Master's thesis

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

Miriam Finjord

Privacy in Recommender Systems: Can Recommendations Reveal Your Location?

Master's thesis in Datateknologi
Supervisor: Özlem Özgöbek
July 2021

NTNU
Norwegian University of
Science and Technology

Miriam Finjord

Privacy in Recommender Systems: Can Recommendations Reveal Your Location?

Master's thesis in Datateknologi
Supervisor: Özlem Özgöbek
July 2021

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

NTNU
Norwegian University of
Science and Technology

NTNU – Trondheim
Norwegian University of
Science and Technology

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

TDT4900 - COMPUTER SCIENCE, MASTER THESIS

Privacy in Recommender Systems: Can Recommendations Reveal Your Location?

Author: MIRIAM FINJORD
Supervisor: ÖZLEM ÖZGÖBEK

July 2021

Abstract

The use of recommender systems has become a crucial part of the online experience and enables the user daily to navigate through the abundance of available services, products and information. However, recent works show that it is possible for an adversary to infer a user's private attributes from their recommendations and interaction history. To extend this research, the thesis aims to investigate inference of private attributes solemnly from personalized recommendations in the context of location information.

A set of experiments were performed, where the location attribute, as well as other private attributes, were attempted to be inferred using standard classification models. The recommendations were generated using a Factorization Machine model with Bayesian Personalized Ranking loss, and two well-known datasets: MovieLens 100K and BookCrossing.

The classification models were able to infer whether a user lives in the USA with AUC scores up to 82.7 %, but struggled when another dataset was utilized, where the scope of the location information was reduced to inside the USA. It can therefore be concluded that it indeed is possible to accurately infer information about the user's location from their recommended items, but that it will depend highly on the characteristics of the recommendations and the scope of the location information.

Sammendrag

Anbefalingssystemer har blