample	The items are sampled based on how well they correspond to a given class.

2.6 Classification Models

Classification models aim to use a set of known examples to fill in missing fields in data that look like the examples. By using such models, predicting classes and labels for data entries that are missing those classes or labels is made possible.

2.6.1 Logistic Regression

Logistic Regression (LogReg) is a classification model that can be used to estimate the probability of whether an instance is related to a class or not (Kleinbaum et al., 2002). The logistic function $P(\mathbf{X})$, which estimates the likelihood of the classification given a set of independent variables X_1, X_2,..X_n, is formulated in equation (2.2).

$$P(\mathbf{X}) := \frac{1}{1 + e^{-(\alpha + \sum \beta_i X_i)}} \tag{2.2}$$

The constant terms α and β represent initially unknown weights that need to be fitted according to the training samples. The training samples consist of values for each of the independent variables, along with the target value that symbolizes whether the instance belongs to the target class or not. All the β_i values are trained such that the model can differentiate between the importance of each of the independent variables.

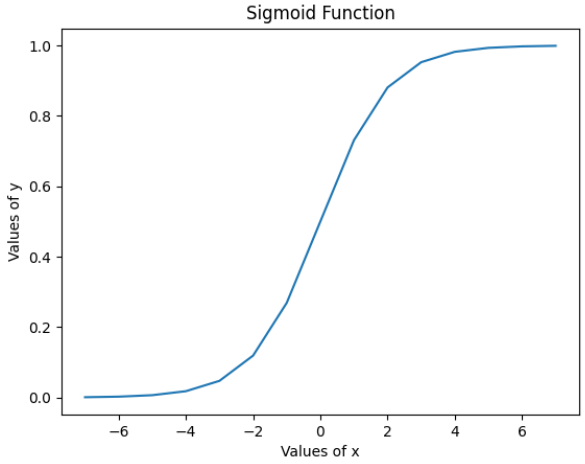


Figure 2.8: The Sigmoid Function.

A characteristic of the logistic function, also known as a Sigmoid function, that makes it fit for classification tasks, is that it converges around 0 and 1, making the range of the function $0 <= P(\mathbf{X}) <= 1$. First of all, this range can also be used to describe a probability as this also is a number between 0 and 1. In detail, the Sigmoid function describes a probability that approaches 0 when $-(\alpha + \sum \beta_i X_i)$ is $-\infty$, and 1 when $-(\alpha + \sum \beta_i X_i)$ is ∞. Secondly, the convergence identity gives the graph a distinctive S-shape where the lower bend symbolizes a threshold that is often evident in e.g. disease

conditions affected by a set of independent variables. The shape of the Sigmoid Function can be seen in Figure 2.8. After the lower threshold is reached, the probability of whether the item belongs to the target class increases much faster for each increase in the value of $\sum \beta_i X_i$.

2.6.2 Naive Bayes

The Naive Bayes (NB) classifier is, as the name suggests, a classifier based on Bayes Theorem. Bayes Theorem describes the probability of an event A as given by the prior known and related event B - described by Equation (2.3). The theorem assumes that all included events are independent, which for the classifier translates to all features being independent.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{2.3}$$

For classification tasks based on Bayes Theorem, the A in the equation represents the target variable - meaning the final classification, and B the all the features - $B = (x_1, x_2, ..., x_n)$ where each x_i is a feature. With these new terms, the theorem can be described as shown in Equation (2.4). The classification is now reduced to the probability given the set of other similar events.

$$P(A|x_1, x_2, ..., x_n) = \frac{P(x_1|A)P(x_2|A)...P(x_n|A)P(A)}{P(x_1)P(x_2)...P(x_n)} \tag{2.4}$$

2.7 Data Splitting

In the scenario of classification, whether that is a classification of user attributes or assumed likings of items, a model is trained to perform the classification. To do this we need a set of labeled data that the model can learn from. This collected labeled data is often divided into three different sets: the training set, validation set, and test set.

As the name suggests, the training set is used for training the model parameters. This means updating the model weights such that when the training data is inserted as input data, the target values of the training data are returned as output data. The training set is traditionally the largest proportion of the whole dataset, compared to the validation and test set.

Where the training set is used to train the model parameters, the smaller set called the validation set is used to tune other kinds of parameters, called hyperparameters. Hyperparameters are typically set when the model is created, for example the selection of loss function or regularization strength. During hyperparameter tuning, however, different configurations of these hyperparameters are tested to find the best hyperparameters for the given task. The "best" model is selected based on the performance of a selected evaluation metric, such as accuracy or F1.

Following this phase of hyper-tuning, the model is to be tested. The test data should be separated from the other data such that is "unseen" when the final model is tested

with this data. If we were to test our model with the data from which the model was trained, we would not be able to test the generalizability of the model. The test set can also be used to compare different trained models to each other.

There are different strategies for splitting a dataset to use it for training, hyper tuning, and testing. Two such strategies are fixed splits and K-fold cross-validation.

2.7.1 Fixed Split

A fixed split is a separation of the complete dataset where the subset is kept from each other and not altered after the split. One way of splitting the dataset in recommender systems is by keeping the oldest interactions in the training and validation set and newer interactions in the test set - known as a temporal split. This will reflect the real world because based on earlier interactions, we try to predict future interactions.

Another way of dividing into subsets is by randomly selecting data to be added to each of the sets. This can be useful when we either lack temporal data or when time is irrelevant. One important thing to think about with random splits is to make the subsets as similar to the original dataset as possible. For example, if the dataset used for binary classification is skewed and there are a lot more instances belonging to the "yes"-class compared to the "no"-class, we might end up with only "yes"-classes in the training set. As a result, the model only learns about these "yes"-classes. To mitigate this problem, we can stratify the split to make sure that there is approximately the same proportion of "yes"-classes in the training set as there is in the complete dataset.

2.7.2 K-Fold Cross-Validation

In K-fold cross-validation, the dataset is split into k subsets that alternate between being the training set and the test set. This strategy is especially useful when you want to evaluate a machine-learning model but have limited data. The alternation is done by having k rounds of testing and evaluating, where a new set serves as the test set for each round, as seen in Figure 2.9.

When using K-fold cross-validation to evaluate a model, the general performance can be found by averaging the evaluation metrics over all the K-folds. The advantage of this way of training and testing is that the final results are less likely to be victims of overfitting and bias.

2.8 Evaluation Metrics

When creating recommender systems and tools for them, there is a need for measuring performance. Since the recommendation of items can be modeled as a classification problem, the metrics used for recommender system evaluation are similar to those used in classification and regression modeling (Aggarwal et al., 2016). This section has two parts, where the first presents metrics that fit the recommender problem and the second one the attack scenario.

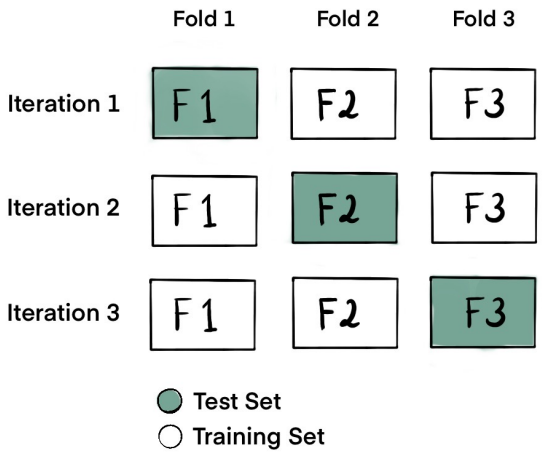


Figure 2.9: Illustration of K-fold cross-validation.

2.8.1 Evaluation of the Recommendations

The methods used for evaluating recommender systems can be separated into online and offline evaluation methods. Online evaluations are performed by exposing real users to the system, making it possible to measure the direct impact on the user. Examples of such online metrics are click-through rate and watch time. Although online evaluations are best to reveal user perception, most research either lacks an online platform where the system can be tested or they are restricted by the potential revenue loss of a system performing poorly (Krauth et al., 2020). As a result, offline methods are the most commonly used evaluation method. Offline methods look at historical datasets when evaluating the performance of a system. One of the advantages of offline evaluation is that it can better test the generalizability of a system by evaluating a system on different types of data. Additionally, it is a greater tool for benchmarking than online evaluation is (Aggarwal et al., 2016).

Since this thesis concerns top-K recommendations and not predicted ratings, the evaluation is based on these binary "relevant" or "not relevant" categories. Some metrics suitable for top-K recommendations are Precision, Recall, and normalized discounted cumulative gain (nDCG). Furthermore, a recommender system's performance is not only restricted to whether it can recommend items a user likes or not. It is also feasible that the system is able to recommend a diverse set of items and that it can recommend more than only the most rated items. A disadvantage of only recommending the top-rated items is that the recommender system only serves a small percentage of the items, meaning that a large proportion of the total item catalog is never presented. Consequently, metrics such as Item Coverage and Shannon Entropy should be considered.

Precision

Precision measures the proportion of relevant items among the set of top-K retrieved items, illustrated through the formula in Equation 2.5. Relevant items are items that are included as "liked" in the test set. Furthermore, the notation tp stands for true positives and fp for false positives. For simplicity, Precision@K - meaning precision at K recommendations - is written as P@K in this thesis. The metric is calculated per user, and can later be averaged over all users.

$$P@K = \frac{|Recommended\ items \cap Relevant\ items|}{|Recommended\ items|} = \frac{tp}{tp + fp} \tag{2.5}$$

Recall

Recall is the number of relevant items successfully retrieved from the whole set of relevant items. Similarly to precision, this metric is calculated per user but can be averaged over all users to find recall for the recommendation algorithm as a whole. Equation (2.6) displays the formula for calculating recall for K recommendations, denoted R@K. The notation fn in the formula represents the false negatives.

$$R@K = \frac{|Recommended\ items \cap Relevant\ items|}{|Relevant\ items|} = \frac{tp}{tp + fn} \tag{2.6}$$

nDCG

As opposed to precision and recall, Normalized Discounted Cumulative Gain considers the position of items in the recommender list. The metric penalizes wrongly recommended items placed early in the list more than wrongly recommended items on the bottom of the list. In other words, to get a high nDCG score, highly relevant items should appear early in the top-K list. To understand nDCG it is helpful to know DCG first, shown in Equation 2.7. The function of DCG returns the sum of the relevance of each item in the recommender list L for user u. The relevance is calculated as the ratio of the rating itself relative to the discount function $d(i)$. Typically, the discount function is defined as $d(i) = log_2(i + 1)$ where i represents the placement in the list.

$$DCG(L, u) = \sum_{i=1}^{|L|} \frac{r_{ui}}{d(i)} \tag{2.7}$$

The normalized DCG is defined as the DCG divided by the ideal DCG value - as seen in equation (2.8). This ratio will have a value between 0 and 1, where higher values symbolize a better top-K list. The advantage of nDCG over DCG is that the former simplifies comparing different queries. E.g. if we have two recommender lists with different lengths, normalizing the DCG value makes it easier to compare the two list's utilities.

$$nDCG(L,u) = \sum_{i=1}^{|L|} \frac{DCG(L,u)}{DCG(L_{ideal},u)} \tag{2.8}$$

Item Coverage

Item coverage is the fraction of different items that the recommender system is able to serve its users. In other words, it is the number of unique items across all recommender lists provided by the system.

Shannon Entropy

Shannon Entropy (shown in Equation 2.9) quantifies the inequality in item selection, and thus measures diversity. For this metric, we will get a number close to 0 if only the same items are shown in the recommender lists and $log(n)$ when all n items are occurring equally often (Gunawardana et al., 2012). The optimal value for Shannon Entropy is achieved when the distribution of recommendations is uniform, giving the value of $log(n)$.

$$SEntropy = -\sum_{i=1}^{n} p(i) * log(p(i)) \tag{2.9}$$

2.8.2 Evaluation of the Inference Attack

As mentioned, metrics that measure the performance of a recommender list can also be used to measure an inference attack. However, one has to consider the nature and characteristics of the data before choosing a fit metric. E.g. if your dataset is imbalanced, and most of the observations only involve users of one target class, a pitfall when evaluating the classification task is that a high number for one specific metric may not necessarily relate to a well-performing classifier. This is because we will get a high score even though the model only predicts the majority class in all cases. An example of a metric that works well on imbalanced datasets is F1-score, presented in this subsection. The metrics accuracy and AUC are also presented because even though they might not predict the inference performance as well in an imbalanced task, they give an indicator of the predictions.

F1-score

F1 is the harmonic mean of precision and recall. The reason why F1 is better for imbalanced datasets than precision and recall can be identified through a simple example. Let's say that in a group of 100 students, only five students are female. If our model always predicts a student to be a female this will give us precision = 5% and recall = 100%. Even though these metrics can be completely fine for some predictions, they are not fit for this skewed binary classification task. However, by finding the harmonic mean we will get a low performance score if one of the values is low. In our example case, the F1 score would be 9.5%.

Equation (2.10) displays the formula for calculating the F1-score. When it comes to the returning scores of F1, the closer we get to 1 the better is the prediction.

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.10}$$

AUC

AUC is a metric that measures the separability of a classifier, meaning how well it predicts 0-classes as 0-classes and 1-classes as 1-classes. AUC stands for "Area under the ROC curve", where ROC is an abbreviation for Receiver Operating Characteristics. To explain AUC further, the concept of the ROC curve is essential to understand.

$$TPR = \frac{tp}{tp + fn} \tag{2.11}$$

$$FPR = \frac{fp}{tn + fp} \tag{2.12}$$

The ROC curve is a graph that plots the true positive rate (TPR, see Equation 2.11) against the false positive rate (FPR, see Equation 2.12). Figure 2.10 illustrates an example of how this graph may look. The value for TRP and FRP will vary for different sizes of the variable K, where K is the number of items in the recommender list (thereby called top-K recommendation). We can plot all these values for different Ks as a monotone graph. A characteristic of the ROC curve is that it is restricted by the coordinate points (0,0) and (1,1) because when $K = 0$ both FRP and TRP are 0, and the highest possible value for FRP and TRP are 1. This results in the maximum value of 1 for the Area Under the Curve (AUC), which occurs when all the true positives are correctly identified. In Figure 2.10 the AUC is illustrated as the blue shaded area.

Accuracy

In short, accuracy is the fraction of items that the model predicted correctly. This can, for binary tasks, be formally described as in Equation 2.13.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.13}$$

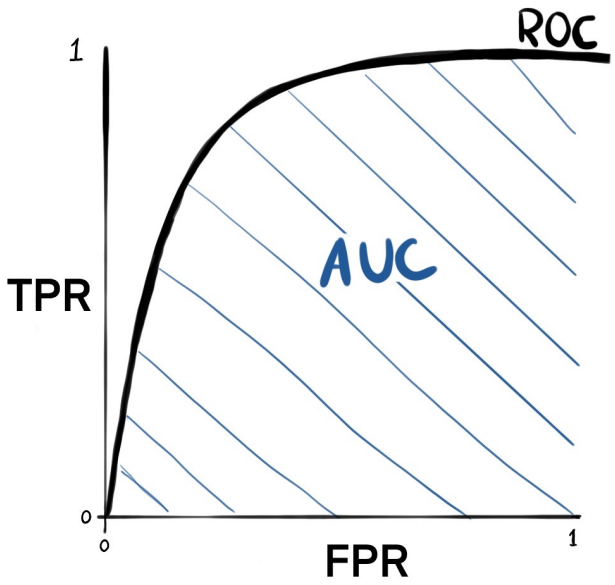


Figure 2.10: An illustration of a ROC curve with the AUC shaded in blue.

3 Related Work

This section is an introduction to recent work related to this thesis. Specifically, this includes a short introduction to privacy in recommender systems, attribute inference attacks, and how synthetic data has been added as a privacy measure in recommender systems.

3.1 Privacy in Recommender Systems

With the widespread use of recommender systems, personalization has become an essential part of several sectors. However, the type and amount of information collected are in conflict with the privacy of the users of the systems. Jeckmans et al. (2013) lists a number of concerns that arises when we allow the system to store and process this increasing amount of data. In summary, these concerns consist of but are not limited to, users not being aware of the vast information collection, the sale of user information being lucrative, and, as is the focus in this work, the recommendations may reveal information about the users.

The work of Martin and Palmatier (2020), discussing the consumer-retailer relationship, indicates that the two groups consisting of A) users demanding personalization and B) privacy-concerned users that are studied, are not mutually exclusive. (Bright et al., 2021) discusses the "privacy paradox" regarding the fact that social media users are actively sharing personal information while simultaneously being concerned about privacy. This paradox is also discussed in Psychoula et al. (2018), where it is found that users would continue using a given service they like, despite being aware of the privacy risk of that given service. Consequently, personalized systems themselves should implement some kind of privacy preservation to help the user with minimizing privacy risks.

3.2 Attribute Inference Attack in Recommender Systems

The efficiency and simplicity of attribute inference attacks have been studied widely. For example, Bhagat et al. (2013) used matrix factorization to learn information about a user's private attribute based on the rating of items. This is relevant for recommender systems because most of the collected data is historical ratings. The possibility of successfully completing an attribute inference attack based on user ratings is also presented in other works. One such example is Weinsberg et al. (2012) which proves the possibility of inferring a user's gender with the help of historical user ratings and a small amount of gender-labeled data.

Equally important are the findings where other data have been used to correctly infer sensitive attributes. A descriptive behavior of users, other than rating behavior, is their writing style. As a result, Rao et al. (2010) was able to foresee the sensitive attributes of gender, age, regional origin, and political orientation based on users' Twitter writing style. In the work, they used a stacked Support Vector Machine-based classification algorithm to do the classification, and in their results, they showed that gender could be predicted with an accuracy of 72.33 %. Moreover, at a recent competition held in 2019 (PAN, 2019) - where the task was to sort Twitter accounts into bots, females, and males - the accuracy of the prediction of gender was as high as 83.56% for tweets written in the English language. This indicated that the models have improved since the paper of Rao et al. (2010) was released. The improvement in the tweet classification environment is likely to apply to other classification tasks, as the models have improved in the last few years. In other words, without privacy measures implemented in systems that handle user information, with each machine learning improvement, it becomes easier to execute attribute inference attacks. Under these circumstances, it becomes essential to incorporate strategies for mitigation of attribute inference attacks in recommender systems.

What is more relevant for this thesis' work is the realization that also the recommender lists generated leak private information. In the master thesis of Finjord (2021), recommender lists are used to infer the location of users. The thesis utilizes well-known classification models such as Support Vector Machine, Logistic Regression, Random Forest, and ZeroR to predict the area of residence. The work of Slokom et al. (2022) is similar, by experimenting with and discussing the fact that private information "survives" from training data passed into a context-aware recommender to the system output (the recommendation list).

Another paper that looks at how the recommender list of a recommender system might leak information about the user is Xin et al. (2022). However, they differ from both other works and this work as they predict the historical behavior of a user, rather than sensitive attributes. To predict the historical behavior, they propose an attacker based on the transformer architecture that exploits the successful concept of self-attention.

3.3 Synthetic Data as Obfuscation Technique

A technique used to prevent attribute inference attacks is introducing synthetic data and noise into the original data. Hiding sensitive attributes evident in the user-item matrices by adding noise has been discussed and proposed in a vast amount of earlier works. The work of Weinsberg et al. (2012), after identifying the privacy issue of simple gender inference, attempts to make user-item matrices more privacy-preserving by adding ratings for items that are typical for the opposite gender. As a follow-up to this mentioned privacy approach, Strucks et al. (2019) extends the obfuscation by making sure that it is less obvious that obfuscation has taken place. To do this, they restrict the number of times an item can be added as an artificial rating. Furthermore, as a second improvement, Slokom et al. (2021) introduces "PerBlur". PerBlur implements a personalization of

the synthetic data added to better balance the privacy-utility trade-off. Slokom et al. (2021) also shows the potential of using obfuscation as a tool for improving fairness and diversity in recommender systems. These three mentioned works relates to this thesis' work as they all want to introduce obfuscation techniques to mitigate gender inference in recommender systems.

Liu et al. (2022) aims to obfuscate user-item matrices according to both privacy in general and users' privacy preferences. This is implemented by using a synthetic data generation module which takes in a selected item from the original user-item matrix and uses this item to synthesize a new item to be inserted. To select an item to be synthesized, the researchers utilize an attention mechanism to find the item that relates the least to the user preference. The results of this work show that removing items that contribute less to a user's preferences may be a way to minimize utility loss while decreasing privacy issues.

To support continuous and customizable obfuscation, Yang et al. (2018) proposes a two-phased obfuscation strategy for user activity preservation. The first phase is the obfuscation of a collection of historical data, which is executed when a user wants to start sharing its data with a third party. Instead of sharing the original user activity vector, the system uses a vector of a similar user, whose activity vector has less privacy leakage. The second phase is the online sharing phase, which enables real-time access to user data for the third party. Another approach that utilizes neighbors' data to provide privacy is the work of Luo and Chen (2014). Their work clusters users into neighborhoods before performing a perturbation technique on the aggregated user interests. Eventually, this data is fed into the recommender system. As a result, malicious attackers are left with only the group's preference data with high deviations if they get access to the data. However, these results are not applicable to the work in this master thesis, as a group's preferences may help with the attribute classification if there is more than a certain degree of homogeneity in each group.

The aforementioned obfuscation techniques concern obfuscation of historical data created by users. However, this thesis focuses on the fact that system-generated data may leak sensitive user data. Therefore, exploring the less explored field of obfuscation of recommender lists is relevant. Beigi et al. (2020) builds a model they call Recommendation with Attribute Protection (RAP), where the purpose is to recommend relevant items while simultaneously mitigating inference attacks of private attributes. Their experiment is conducted as a min-max game between two components: a private attribute inference attacker and a Bayesian personalized recommender. The work that is closest to this work however is the work of Xin et al. (2022). They perform obfuscation on the exposure data of a recommender system, meaning the recommender list, with the purpose of hiding the users' historical behaviors. The obfuscation is performed with two steps that each have their own set of strategies. First is the task of selecting which positions in the recommender list to be replaced, where they found that choosing the items that correspond the least to the user interests was best. Secondly is the task of choosing which items to add to the recommendations. The proposed strategies for the insertion are random selection, or to choose the items that are most interacted with (popularity based).

4 Dataset

In this thesis, the dataset MovieLens-100k[1] (ML100K) is used. To carry out the experiments of the thesis, the original dataset is processed and recommendations are generated beforehand. The following sections of this chapter present the dataset in general and also how it was pre-processed.

4.1 The MovieLens Dataset

There exist different variations of the MovieLens datasets (Harper and Konstan, 2015). First and foremost, the size of the datasets differ from each other, but also the time period of the retrieval and the attributes describing the users and items are varying. As the name may reveal, MovieLens contains rating data from a movie database, more specifically the MovieLens website [2]. The MovieLens website was developed as a tool for experimenting with recommendation services, with the goal of advancing the art of recommendations (PAN, 2019). As a result, the MovieLens datasets are heavily used in recommender system research.

MovieLens-100k was released in 1998, meaning that it is beginning to get old. However, it is one of the datasets from MovieLens containing demographic information, thus making it relevant for this thesis. The dataset contains 100,000 ratings of 1682 movies rated by 943 unique MovieLens users.

Table 4.1 presents the user attributes available in the dataset. The sensitive attributes "Age", "Gender", "Occupation" and "ZIP code" are attributes that may be inferred based on a user's generated recommendations. To anonymize the users, the UserIDs are an unidentifiable integer between the number 1 and 943. Among the 21 occupation groups, there are groupings such as "administration", "engineer", "librarian" and "student".

In Table 4.2 we find information about the movies in the MovieLens-100K dataset. There are 1682 different movies in the dataset, each described by its title, the release date, and a URL that links to the IMDB site of the movie. Furthermore, the movies are described by a set of genres such as action, documentary, and thriller. There are no limits on how many genres a movie can correspond to, but the list of possible genres consists of only 18 (19 if we include unknown) different genres.

The last informational table for the dataset, Table 4.3, presents each rating action. The ratings are collected as explicit ratings on a five-star scale, meaning that the values are an integer between 1 and 5. As seen in the rating distribution diagram in Figure 4.1,

[1] https://grouplens.org/datasets/movielens/100k/
[2] https://movielens.org/

Table 4.1: User attributes available in the dataset.

User Data	
Attribute	**Description**
UserID	An integer between 1 and 943.
Age	The age of the user.
Gender	"M" for male and "F" for female
Occupation	One of 21 different occupation groups.
ZIP-code	The ZIP-code of the user.

Table 4.2: Movie attributes available in the dataset.

Movie Data	
Attribute	**Description**
MovieID	An integer in the range between 1 and 1682.
Title	The title of the movie along with the year of release.
Release date	The full date of the movie release
IMDB URL	The IDMB URL.
Genre	A list of genres that corresponds to the movie.

the most occurring rating is four stars. The figure also displays that the ratings recorded from the MovieLens platform include mostly positive ratings and not many bad ratings. Each rating action is also described by the time of the rating, which is useful when performing a temporal splitting.

Table 4.3: Attributes describing each rating available in the dataset.

Rating Data	
Attribute	**Description**
UserID	Integer
MovieID	Integer
Rating	An integer between 1 and 5 (five-star scale).
Timestamp	Seconds since the Epoch. Data is in the format of an integer.

In terms of gender diversity, the dataset is extremely unbalanced. The number of women present is under half of the number of men, as visualized in Figure 4.2. Although this might not affect the results of the obfuscations performed, it might affect the result of the recommendations. Another statistic for the dataset is the rating count of the 15 most rated items in the catalog, see Figure 4.3. The top four most-rated movies, corresponding Star Wars (1977), Contact (1997), Fargo (1996), and Return of the Jedi (1983), are rated by close to 2/3 of the user base. In contrast, the least-rated movies, consisting of around 140 movies, are only rated by a single user.

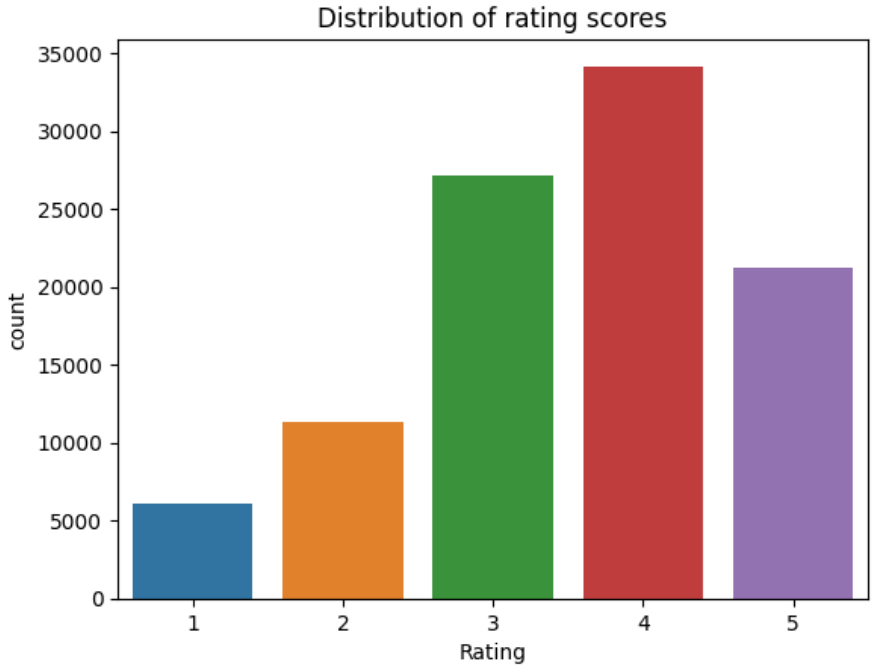


Figure 4.1: The distribution of the different numbers of stars given to the items in the dataset.

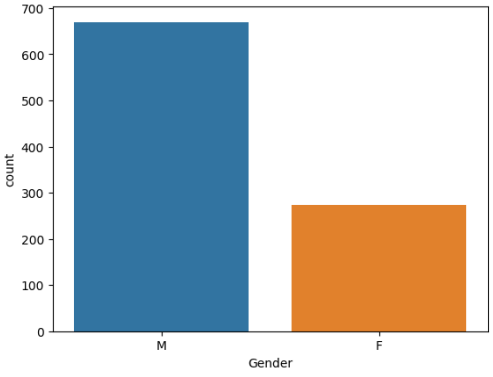


Figure 4.2: The gender balance of the MovieLens 100k dataset.

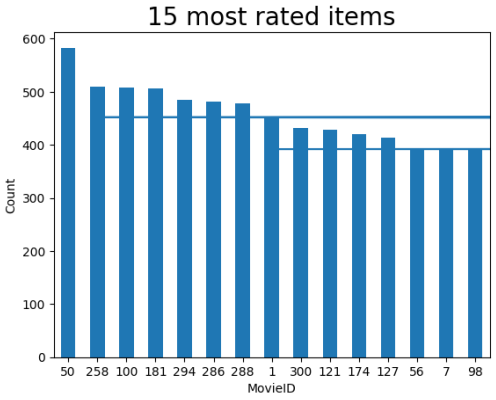


Figure 4.3: The most rated items in the dataset.

4.2 The Pre-processed Dataset

Because the work of this paper concerns the obfuscation of recommendation lists, the original dataset is processed to generate a set of recommendations for each user. This processing includes both an exclusion of some elements and a recommendation generation.

The methods presented in this section, along with the final recommendation lists, describe the work of the pre-processing of data as performed for the work of Slokom et al. (2022). The pre-processed data is obtained from the authors of that same paper.

Originally, as seen in the previous section, the original dataset consists of explicit ratings. However, to make the data fit the recommender model used, the ratings are transformed into implicit ratings. To do this, a threshold is set at the three-star rating score. For ratings $r >= 3$, the items are defined as relevant, leaving $r < 3$ as irrelevant items. To ensure that there are sufficient ratings provided by each user for predicting and testing their recommendations, users with less than 20 ratings are cut off. This leaves the subset containing 80962 interactions of 845 users and 1574 items. The distribution of ratings between males and females is very unbalanced, as seen in Figure 4.4.

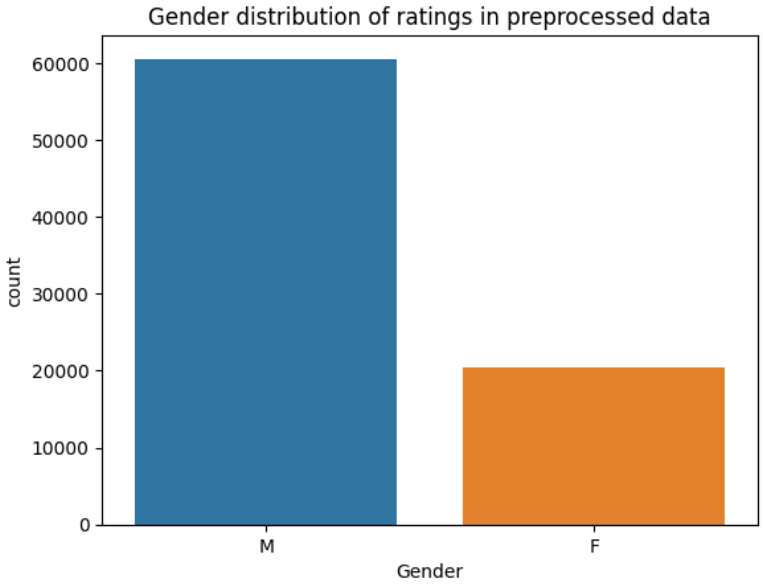


Figure 4.4: The distribution of the ratings from different genders in the processed dataset.

The recommendations are generated by using the RankFM implementation of Factorization Machine (FM). To reduce errors in the predictions, the model weights are learned with Weighted Approximate-Rank Pairwise as the loss function via Stochastic Gradient Descent. The data is split with a temporal splitting strategy, where the 10% most recent rating actions are used as the test set, whereas the rest is split such that the validation set consists of 10% of the data and the training set 80%.

In Slokom et al. (2022) it was found that the ML100K dataset had the best recommendation performance when the user attribute gender was used as side information in the FM. Moreover - and likely as a result - in a classification task of user attributes, the classifier performed best when trying to infer gender from recommendations generated with that same user attribute as side information. This is compared to the classification of age, occupation, and state based on recommendations generated with the respective user

attribute as side information. Consequently, as this thesis focuses on gender inference attacks, the recommendations generated with the gender attribute are used.

From this described pre-processing of the ML100K dataset, a list of the top 10 items is generated for each user. The recommendations are sorted such that the top recommendations are the items that it is most likely that the user will enjoy.

5 Method and Experiment

The goal of this section is to concretize the experiments conducted. This includes a presentation of tools and libraries utilized in the implementation, along with a more detailed description of the selected obfuscation strategies. Since the purpose of this master's thesis is to identify whether the obfuscation strategies do protect the user attributes or not, the inference attack strategy is also presented.

5.1 Tools, Libraries, and Repositories

Elliot Elliot is a recommender system framework that offers tools for handling the whole recommender problem, from dataset loading to performance measuring. As for performance measuring, Elliot generates reports that consider the experimental results.

NumPy Numpy is a Python library useful for scientific computing. The library offers numerous operations on arrays and matrices, in addition to comprehensive mathematical tools such as random number generation.

Pandas Pandas is a Python library developed for data analysis and manipulation. The library can be used as a tool to simplify, structure, and clean up datasets for simpler visualization and management.

PyCaret As stated on their website (PyCaret), PyCaret is a low-code machine learning library for Python that automates machine learning workflows. What this means is that it simplifies the implementation of machine learning algorithms, and speeds up the development cycle. PyCaret makes use of other popular machine learning libraries and frameworks such as, but not limited to, scikit-learn and XGBoost.

Scikit-learn The Scikit-learn machine learning library for Python provides a set of algorithms for classification, regression, clustering, and dimensionality reduction. Moreover, it includes other tools such as cross-validation splitting, evaluation metrics, and tools used for pre-processing.

User-based-Collaborative-Filtering The Git repository User-based-Collaborative-Filtering by ZwEin27 (2015) is an Open Source project that provides code for generating recommendations based on the User-Based collaborative filtering technique.

5.2 Attack Implementation

As the objective of this master thesis is to explore the mitigation of attribute inference attacks, there is a need for an attack implementation to test the success of an attack. By applying this attack to both the original and obfuscated recommendation lists, the degree of whether the obfuscations help or not can be measured.

It is assumed that the attacker holds a collection of recommendations and the corresponding user genders, such that he is able to train a classification model. This classification model is, in this thesis, first of all, considered to be a Logistic Regression (LogReg) model. The selection of the attacker is based on the results of Slokom et al. (2022), which found that the LogReg model performed well on gender inference attacks based on the same generated recommendations as used in this work. However, by using PyCaret, different classification models can fast and easily be compared to each other. In Figure 5.1, the performances of a set of different models (trained on the same recommender lists) are presented. The yellow cells indicate the best classifier based on the metric in that given column. The comparison in Figure 5.1 shows that in terms of F1, which is a good metric to look at when dealing with imbalanced data as this data, Naive Bayes (NB) is a good alternative. Thus, NB is also evaluated as one of the attacker's available models.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
nb	Naive Bayes	0.3915	0.5317	0.8447	0.2918	0.4335	0.0405	0.0728	0.2150
qda	Quadratic Discriminant Analysis	0.2757	0.5000	1.0000	0.2757	0.4322	0.0000	0.0000	0.2320
svm	SVM - Linear Kernel	0.6426	0.0000	0.3683	0.3652	0.3603	0.1158	0.1179	0.1870
lda	Linear Discriminant Analysis	0.6343	0.5724	0.3688	0.3507	0.3569	0.1034	0.1039	0.2760
ridge	Ridge Classifier	0.6709	0.0000	0.3252	0.3928	0.3509	0.1341	0.1376	0.1980
lr	Logistic Regression	0.6862	0.5839	0.2312	0.3904	0.2873	0.1034	0.1107	0.7610
dt	Decision Tree Classifier	0.6154	0.5083	0.2699	0.2860	0.2756	0.0160	0.0159	0.2100
lightgbm	Light Gradient Boosting Machine	0.6806	0.5462	0.1929	0.3706	0.2442	0.0690	0.0784	0.2650
knn	K Neighbors Classifier	0.6758	0.5306	0.1723	0.3205	0.2218	0.0455	0.0470	0.2350
ada	Ada Boost Classifier	0.6841	0.5590	0.1504	0.3094	0.1995	0.0420	0.0416	0.3100
et	Extra Trees Classifier	0.7030	0.5695	0.0598	0.2868	0.0931	0.0099	0.0121	0.3370
gbc	Gradient Boosting Classifier	0.6805	0.5557	0.0514	0.1345	0.0734	-0.0377	-0.0613	0.3950
rf	Random Forest Classifier	0.7196	0.5752	0.0303	0.2786	0.0521	0.0170	0.0321	0.2860
dummy	Dummy Classifier	0.7243	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2020

Figure 5.1: Performance of a set of different classifiers in the task of predicting gender based on recommender lists. The cells marked in yellow symbolize the best-performing classifier for that given metric.

To train and test the attacker of this work, the data is split into two parts: the training data and the test data. It is ensured that both the training and test set reflect the gender distribution of the original dataset, which is of importance since the dataset is highly

imbalanced. The training set consists of 80% of the users, while the test set consists of the remaining 20%.

After splitting the data, the model is trained and hyperparameter tuned by using a stratified k-fold with 10 folds on the training set. The hyper tuning is performed using the random grid search from scikit-learn with the F1 metric as optimize parameter. The results after hyperparameter tuning are visualized in Figure 5.2 for the LogReg model and Figure 5.3 for the NB model.

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
Fold							
0	0.6588	0.6417	0.5652	0.4062	0.4727	0.2304	0.2373
1	0.6118	0.5856	0.5652	0.3611	0.4407	0.1649	0.1747
2	0.6706	0.6168	0.5000	0.4286	0.4615	0.2263	0.2277
3	0.5882	0.5704	0.5000	0.3429	0.4068	0.1079	0.1125
4	0.5529	0.5499	0.4583	0.3056	0.3667	0.0421	0.0442
5	0.6071	0.6165	0.4348	0.3333	0.3774	0.0977	0.0995
6	0.6429	0.5588	0.5217	0.3871	0.4444	0.1897	0.1943
7	0.5833	0.5217	0.4348	0.3125	0.3636	0.0661	0.0681
8	0.6071	0.6101	0.5217	0.3529	0.4211	0.1402	0.1463
9	0.6071	0.5973	0.4783	0.3438	0.4000	0.1194	0.1230
Mean	0.6130	0.5869	0.4980	0.3574	0.4155	0.1385	0.1427
Std	0.0339	0.0348	0.0449	0.0375	0.0371	0.0608	0.0619

Figure 5.2: Performance after the hyperparameter tuning for the Logistic Regression model.

Since the goal is to test whether the obfuscated items leak user genders, the trained model is tested with both the original recommendations and the corresponding obfuscated recommendations.

5.3 Protection Strategies

The general strategy used to protect the gender user attribute in this thesis is recommender list obfuscation. Specifically, some items of the originally generated recommender lists are removed, and thus new items are added, with the goal of preserving privacy. This replacement strategy is selected because previous works, presented in chapter 3, have suggested that introducing noisy and new data into sensitive data can lessen the simplicity

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
Fold							
0	0.4824	0.5929	0.7826	0.3158	0.4500	0.1048	0.1452
1	0.4588	0.5375	0.7391	0.2982	0.4250	0.0641	0.0888
2	0.5529	0.6646	0.7917	0.3654	0.5000	0.1852	0.2315
3	0.4588	0.5205	0.7917	0.3167	0.4524	0.0822	0.1181
4	0.4471	0.6332	0.8333	0.3175	0.4598	0.0860	0.1320
5	0.3690	0.4729	0.6522	0.2500	0.3614	-0.0570	-0.0844
6	0.4762	0.6151	0.8696	0.3279	0.4762	0.1304	0.1974
7	0.3571	0.4968	0.6957	0.2540	0.3721	-0.0485	-0.0771
8	0.4286	0.5923	0.8696	0.3077	0.4545	0.0841	0.1405
9	0.4286	0.5784	0.8696	0.3077	0.4545	0.0841	0.1405
Mean	0.4460	0.5704	0.7895	0.3061	0.4406	0.0715	0.1033
Std	0.0533	0.0586	0.0717	0.0320	0.0412	0.0701	0.0994

Figure 5.3: Performance after the hyperparameter tuning for the Naive Bayes model.

of extracting sensitivity characteristics. Furthermore, as this thesis concern recommender lists, it is relevant that all lists are of the same length - resulting in a replacement being the appropriate choice as opposed to only removal or insertion. However, since replacement can be seen as a combination of removal and insertion, removal and insertion are presented separately in this section.

5.3.1 Traditional Obfuscation Strategies

The traditional obfuscation strategies are the strategies that are applied as obfuscation strategies in user-item matrices in previous work. The implemented strategies are a sample of the presented strategies from section 2.5.

Removal Strategies

The first part of replacement is to find items to be removed. Characteristics that are advantageous for the items to be removed are that they are the least user-typical items in the recommender list and the most gender-typical. For the selection of x items to be removed, defined as a percentage of the original recommender list length of 10, the following strategies are implemented.

Random Removal The random strategy is very straightforward. x items from the

original recommender list are chosen randomly.

Gender-Based Removal With the gender-based removal, the x items to be replaced are chosen based on the strength of the items' correlation to the user's gender. A prerequisite for this implemented strategy is a classification of typical male and female items. To separate the gender-indicative items into female and male items, users' historical ratings and their corresponding gender is fed into a LogReg model. The average coefficients based on 10 folds of the LogReg model fitting determine the degree to which each item correlates to one of the two genders - male and female. The items that correlate the most to female users are added to a list L_f, and the items that correlate the most to male users are added to a list L_m. This separation of genders is based on the code implementation of Slokom et al. (2021). L_m and L_f are ordered according to the strength of the correlation between a given item and the gender class.

The more gender indicative an item is, the more likely it is that this item is removed. There are two implemented options for gender-based removal, either **gender-random** or **gender**. The former option will select items randomly if there, eventually, are no more items existent in the intersection of the gender list and the user's recommender list. The latter only selects gender-indicative items, and if there are no more gender-indicative items, no more items are replaced, even if less than x items are selected.

Insertion Strategies

Insertion strategies are strategies for selecting new items to be inserted into the recommender list. The main goal for these strategies is to make the modified recommendation list less gender indicative, but it is also preferable that the inserted items correspond to the users' interests. Whenever we are using a presumed ideal recommender algorithm, the top k recommended items are sorted based on user relevance, where items higher in the list correspond to a more likable item for the user. With this knowledge, it is reasonable to assume that inserting the new items at the end of the recommender list will yield the best recommendation result.

Random Insertion With the random insertion strategy, items are selected randomly from the item catalog. This strategy has no guarantee for adding user-relevant items, as it inserts uncontrolled noise to the lists.

Popularity Insertion For popularity insertion, items are sampled based on the number of times they are rated. Items that are rated more often are more likely to be inserted into the new recommendation lists. There is often a reason why popular items, in this case, movies, are popular, and it is therefore likely that these added items are enjoyed by the users. By sampling the items, the same most popular item is not always chosen, making it a little less obvious that an obfuscation has taken place.

5.3.2 Obfuscation with k-Furthest Neighbors

To obfuscate on a less naive level, obfuscation with the help of k-Furthest Neighbors (kFNs) is proposed. The idea of kFN was first presented in Said et al. (2012), aiming to increase diversity in the recommendations - not focusing on privacy preservation. kFN is based on the phrase "The enemy of my enemy is my friend". The general idea is that items disliked by users that are dissimilar to you can be relevant to you. It is likely that these items are less known items - not the typical top items that most users have rated - thus resulting in a more diverse set of recommendations. The intention of adding kFN-based obfuscation is that it adds unexpecting items while simultaneously including some degree of personalization.

As presented in chapter 4, the pre-processed dataset used to generate the original recommendations in this thesis only considers the positive ratings, meaning ratings >= 3. Consequently, when implementing the kFN algorithm, the ratings under 3 need to be included. Therefore, for all users present in the processed dataset, any existent rating $r < 3$ for items present in that same dataset is retrieved. The retrieved bad ratings are then appended to the pre-processed dataset for the kFN-generation. The number of unique items and users in the dataset is thus preserved, but the number of total interactions has increased.

kFN is a modification of the more known k-Nearest Neighbor (kNN) algorithm, which is widely used in memory-based collaborative filtering. kNN finds the k most similar users to a user u, and based on these neighbors' highly rated items, the ratings for items unseen by user u are predicted. To calculate the similarity between two users, similarity formulas such as adjusted Cosine similarity and Pearson correlation is employed.

kFN is mostly used in the context of serendipity, where the goal is to recommend *unexpected* and *relevant* items. In this experiment, the hope is that this "unexpectedness" introduces less gender-indicative and known items. In terms of the performance of kFN, Said et al. (2013) conducted an experiment where they compared kFN to kNN based on precision/recall and a questionnaire answered by the users of the system. The results showed that in terms of precision and recall, kNN outperforms kFN. However, in their questionnaire, they found that a high predictive accuracy does not always correspond to high perceived usefulness for the end user, supporting that kFN is a suitable alternative.

$$Pearson = \frac{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \overline{r_v})^2}} \tag{5.1}$$

For the implementation of kFN, the dissimilarity between two users is calculated based on the inverse Pearson similarity. Instead of inputting neighbor v's real ratings r_{vi} and $\overline{r_v}$ in the formula for Pearson correlation, as seen in Equation 5.1, the inverse ratings of user v, r'_v are inserted. r'_v is calculated as shown in Equation 5.2, where r_{max} is the maximum possible rating value, r_{min} the minimum value, and r the true rating.

$$r' = r_{max} - r + r_{min} \tag{5.2}$$

The recommendation generation based on the kFN algorithm in this thesis is performed

with the help of the presented User-Based-Collaborative-Filtering repository ZwEin27 (2015). As that project only handled the prediction of a given item for a given user based on kNN, the code had to be modified to fit the experiments in this thesis. These modifications included adding the kFN-algorithm and making the input parameters match the dataset of the thesis.

As it is desirable that the dissimilar users have opposite interests, the concept of minimum co-rated items is introduced. In contrast, if we only looked at dissimilar rating habits, two users would be seen as dissimilar if they have no ratings of common items. This is not in our interest, as we require rating pairs between dissimilar users. Hence, throughout this experiment, the minimum co-rated items variable is set to 3. This ensures that for each dissimilar user pair, both users must have rated at least three of the same items.

When it comes to the selection of which items to remove to make room for the new recommendations, the naive solution of removing the x last items in the recommender list is used. As recommendations traditionally are arranged in descending order - based on predicted rating -, this leads to removing the less relevant items from the original recommender lists. Then, the top x items of the kFN-generated recommendations are appended to fill the now remaining spots in the top-10 list.

5.4 Hypotheses

In short, the research questions consider the ability to use obfuscation of recommender lists to mitigate the simplicity of attribute inference attacks while simultaneously protecting the degree of personalization. The hypotheses for the presented protection strategies and attack scenario are summarized into the following:

1. As the goals of personalization and privacy are conflicting, the first hypothesis is that decreased inference attack performance - as a result of the obfuscations - comes with the drawback of decreased recommendation accuracy.

2. Since a property of popular items is that many people like it, it is assumed that insertion based on popularity will create a more well-performing recommender list than random item insertion.

3. Furthermore, as an extension to the previous hypothesis, since adding the most popular items presumably adds typical male items, based on the knowledge that around 60.000 ratings are provided by men and only 20.000 by women, the attack performance is assumed to be higher for the popularity item insertion.

4. For kFN, one can assume that less popular - meaning both lower-rated and less-rated - items will be added to the recommender lists, resulting in more diverse ones. With more diverse recommender lists, the hypothesis is that the classifier cannot draw a conclusion as easily as when users have similar recommender lists. Moreover, since related works of kFN have shown that it performs decently in terms of accuracy, it should do so in this experiment as well.

5. When it comes to the side effects, it is expected that both coverage and diversity will increase when new less personalized items are added.

5.5 Experiments

The experiments of this thesis investigate the possibilities of using obfuscation as a privacy tool for recommendations. Figure 5.4 illustrates the different combinations of removal and insertion strategies that are tested.

First of all, as this thesis' research questions reveal, the idea is to use previously known obfuscation methods. One such familiar strategy, used in e.g. Xin et al. (2022), is the popularity-based insertion. Additionally, popularity-based insertion is simple and a naive way to incorporate items that are liked to some degree by most users. This is similar to how a user can find new movies to watch without a personalized system. If a user looks at sites like IMDb [1], he can assume that a movie rated highly by a lot of users is a movie he will enjoy - although he has no knowledge of whether the other users are like him or not. In essence, this strategy aims at protecting personalization more than mitigating inference attacks.

When it comes to the selection of gender-based removal, one of the reasons why it is implemented is that this work is similar to the gender-based user-item obfuscation that is found in, among other works, Weinsberg et al. (2012) and Strucks et al. (2019). Their works introduce gender removal and/or insertion and have luck with it. Moreover, the hope is that this strategy will lower the attack performance by removing items that are highly correlated to a specific gender. To maintain some degree of personalization, this protection strategy is combined with popularity insertion.

The reason why gender-based removal is parted into two strategies is that for some users, the number of gender-indicative items is less than the items we want to remove. Consequently, it is interesting to compare the exclusively gender-based removal with a strategy that removes an additional amount of random items such that the same amount of items is replaced for each user.

The random strategies - both for insertion and removal - create a "baseline". First, we want the obfuscated strategies to perform better than the original, but we also want the goal-oriented strategies to perform better than a "zero strategy"-strategy, which randomness is. With random insertion and removal, we have no control over what is added or discarded. Nevertheless, it is likely that the strategies clear away human-based patterns that the attacker is trained on, lowering the attack performance. Likewise to gender-based removal, random removal is therefore combined with the personalization strategy "popularity insertion".

Lastly, kFN-based obfuscation is a way of introducing more structured and personalized recommendations into the recommender lists. Because the focus is on personalization, the items that correlate the least to the users' historical activity are removed first - this means the last items in the recommender lists. The assumption is furthermore that the

[1] https://www.imdb.com/

"nature" of kFN enables it to add unknown items that break the gender pattern existent in the original recommender lists. Overall, the strategy aims to increase privacy with the insertion strategy instead of the removal strategy, as contradictory to for example the "gender-based removal & popularity-based insertion".

For each of the cells in Figure 5.4 with a number inside, the number denotes the experiment ID. Black cells symbolize no conducted experiment. For each of the combinations, an obfuscation of 20%, 40%, and 60% is performed.

After the different obfuscations are performed, they are evaluated based on their recommendation accuracy (presumed user likability) and how well a gender inference attack performs on the obfuscated recommender lists. For comparison purpose, both the evaluation of recommendations and the inference attack is also performed on the original user recommendations.

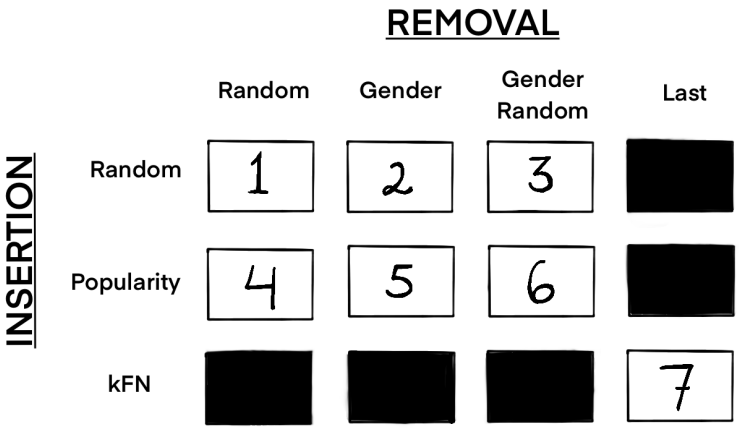


Figure 5.4: The combinations of configurations for the experiments. A black cell denotes that this combination will not be used.

In terms of performance measuring for the recommendations and attack, the following metrics are used to determine whether the obfuscations are useful.

5.5.1 Recommendation Performance

The following performance metrics are used to evaluate the performance of the obfuscated recommendations. They are all implemented with Elliot [2].

Precision Precision is selected as a performance metric because it describes the size of the relevant proportion of the recommender lists. The precision metric answers the question, "How many relevant items do the new recommender lists hold?".

Recall The recall metric gives an indicator of to which degree the new recommender lists manage to retrieve the relevant items from the test set.

nDCG To also evaluate the order of the recommendations, where the top relevant items should be presented first, nDCG is included in addition to precision and recall.

Item Coverage and Shannon Entropy (SEntropy) These metrics measure the diversity and coverage of the recommender lists. Adding random or semi-random items to recommender lists may also result in side effects. In this work, the potential side effects are measured in terms of diversity and coverage.

5.5.2 Inference Attack Performance

The attack performance is measured with the metrics described in this subsection, all implemented automatically with the help of PyCaret.

Accuracy Accuracy simply measures the number of correct predictions and is thus a relevant metric for classification tasks.

F1 Score Accuracy alone is not a good way to measure the performance of a system. Especially in these experiments, where the genders are highly imbalanced, high values for accuracy could mean that only the majority class is predicted. Consequently, the attack performance is also measured with an F1 score that better reflects prediction results on imbalanced datasets.

AUC Score AUC score is a metric that also looks at the falsely predicted items in its calculation. However, as the AUC score can be optimistic on skewed datasets, it should not stand alone as a metric in an imbalanced classification task.

[2] https://elliot.readthedocs.io/en/latest/

6 Results

The results from the experiments in this thesis are separated into two. The first part is the results from the recommendations with regard to the recommendation accuracy, and the second part the results of the attribute inference attack. In each of the separations, the results are presented in one table for each obfuscation strategy pair (removal and insertion). The tables display the results for the three different replacement proportions, namely 20%, 40%, and 60%.

6.1 Recommendation Performance

This section presents the results of the recommender performance for each of the conducted experiments. First of all, Table 6.1 illustrates the performance of the originally generated recommendations, for comparisons. The succeeding tables present the different obfuscation methods and the impact of the varying number of items replaced. Generally, we can see that a low degree of obfuscation results in more accurate recommendations for the users compared to a high degree of obfuscation. Simultaneously, the results show that with a higher degree of obfuscation, the recommender lists display more of the items in the catalog, thereby increasing the coverage.

Table 6.1: Recommendation performance of the original user recommendations.

Original Recommendations				
Precision	**Recall**	**nDCG**	**Item Coverage**	**SEntropy**
0.0818	0.110	0.110	422	7.66

6.1.1 Random Insertion

Table 6.3, Table 6.4, and Table 6.2 presents the results of the obfuscation strategies where new items are selected randomly from the whole item set. The first table, Table 6.3, illustrates the random removal case, whereas Table 6.4 displays the results where items are removed based on how related they are to the user's gender. Lastly, the results in Table 6.2 also present a gender-based removal selection, but here random items are removed if there are no more gender-specific items present in the recommender list.

Table 6.2: Recommendation performance of the random-random obfuscation of user recommendations.

Random Removal & Random Insertion (Ex. 1)					
Replacement	**Precision**	**Recall**	**nDCG**	**Item Coverage**	**SEntropy**
0.2	0.0684	0.0927	0.0982	1125	8.57
0.4	0.0520	0.0686	0.0805	1371	9.30
0.6	0.0359	0.0470	0.0629	1461	9.86

Table 6.3: Recommendation performance of the gender-random obfuscation of user recommendations.

Gender Removal & Random Insertion (Ex. 2)					
Replacement	**Precision**	**Recall**	**nDCG**	**Item Coverage**	**SEntropy**
0.2	0.0659	0.0886	0.0936	1150	8.55
0.4	0.0533	0.0696	0.0777	1368	9.17
0.6	0.0422	0.0566	0.0672	1426	9.49

Table 6.4: Recommendation performance of the gender with random-random obfuscation of user recommendations.

Gender w/ Random Removal & Random Insertion (Ex. 3)					
Replacement	**Precision**	**Recall**	**nDCG**	**Item Coverage**	**SEntropy**
0.2	0.0659	0.0894	0.0938	1138	8.54
0.4	0.0521	0.0687	0.0774	1377	9.24
0.6	0.0337	0.0454	0.0566	1455	9.81

6.1.2 Popularity Insertion

The tables presented in this section concern the obfuscation strategies where new items are inserted based on the general popularity of the item. Similarly to the former subsection, Table 6.6 concerns random item removal, Table 6.6 gender-based removal, and Table 6.7 the gender-based with added random removal.

Table 6.5: Recommendation performance of the random-popularity obfuscation of user recommendations.

Random Removal & Popularity Insertion (Ex. 4)					
Replacement	**Precision**	**Recall**	**nDCG**	**Item Coverage**	**SEntropy**
0.2	0.0696	0.0930	0.0970	709	8.14
0.4	0.0549	0.0761	0.0820	891	8.58
0.6	0.0419	0.0577	0.0675	962	8.94

Table 6.6: Recommendation performance of the gender-popularity obfuscation of user recommendations.

Gender Removal & Popularity Insertion (Ex. 5)					
Replacement	**Precision**	**Recall**	**nDCG**	**Item Coverage**	**SEntropy**
0.2	0.0672	0.0902	0.0942	710	8.11
0.4	0.0554	0.0727	0.0796	846	8.46
0.6	0.0464	0.0619	0.0702	906	8.65

Table 6.7: Recommendation performance of the gender-random-popularity obfuscation of user recommendations.

Gender w/ Random Removal & Popularity Insertion (Ex. 6)					
Replacement	**Precision**	**Recall**	**nDCG**	**Item Coverage**	**SEntropy**
0.2	0.0671	0.0901	0.0942	723	8.11
0.4	0.0535	0.0710	0.0786	849	8.49
0.6	0.0379	0.0506	0.0609	957	8.91

6.1.3 k-Furthest Neighbor-based obfuscation

The results for the k-Furthest Neighbor-based obfuscation are presented in Table 6.8 .

Table 6.8: Recommendation performance of the kFN-based obfuscation of user recommendations.

kFN Obfuscation (Ex. 7)					
Replacement	**Precision**	**Recall**	**nDCG**	**Item Coverage**	**SEntropy**
0.2	0.0708	0.0929	0.100	740	8.28
0.4	0.0570	0.0751	0.0888	801	8.65
0.6	0.0418	0.0544	0.0739	866	8.92

6.2 Attack Performance

In terms of the attack performance, two models were used to perform the attribute inference attack: a Logistic Regression (LogReg) and a Naive Bayes (NB) model. For the original recommendations, Table 6.9 displays the metric scores of the classification task performed by the two models. As can be seen from this table, the LogReg model performs significantly better than NB. The same performance gap is seen in the results for the obfuscated recommendations, and consequently, the NB results are presented in the appendix (chapter 8.0.4) for better readability. When it comes to the results of the attack performance, the general trend is that increased obfuscation leads to more incorrect guesses of gender.

Table 6.9: Attack performance on the original user recommendations.

Original Recommendations			
Model	**Accuracy**	**AUC**	**F1**
LogReg	0.8284	0.9199	0.7212
NB	0.5503	0.7265	0.5280

6.2.1 Random Insertion

Table 6.10, Table 6.11, and Table 6.12 presents the attack performance on the different obfuscation strategies where new items are inserted randomly.

Table 6.10: Attack performance on the obfuscated recommendations with random removal and random insertion.

Random Removal & Random Insertion (Ex. 1)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.7515	0.8279	0.6111
0.4	0.7396	0.7764	0.5111
0.6	0.7278	0.6802	0.4250

Table 6.11: Attack performance on the obfuscated recommendations with gender removal and random insertion.

Gender Removal & Random Insertion (Ex. 2)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.8284	0.9013	0.7071
0.4	0.8757	0.9297	0.7789
0.6	0.8521	0.9008	0.6988

Table 6.12: Attack performance on the obfuscated recommendations with gender-random removal and random insertion.

Gender w/ Random Removal & Random Insertion (Ex. 3)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.8284	0.9241	0.7129
0.4	0.8462	0.9165	0.6829
0.6	0.7751	0.8518	0.4722

6.2.2 Popularity Insertion

Table 6.13, Table 6.14, and Table 6.15 presents the attack performance on the different obfuscation strategies where new items are inserted based on their popularity.

Table 6.13: Attack performance on the obfuscated recommendations with random removal and popularity insertion.

Random Removal & Popularity Insertion (Ex. 4)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.7870	0.7881	0.6538
0.4	0.7219	0.7323	0.5053
0.6	0.6036	0.5680	0.3232

Table 6.14: Attack performance on the obfuscated recommendations with gender removal and popularity insertion.

Gender Removal & Popularity Insertion (Ex. 5)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.8521	0.9219	0.7525
0.4	0.8225	0.8696	0.7000
0.6	0.8343	0.8696	0.6818

Table 6.15: Attack performance on the obfuscated recommendations with gender-random removal and popularity insertion.

Gender w/ Random Removal & Popularity Insertion (Ex. 6)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.8343	0.9041	0.7255
0.4	0.8225	0.8671	0.6875
0.6	0.7633	0.7972	0.5238

6.2.3 k-Furthest Neighbor-based obfuscation

The attack performance for the k-Furthest Neighbor-based obfuscation is presented in Table 6.16.

Table 6.16: Attack performance of the kFN-based-based obfuscation of user recommendations.

kFN Obfuscation (Ex. 7)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.7574	0.7918	0.6095
0.4	0.7101	0.7241	0.4615
0.6	0.6864	0.6603	0.3614

7 Evaluation and Discussion

The presentation of results in tables is useful in itself, but to better generalize and compare the results, the attack performance measured in F1 score and recommender performance measured with nDCG are visualized Figure 7.1 and Figure 7.2, respectively. This chapter will describe and discuss the presented results and relevant findings.

7.1 Evaluation

The evaluation of the results from chapter 6 is in this section described in the light of the hypotheses from section 5.4, along with the obvious and interesting characteristics of the performance graphs. For simplicity, Table 7.1 summarizes whether the presented hypotheses hold or not.

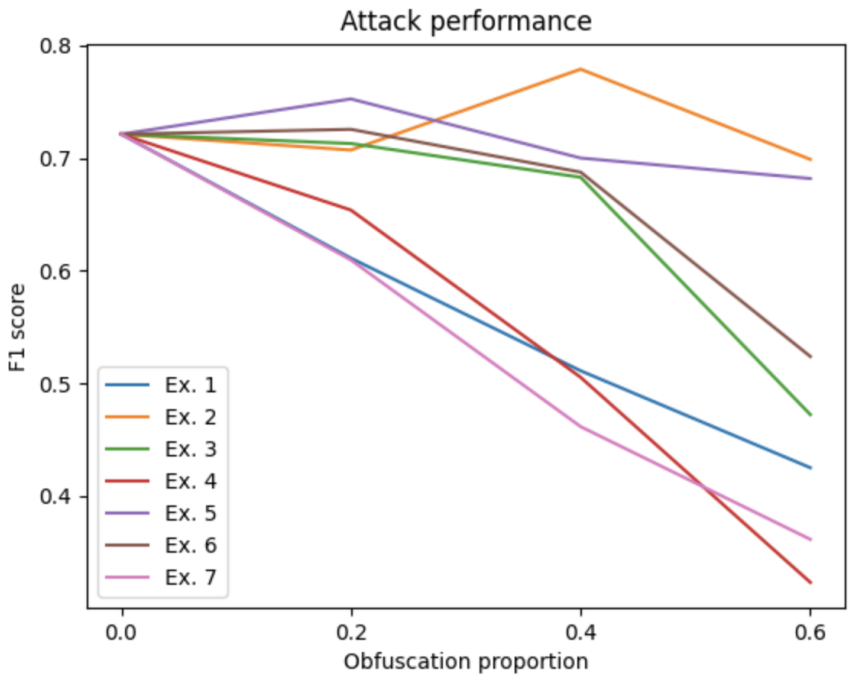


Figure 7.1: Attack performance measured with the F1 metric for the different obfuscation strategies.

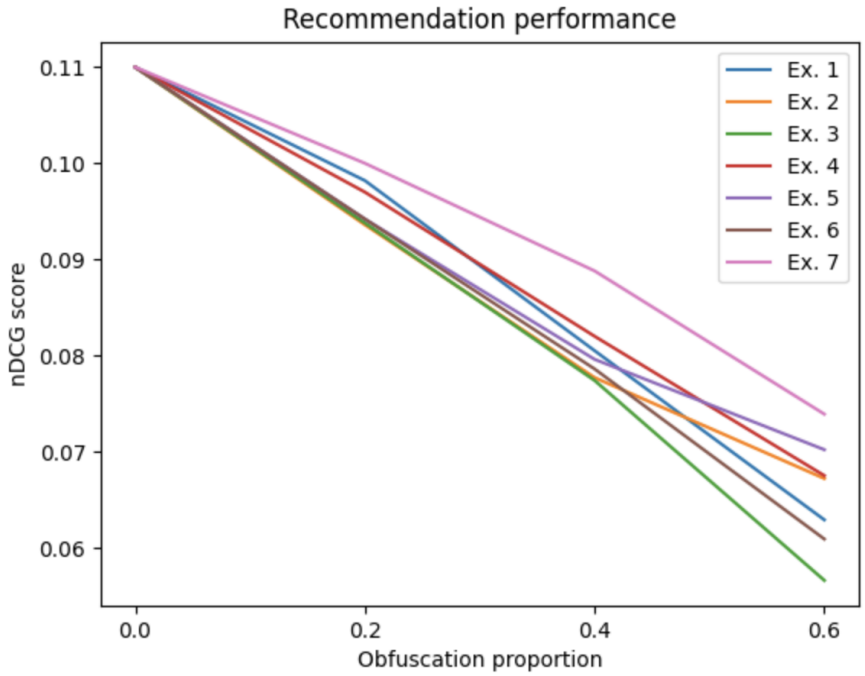


Figure 7.2: Recommender performance measured with the nDCG metric for the different obfuscation strategies.

By looking at Figure 7.1 and Figure 7.2 in conjunction, it is apparent that for this set of experiments, decreased inference attack performance is generally in accordance with less personalized recommendations (measured with recommendation performance). This supports the first of the hypotheses, that brings up the trade-off between privacy and personalization - an increase in privacy preservation corresponds to decreased recommendation accuracy. Nonetheless, by investigating Figure 7.1 further, it can be seen that an increased degree of obfuscation does not automatically lead to more privacy-preserving recommender lists. For experiments 2, 5, and 6, the inference attack performs better on some of the more obfuscated recommender lists than the original lists, on average. To conclude, even though there is a trade-off between utility and privacy, lower utility in the recommendations does not automatically mean higher privacy.

Although increasing the degree of obfuscation generally results in lower assumed user utility, some of the strategies perform better than others. This thesis' second hypothesis implies that inserting popular items will lead to better performance than adding random items will. The popularity strategies correspond to experiments 4, 5, and 6. If we compare each of these experiments with the experiments with the corresponding removal strategy, but the opposite insertion strategy (e.g. "random removal & popularity insertion" with "random removal & random insertion", etc.) we see that this hypothesis holds. To conclude, it is evident that the popularity-based insertion strategies outperform the

random-based insertion with regard to personalization.

Even though experiment 6 - "removal with gender-random and insertion based on popularity" - performs better than the corresponding strategy with random insertion, the recommender performance is worse than all other strategies. Moreover, the combination of gender and random removal (into the gender-random removal strategy) performs poorly also in terms of protection against inference attacks. The only removal strategy that gets higher inference accuracy, meaning less privacy protection, is gender-based removal. In essence, this is likely because both strategies remove the same gender items, but the strategy that does not remove an additional number of randomized items has a lesser obfuscation degree. On the other hand, this shortage of replaced items is advantageous for recommender performance. Altogether, it can be concluded that the implemented "gender"- and "gender-random"-based strategies are not beneficial strategies for privacy protection.

The third hypothesis questions the dataset's rating imbalance. It suggests that adding popular items will result in higher inference attack accuracy because many males have rated these popular items. Guessing male for all user's in the classification task will result in high attack accuracy because most users are men. In terms of the F1 metric, which can be seen in Figure 7.2, there is no indication of this hypothesis being correct. Since F1 is designed to work well on imbalanced data, looking at the accuracy metric to see whether popularity-based insertion performs worse in terms of privacy protection is also interesting. What the results show is that the accuracy of the inference attack is generally lower for popularity insertion, compared to random insertion. Consequently, hypothesis three can be rejected.

Experiment 7 - the "smarter" obfuscation method that tries to add personalized and unexpected items with the help of k-Furthest Neighbor (kFN) - gets some of the highest Shannon Entropy values compared to the other strategies. This high diversification score can imply that there is indeed some kind of unexpectedness and novelty in the recommendations. To clarify, diversity does not necessarily mean unexpectedness, but if users are shown different kinds of items in their recommender lists, it might result in them getting recommendations that they would not expect. Hypothesis four first assumes that this diversification may lead to lower accuracy of the inference attack. Figure 7.1 supports this, as the line for Experiment 7 is in the lower section of the attack scores. Furthermore, the hypothesis also refers to earlier works and indicates that the recommender performance for kFN obfuscation should be comparable to the original recommendations. Also, this part of the hypothesis holds, as the kFN-based obfuscation outperforms all the other strategies in terms of recommender performance.

What is interesting in Figure 7.1 is that even though Experiment 7 seems to be the lower bound of the attack score in each of the obfuscation proportions, at 60% obfuscation degree, Experiment 4 scores lower. Additionally, with reference to the recommendation performance, Experiment 4 does not perform badly. Anyhow, experiment 4 implements a "random removal strategy", meaning that for each randomly generated removal, we might remove either good or bad items. This means that even though this implementation and random item selection performs well in total, it does not explicitly ensure good

recommender performance or attack protection for each individual user.

Lastly, the side effects evident in the results of chapter 6 should be evaluated. The hypothesis that concerns these side effects implies that when we add new items, this will lead to more diverse recommendation lists, that use a larger set of the item catalog. When looking at the trend of item coverage and Shannon Entropy for increased obfuscation proportion, this hypothesis holds.

Table 7.1: Summary of whether the hypotheses hold or not.

Hypothesis	**Result**
H1	Holds.
H2	Holds.
H3	Does not hold.
H4	Holds.
H5	Holds.

7.2 Discussion

This section brings up topics that are relevant to this work, considering both the experiments with results and the general field of privacy in recommender systems. The goal is to discuss this thesis' findings in the context of earlier works and how they might be limited or affected by the choices taken.

7.2.1 Limitation of Offline Testing

Offline testing does not automatically measure the correct user opinions, because the assumed "likes" are only based on items that the user has interacted with, historically. Figure 2.2 from section 2.1 illustrates, in general, how users find new items to rate, which is by looking at items that the recommender system provides them. Often, the total item catalog is so large that it is difficult for a given user to navigate through the items. As a result, finding unknown items that are not presented by the recommender system is rare, but that does not mean that these unknown and not-shown items are all irrelevant to the user.

To repeat the property of the long tail, presented in subsection 2.1.1, there exists a few items that are rated extremely often while most items are rarely rated. Since many recommender systems utilize, at least to some degree, the concept of similarity between users, it is an inherent property that popular items are recommended more often. Under these circumstances, the probability that the historical data prove a long-tail item to be relevant is low. This affects offline testing because it results in test sets mainly consisting of ratings for popular items. As a result, it is difficult to measure the performance of serendipitous recommendations.

The kFN-generated items are based on the concept of serendipity and do therefore fall short in offline evaluations. As mentioned in subsection 5.3.2, the paper that this

implementation of kFN is inspired by, conducted a user study in combination with the traditional evaluation metrics. In their user-centric results, they found that the usability and relevance of the kFN-generated items scored similarly to those who were generated based on k-Nearest Neighbor (kNN). However, as earlier presented, the kNN-generated recommendations outperformed kFN in terms of precision and recall (traditional evaluation metrics). As a result, the generated inserted items of this work may be of more relevance to the user than what the traditional metrics suggest.

7.2.2 The Necessity of Obfuscation

Now, is it really necessary to modify people's recommendations at the expense of the recommendations' relevance? It is important to note that the collection of experiments in this thesis only presents a simplified example from the real world, which can be extended to more sensitive domains. For most people, revealing their gender is not something they worry about. However, if we are considering the health or news domain, and sensitive attributes such as health status or political orientation, it gets more serious. In light of this thesis' results, it can be seen that an attacker is able to infer gender with high probability, based on generated recommendations. The binary inference of gender is simple compared to, for example, disease inference where there are numerous different classes to predict. However, when combining user interactions with some background information, such as publicly known correlations about health, e.g. that obese people have a higher risk of getting diabetes type 2, an adversary may discover the person's health status.

To conclude, gender reveals based on movie history may seem harmless. Instead of removing stereotypical items, one can think that "stereotypical items are stereotypical for a reason, so why change recommendations?". Yet, if recommendations of a medical help site were revealed, and an attacker got access to conditions that were relevant to your earlier click history of the site, that could be harmful. As shown in this thesis, obfuscation could be a tool to better preserve the privacy of users.

7.2.3 Poor Performing Gender Removal

As commented in section 7.1, the gender removal strategy did not perform as well as intended. When it comes to recommender performance, it is intuitive that removing gender-based items may lower the user-perceived utility of the recommendations. As stated in the last subsection, stereotypical items are stereotypical for a reason. The purpose of this thesis is not about diving deep into the psychology of the different genders, so the stereotypical male and female behavior is not further elaborated. However, one theory behind the lack of personalization is that typical gender-indicative items are items relevant to the user.

In addition to reducing user-perceived likability, removing gender-based items also increased the attack accuracy. One assumption of why this happens is that when we insert new items, either based on popularity or randomly, the probability of these being typical male-rated items is high since the dataset consists of mostly male ratings. However, this

assumption alone does not hold, as all of the combination strategies also implement either popularity- or random-based insertion.

Figure 7.1 shows that by selecting "random removal" instead of "gender-based removal", the attack performance can be lowered. For each new obfuscation proportion, a new removal and insertion is carried out. This means that in six out of six randomized removals (Ex. 1 at 0.2, 0.4, and 0.6 in addition to Ex. 4 at 0.2, 0.4, and 0.6), the random removal strategy outperforms the gender-based removal. Altogether, the gender-based removal implemented in this thesis is not to be recommended.

A reaction to the poor privacy performing gender removal was to inspect the lists of gender-indicative items. The naive inspection performed was to make sure that the items of the gender indicative lists consisted of items that, for the human eye, looked fitting to each of the genders. Note that this is an extremely naive verification, as classification models may be able to extract information that is not as gender biased as the human eye is, in addition to finding correlations that are not as clear. The top three male indicative items found by the Logistic Regression (LogReg) model were "The American President (1995)", "Four Rooms (1995)" and "A Fish Called Wanda (1998)". For females, the top movies as presented by the item's coefficients were "Raise the Red Lantern (1991)", "The First Wives Club (1996)," and "I Shot Andy Warhol (1996)". The reader can look at the respective movie covers to either agree or disagree on what genders these movies have as their target group. Overall, it seems like the problem is rooted in the implementation of "gender-based removal", not the gender indicative lists L_m and L_f alone.

To sum up, the results and discussions indicate that items the attacker finds gender-indicative are not the same items that the female-male item-separator, presented in subsubsection 5.3.1, finds gender indicative. The two of them are both implemented as LogReg-models. However, they differ from each other because the attacker is trained on recommendations, while the item separator used in the gender-based obfuscation strategy is trained on historical ratings. This is likely why the gender-based removal strategy performs poorly in terms of attack mitigation.

7.2.4 Choice of Classifier

Compared to the work of Xin et al. (2022), the attacker in this thesis is very simple. There is no implementation of new and advanced statistical models; only the simple and traditionally used LogReg is utilized. If we look at what the work of Xin et al. (2022) tries to predict, it is a prediction of the sequence of earlier interactions. This ordered sequence requires a more advanced model than the binary gender classification of this work. There is no need to over-complicate the works, and the results of this thesis show that LogReg is sufficient and fit for the purpose of binary gender classification. As seen in chapter 6, in almost every obfuscation strategy and proportion, the LogReg predicts the correct gender for over 70% of the users. Consequently, for the sake of the purpose of this thesis - experimenting with the effect of different obfuscation strategies - LogReg is sufficient. However, what earlier works along with this thesis show is that there exists potential for attackers that want to infer more complicated information about users.

7.2.5 The Privacy Paradox

In chapter 3, the concept of "privacy paradox" is introduced. Users are sharing more information while simultaneously being more privacy-concerned. If one would like to participate in the digital world today, with the products and services that includes, it cannot be done without paying with your privacy. As mentioned in the introduction, there is an all-or-nothing mentality, where you either have to accept all terms or be unable to use the system. The paper discussing the all-or-nothing mentality, Chen et al. (2022), therefore proposes user control over personal data. This user control, which enables more selectable privacy protection, can be achieved by implementing different degrees of obfuscation for different users. The results of this thesis have shown that if one is willing to sacrifice recommendation utility, one can achieve more privacy protection.

However, this thesis only covers a small and simplified part of the possible inference attack implementations. For example, also discussed in chapter 3 is the vulnerability that lies in the relationship between users. If user u shares no personal information, but their friend, user v, shares everything, earlier works reveal that one can identify attributes of user u. This should be taken into consideration when implementing obfuscation, or other privacy-preserving strategies, into a system. Even though movie streaming services do not consider social relations traditionally, there exist other systems that build on the social structure of users.

7.2.6 Understandable, Unobtrusive, and Useful

Inspired by the reflections of Slokom et al. (2021) and mentioned in section 2.5, recommendations - and especially important when adding synthetic data - should be understandable, unobtrusive, and useful. Even though the experiments of this thesis have not directly focused on these three requirements, they are met to some degree. First of all, the obfuscation should be *understandable*. The reason why obfuscation is applied initially is to preserve users' privacy, which is understandable in itself. By adding fewer items that reflect the users' interactions, one can restrict the user characteristics that may be inferred through user recommendations. However, understanding why the specific items have been added can be more complicated. The implementation of intuitive obfuscation methods, as implemented in this thesis, can help to ease the understanding. First of all, for the item removal, removals are performed either by selecting the most gender-indicative items, by randomly picking a percentage of the recommender list, or by removing the last x items from the list. With the exception of randomly picking, these strategies can be justified and understood easily. If you want to hide your gender, you could hide the most gender-typical traits, and if you want to remove items that matter the least to you, simply remove the items that match the least to your preferences.

Over to the next concept, the concept of *unobtrusive*. There is no work in this thesis that ensures non-problematic or non-disturbing added items. In fact, at least one of the implemented strategies might actually increase the likelihood of adding problematic items. With the strategy of "k Furthest Neighbors", the goal is to find items that are disliked by a dissimilar user. If user u and user v have opposite interests, recommending

items disliked by v to user u may introduce user u to things he likes. However, some items are simply bad items - that would be disliked by everyone. These items may romanticize problematic topics or be of very bad quality. By introducing such items into the user u's recommender list, the recommendations for u may get obtrusive. For the popularity strategy, on the other hand, we have the reversed case. If an item is approved by numerous users, it is likely that the item is of good quality and non-problematic.

The last term presented is *useful*. One of the goals of this thesis is to preserve user utility in recommendations, meaning to preserve the usefulness of the recommender lists. The results of the obfuscations show that even though the usefulness of the recommender lists, measured in nDCG, has decreased compared to the non-obfuscated recommender lists, the recommendations are still somewhat useful. The definition of useful can also be extended to more than the traditionally measured match between historical ratings and generated recommendations. For example, increased diversity or more item-introducing recommender lists may also count as useful. In that case, the obfuscated recommendations in this thesis can be said to be useful.

7.2.7 Side Effects of Obfuscation

Visible in all tables in the recommender performance results, see chapter 6, is the fact that an increasing degree of obfuscation leads to a higher item coverage. This means that our obfuscation insertion strategies - including random insertion, popularity insertion, and kFN-based insertion - all manage to retrieve a higher percentage of the total item catalog compared to the original recommendations. High item coverage is favorable because it signifies that the items in each recommender list are different from each other and that the users get different items recommended than their neighbors. Moreover, a system that is able to present both popular items and less-rated items is a system that evades the pitfalls of the long tail. Despite the positive notions of coverage, if the coverage overrides performance measured by traditional metrics such as precision and recall, that is not favorable.

The item coverage of the original recommendations is 422, meaning that less than 27% of the item catalog is presented in the total of all users' recommender lists. For comparison, take a look at the catalog coverage of the overall best-performing obfuscation technique, meaning kFN. By replacing only two items in each of the recommender lists of length 10, this percentage is increased to 47%.

One reason why kFN manages to retrieve less known items is that popular good items are rated more often than popular bad items. In other words, if a system looks at items rated highly by users similar to you, it is likely that there exist many other users with neighbors who have rated the same items highly. Contrarily, if a system looks at items rated with a low score by users dissimilar to you, it is less likely that other users have neighbors with low ratings on the same items. The statistics of the pre-processed dataset consisting of both good ratings (>=3) and bad ratings (<3) show that the 3 items that have gotten the most 5-star ratings are rated by 300, 214, and 198 users, respectively. On the opposite side of the scale, the items that have gotten the most 1-star ratings are only rated by 39, 34, and 34 users. To conclude, the obfuscation strategies - here exemplified

as kFN - are able to contribute to increased item coverage.

Also to be mentioned in terms of item coverage is the high scores of random insertion. However, a high item coverage is not impressive if one randomly selects items. For the best-performing item coverage, found by implementing the "random removal and random insertion", replacing six items results in the total of all recommender lists presenting almost 93% of all items in the catalog. The relevance of items for this implementation (recommender performance), on the other hand, does not make up for the high item coverage score.

The other metric used to measure the side effects is Shannon Entropy (SEntropy). Similarly to coverage, the diversity also increases when there is an increase in the obfuscation degree. The same explanations used for item coverage can be applied to describe the increasing values for Shannon Entropy. With the increasing number of new, introduced items, some of the items in the original recommender lists are removed.

As the item coverage of the original recommendations reveals, the 845 users' recommender lists are filled with 10 of the in total 422 different items. In other words, the same items are repeatedly recommended for the different users. Altogether, when more items are introduced into the recommender lists (higher item coverage), the appearance frequency of the items will decrease.

There is a gap between the diversity score for the random-based and popularity-based insertion strategies, where the random-based strategy is superior. This is intuitive because adding random items is more likely to give diverse recommender lists than when adding popular items. This is also reflected in the item coverage. kFN-based obfuscation performs similarly to the popularity-based insertion strategy. kFN does not really operate similarly to the popularity-based insertion strategies, so it is more relevant to discuss why it performs worse than the random-based insertion strategy. First of all, this can be a result of the kFN-implementation removing the least relevant items first. If we follow the concept of the long tail, it is likely that the top items for each user are also recommended to a large number of other users, meaning that we remove the least overall popular items. Secondly, it can also simply be a result of the random insertion strategy being able to introduce a more diversified set of items, thereby being superior to kFN.

7.2.8 Research Questions

The goal of this thesis is summed up in the abstract description of "exploring techniques for more privacy-preserving recommendation lists". This main goal can be said to be fulfilled, as different techniques that have proven to create less gender-leaking recommender lists are proposed and tested. Moreover, the research questions presented in the introduction were the following:

1. To what extent is it possible to obfuscate recommender lists in order to lower the accuracy of attribute inference attacks by using techniques proven effective in the obfuscation of user-item matrices?

In terms of the first research question, first of all it is possible to utilize the given obfuscation techniques. However, not all of the tested obfuscation strategies justified

themselves to be useful, thereby resulting in different degrees of suitability. For example, the promising gender-based obfuscation technique discussed in Weinsberg et al. (2012), Strucks et al. (2019), and Slokom et al. (2021) did not perform well. In fact, for some degrees of obfuscation the attack performance of recommendations with gender removal performed worse than the original recommendations. In contrast to the previous works, this work exclusively removes gender-specific items. For comparison, Weinsberg et al. (2012) only adds ratings based on the typical items for the opposite gender. Yet, Slokom et al. (2021) also remove items based on the most gender-indicative items, which in their case leads to lower gender inference results. The differences between this work and Slokom et al. (2021) are that their work concerns user-item matrices, they use another dataset (Movielens-1M), and they also add items to the recommender lists by inserting items from the opposite gender's indicative list. A decisive comparison between my work and Slokom et al. (2021) is difficult to perform, because of the differences in data format and size. Whereas this work concern replacement of 20%, 40%, and 60% of lists of length 10 (the sum of all the lengths of recommender lists in the dataset is 8450), the other work replaces 1%, 2%, 5%, and 10% of the total user-item matrix (which contains one million ratings).

Another comment regarding this research question is that the introduced kFN-based obfuscation strategy was the overall best for minimizing the attack performance. This strategy is not included in the "existing techniques proven effective in the obfuscation of user-item matrices", but it is relevant to mention that it is an existing technique used for recommendation generation that is transferable to the protection domain. It was also apparent that the Random Removal-based strategies (Experiments 1 and 4) perform comparably to kFN, making also these fit for attribute inference attack protection.

2. In terms of recommendation performance, how do the obfuscated recommendation lists perform compared to the originals? Moreover, are there considerable differences between the explored obfuscation techniques?

Similarly to the previous research question, the kFN-based obfuscation performs best in terms of the overall presumed user evaluation. If we look at the precision of recommender lists obfuscated with kFN, we see that the obfuscation of 60% manages to present 4,2% relevant items per user on average. This number might seem very low, as this means less than half of a relevant item per user, but the original recommendations only come up with the average of 8.2% relevant items per user. Whether the obfuscated recommendations perform well enough or not depends on the acceptance of the recommender's performance.

When comparing the different strategies, one can first identify that seemingly all of the obfuscation strategies follow a linear graph that decreases with constant speed based on the proportion of replaced items. This makes it easier to compare the strategies to each other because the highest rate of decrease is the least favorable strategy. The gender-random replacement strategies perform the worst, along with Experiment 2: "Gender Removal & Random Insertion", whereas kFN is relatively superior to the other strategies. In the upper middle, we find strategies such as "Random Removal & Popularity Insertion"

and "Gender Removal & Popularity Insertion", where the popularity insertion makes itself useful by showing that popular items are indeed liked.

If we look at the answers to the research questions in combination, the overall best-performing obfuscation strategy, considering both low attack accuracy and high recommendation utility, is the kFN-based obfuscation. When 60% of the original recommender list is being replaced, the kFN based protection of the gender attribute achieves 50% of the original attack score, measured in F1. In terms of recommendation performance, this same obfuscation achieves 67% of the original score in terms of nDCG. The randomized removal strategies also perform well, but as they are randomized one cannot ensure that they perform well for each individual user.

Returning to the general goal and purpose of this thesis, are the techniques used in user-item matrices transferable to the recommender list domain? Yes, one could use the traditionally used obfuscation methods such as popularity insertion, random removal, etc. But, the most favorable is to look to serendipity and novelty research instead. Serendipity techniques, such as kFN is, balances unexpected and relevant in a way that looks promising also for the privacy domain.

7.3 Contributions

In this master thesis, I have presented a study of different obfuscation techniques used in recommender systems. The work has contributed to the understanding of how obfuscation can be used to protect user privacy in recommender lists. An important factor of the obfuscations was to avoid modifications that largely harmed the accuracy or quality of recommendations.

Through the research, I have identified several methods of obfuscation that can be used in recommender systems, where the most promising method is the kFN-based method. Moreover, the results also point out that an increased degree of obfuscation, where the obfuscation decreases personalization, does not necessarily correspond to more privacy-preserving recommender lists.

Overall, the work can be valuable as a starting point for a needed further research into the privacy protection of recommender lists.

8 Conclusion and Future Work

This Master Thesis concerns the privacy issue in recommender lists. Previous research has shown that recommender lists are vulnerable to attribute inference attacks, and thus this work has experimented with various obfuscation strategies to mitigate the accuracy of these attacks. More specifically, it tries to prevent gender inference attacks.

The implemented obfuscation strategies consisted of a set of formerly presented obfuscation techniques used in the obfuscation of user-item matrices. These strategies included random-based and popularity-based insertion of items, and removal strategies that include gender-based and random-based strategies. In addition to these simple obfuscation implementations, a technique for inserting serendipitous items was introduced. The reasoning behind this introduction is that serendipity in recommender systems is summed up in the two words "unexpected" and "useful". Unexpectedness of recommendations does affect the attacker because it results in less distinct gender patterns in the recommender lists. The serendipity strategy implemented is based on k-Furthest Neighbor (kFN), which builds on the idea that "The enemy of my enemy is my friend".

The findings from the results show that most of the presented and tested obfuscation strategies can be applied to recommender lists to lower inference accuracy. The strategies that performed badly were the gender-based removal strategies, because the implementation of them leads to higher gender inference accuracy along with a noticeable decrease in the recommender performance. On the other hand, in terms of good results, the serendipity strategy turned out to give a good decrease in inference accuracy while simultaneously giving little decrease in recommender performance, compared to the other strategies. Explained in percentages, the attack score for kFN, measured in F1, is as low as 50% of the F1 score for an attack performed on the original recommendations. In terms of recommendation performance, this same obfuscation achieves 67% of the original score in terms of nDCG.

Future Work

The work presented in this thesis is not a final answer to the privacy issue in the recommender system field, and there are numerous directions in which further development can follow. The future works proposed in this section suggest four tracks of various difficulties that could be interesting to look deeper into.

8.0.1 More Advanced Inference Attackers

Consistent with what was mentioned in the discussion, the attack strategy used in this thesis is not particularly advanced. It is a simple binary classifier that takes in a set of recommendations, and during training also the gender. However, one could look more into an advancement in the attacker, for example by utilizing context. Some recommender systems do differ between different contexts for the user, e.g. where there is a separation between recommendations given in the morning and recommendations given late at night. Moreover, one could utilize social relations more, given that the recommender system is used by a system that handles these relations. Such an advanced attacker is not relevant to the recommendations and data used in this thesis, but it could be relevant in real-world implementations of recommender systems. To implement this attacker, the adversary needs to know what type of contexts the recommender system differentiates between and whether it utilizes social relations. Finally, the purpose of this advanced attacker is to further test the effect of new obfuscation strategies.

8.0.2 Other Domains

Considering the binary, and for a lot of people insensitive, data that gender is, it would be interesting to look more into how obfuscation affects other domains. For example, what happens when obfuscations are employed in a news recommender system? Are the users still served relevant articles? Relevant for the news domain is that A) the article's time of publication matters, B) some articles should be presented to all users and C) the recommendations may leak more attributes that are classified as "sensitive personal information". Are requirements related to these properties fulfilled when obfuscation is added to the recommendations?

Another domain that might not fit with the obfuscation strategies presented in this thesis is location-based recommender systems. Looking at if it is possible, and in that case how, to use replacement strategies to hide users' location evident in their recommender lists. For location-based systems, adding recommended items that are far from the user is highly irrelevant, making it difficult to hide the location property.

8.0.3 Improve Obfuscation Strategies

Most of the obfuscation strategies in this thesis are based on very naive premises. A further direction for work within the field of privacy in recommender lists is therefore further development of obfuscation techniques. Since the serendipitous strategy introduced both more privacy and new, interesting recommendations, there could be more relevant techniques to be found in the field of serendipitous recommender systems.

Furthermore, improving the gender-based strategy is a possible direction. Since the gender-based implementation in this thesis seemingly failed to provide privacy-preserving recommender lists but has shown to be useful in previous work, it could be worth the try to further improve this technique for recommender lists.

8.0.4 Test in Online Evaluation

As a last proposed continuation of this work, testing the serendipitous obfuscation, along with the other strategies as well, in an online environment could be interesting. By conducting a user study or finding the conversion rates, that work could measure the real recommender performance concerning user likability.

Bibliography

Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

Ingebjørg Barthold. Privacy preservation in recommender systems: Exploring possibilities and limitations. 2022.

Ghazaleh Beigi, Ahmadreza Mosallanezhad, Ruocheng Guo, Hamidreza Alvari, Alexander Nou, and Huan Liu. Privacy-aware recommendation with private-attribute protection using adversarial learning. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 34–42, 2020.

Smriti Bhagat, Udi Weinsberg, Stratis Ioannidis, and Nina Taft. Recommending with an agenda: Active learning of private attributes using matrix factorization, 2013. URL https://arxiv.org/abs/1311.6802.

Laura F. Bright, Hayoung Sally Lim, and Kelty Logan. "should i post or ghost?": Examining how privacy concerns impact social media engagement in us consumers. *Psychology & Marketing*, 38(10):1712–1722, 2021. doi:https://doi.org/10.1002/mar.21499. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/mar.21499.

Finn Brunton and Helen Nissenbaum. *Obfuscation: A User's Guide for Privacy and Protest*. The MIT Press, 2015. ISBN 0262029731.

Ziqian Chen, Fei Sun, Yifan Tang, Haokun Chen, Jinyang Gao, and Bolin Ding. Studying the impact of data disclosure mechanism in recommender systems via simulation. *ACM Trans. Inf. Syst.*, oct 2022. ISSN 1046-8188. doi:10.1145/3569452. URL https://doi.org/10.1145/3569452. Just Accepted.

Ratan Dey, Cong Tang, Keith Ross, and Nitesh Saxena. Estimating age privacy leakage in online social networks. In *2012 proceedings ieee infocom*, pages 2836–2840. IEEE, 2012.

Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*, pages 1–12. Springer, 2006.

Miriam Finjord. Privacy in recommender systems: Can recommendations reveal your location? Master's thesis, NTNU, 2021.

Yingqiang Ge, Shuchang Liu, Zuohui Fu, Juntao Tan, Zelong Li, Shuyuan Xu, Yunqi Li, Yikun Xian, and Yongfeng Zhang. A survey on trustworthy recommender systems. *arXiv preprint arXiv:2207.12515*, 2022.

Neil Zhenqiang Gong and Bin Liu. Attribute inference attacks in online social networks. *ACM Trans. Priv. Secur.*, 21(1), jan 2018. ISSN 2471-2566. doi:10.1145/3154793. URL https://doi.org/10.1145/3154793.

Asela Gunawardana, Guy Shani, and Sivan Yogev. Evaluating recommender systems. In *Recommender systems handbook*, pages 547–601. Springer, 2012.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015. ISSN 2160-6455. doi:10.1145/2827872. URL https://doi.org/10.1145/2827872.

Yassine Himeur, Shahab Saquib Sohail, Faycal Bensaali, Abbes Amira, and Mamoun Alazab. Latest trends of security and privacy in recommender systems: A comprehensive review and future perspectives. *Computers Security*, 118:102746, 2022. ISSN 0167-4048. doi:https://doi.org/10.1016/j.cose.2022.102746. URL https://www.sciencedirect.com/science/article/pii/S0167404822001419.

Arjan JP Jeckmans, Michael Beye, Zekeriya Erkin, Pieter Hartel, Reginald L Lagendijk, and Qiang Tang. Privacy in recommender systems. *Social media retrieval*, pages 263–281, 2013.

Jerry Kang. Information privacy in cyberspace transactions. *Stan. L. Rev.*, 50:1193, 1997.

David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

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

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? *arXiv preprint arXiv:2011.07931*, 2020.

Fan Liu, Zhiyong Cheng, Huilin Chen, Yinwei Wei, Liqiang Nie, and Mohan Kankanhalli. Privacy-preserving synthetic data generation for recommendation systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1379–1389, 2022.

Zhifeng Luo and Zhanli Chen. A privacy preserving group recommender based on cooperative perturbation. In *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 106–111, 2014. doi:10.1109/CyberC.2014.26.

Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

Kelly D. Martin and Robert W. Palmatier. Data privacy in retail: Navigating tensions and directing future research. *Journal of Retailing*, 96(4):449–457, 2020. ISSN 0022-4359. doi:https://doi.org/10.1016/j.jretai.2020.10.002. URL https://www.sciencedirect.com/science/article/pii/S0022435920300658.

PAN. Bots and gender profiling 2019, 2019. URL https://pan.webis.de/clef19/pan19-web/author-profiling.html#results.

Ismini Psychoula, Deepika Singh, Liming Chen, Feng Chen, Andreas Holzinger, and Huansheng Ning. Users' privacy concerns in iot based applications. In *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1887–1894, 2018. doi:10.1109/SmartWorld.2018.00317.

PyCaret. Gpycaret. URL https://pycaret.org/.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. Classifying latent user attributes in twitter. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, pages 37–44, 2010.

Steffen Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pages 995–1000. IEEE, 2010.

Alan Said, Benjamin Kille, Brijnesh J Jain, and Sahin Albayrak. Increasing diversity through furthest neighbor-based recommendation. *Proceedings of the WSDM*, 12, 2012.

Alan Said, Ben Fields, Brijnesh J. Jain, and Sahin Albayrak. User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, page 1399–1408, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450313315. doi:10.1145/2441776.2441933. URL https://doi.org/10.1145/2441776.2441933.

Yasir Saleem, Mubashir Husain Rehmani, Noel Crespi, and Roberto Minerva. Parking recommender system privacy preservation through anonymization and differential privacy. *Engineering Reports*, 3(2):e12297, 2021.

Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. 1998.

Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, and Shaoping Ma. How good your recommender system is? a survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics*, 10(5):813–831, 2019.

Manel Slokom, Alan Hanjalic, and Martha Larson. Towards user-oriented privacy for recommender system data: A personalization-based approach to gender obfuscation for user profiles. *Information Processing & Management*, 58(6):102722, 2021.

Manel Slokom, Özlem Özgöbek, and Martha Larson. Gender in gender out: A closer look at user attributes in context-aware recommendation. *arXiv preprint arXiv:2207.14218*, 2022.

Christopher Strucks, Manel Slokom, and Martha Larson. Blurm (or) e: revisiting gender obfuscation in the user-item matrix. 2019.

Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. Blurme: Inferring and obfuscating user gender based on ratings. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 195–202, 2012.

Xin Xin, Jiyuan Yang, Hanbing Wang, Jun Ma, Pengjie Ren, Hengliang Luo, Xinlei Shi, Zhumin Chen, and Zhaochun Ren. On the user behavior leakage from recommender system exposure. *ACM Transactions on Information Systems*, 2022.

Dingqi Yang, Bingqing Qu, and Philippe Cudré-Mauroux. Privacy-preserving social media data publishing for personalized ranking-based recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31(3):507–520, 2018.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

ZwEin27. User-based-collaborative-filtering. `https://github.com/ZwEin27/User-based-Collaborative-Filtering`, 2015.

Appendix

Attack Performance with Naive Bayes

Table 1: Attack performance on the obfuscated recommendations with random removal and random insertion.

Random Removal & Random Insertion (Ex. 1)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.5266	0.7122	0.5122
0.4	0.5562	0.6230	0.4606
0.6	0.5444	0.5691	0.4122

Table 2: Attack performance on the obfuscated recommendations with gender removal and random insertion.

Gender Removal & Random Insertion (Ex. 2)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.5325	0.6065	0.4698
0.4	0.5444	0.6758	0.5096
0.6	0.5444	0.6273	0.4460

Table 3: Attack performance on the obfuscated recommendations with gender-random removal and random insertion.

Gender w/ Random Removal & Random Insertion (Ex. 3)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.5562	0.6798	0.4966
0.4	0.5266	0.6147	0.4118
0.6	0.5444	0.5466	0.3840

Table 4: Attack performance on the obfuscated recommendations with random removal and popularity insertion.

Random Removal & Popularity Insertion (Ex. 4)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.5858	0.6448	0.4853
0.4	0.5503	0.5963	0.4154
0.6	0.5148	0.5148	0.3492

Table 5: Attack performance on the obfuscated recommendations with gender removal and popularity insertion.

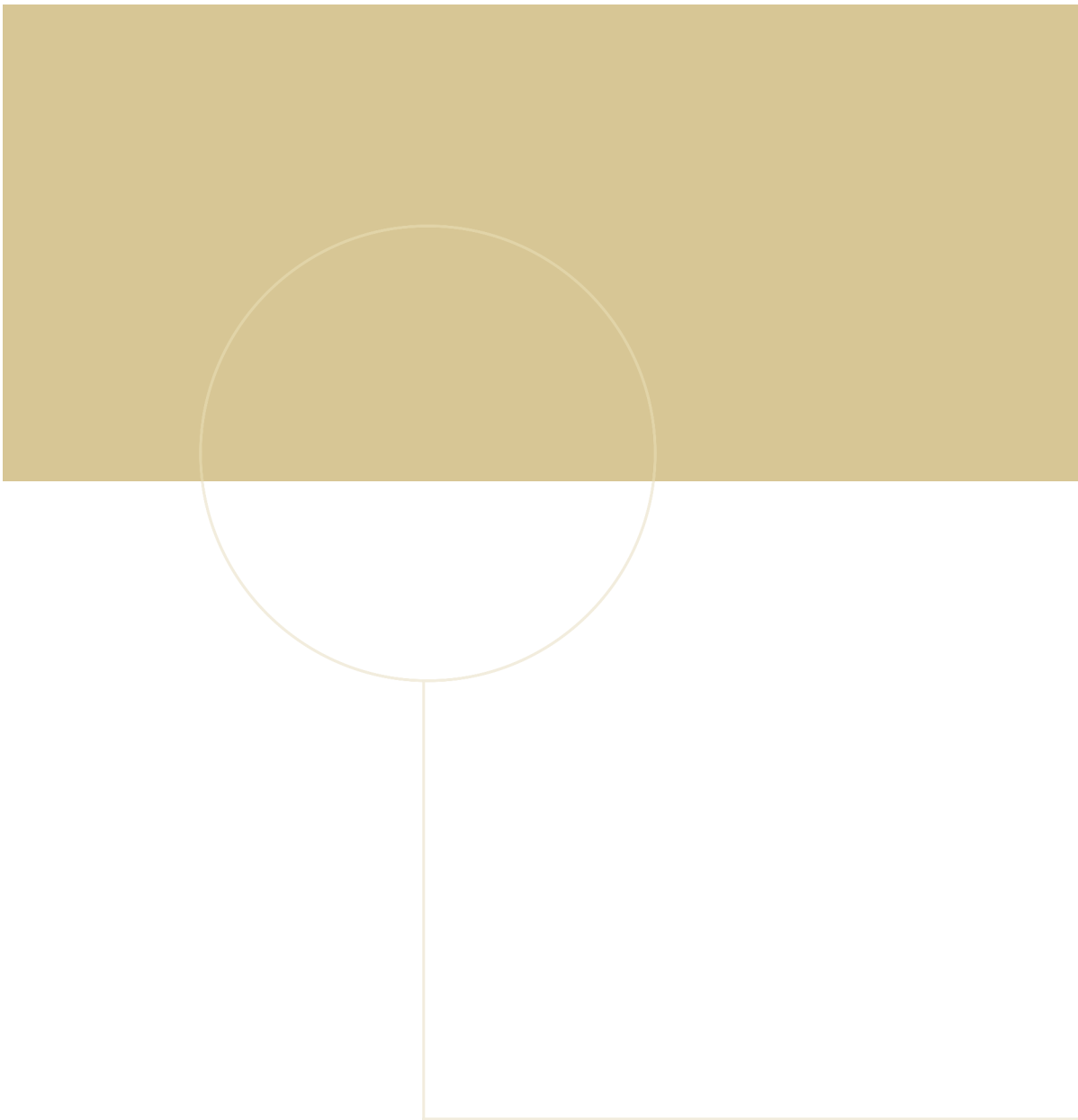
Gender Removal & Popularity Insertion (Ex. 5)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.5917	0.6691	0.5036
0.4	0.6154	0.6580	0.4882
0.6	0.5858	0.5924	0.4167

Table 6: Attack performance on the obfuscated recommendations with gender-random removal and popularity insertion.

Gender w/ Random Removal & Popularity Insertion (Ex. 6)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.5562	0.6428	0.7255
0.4	0.6746	0.6836	0.5378
0.6	0.5917	0.5845	0.3784

Table 7: Attack performance of the kFN-based obfuscation of user recommendations.

kFN Obfuscation (Ex. 7)			
Replacement	**Accuracy**	**AUC**	**F1**
0.2	0.5444	0.6379	0.4762
0.4	0.5444	0.6027	0.4296
0.6	0.5562	0.5477	0.3902



NTNU

Norwegian University of
Science and Technology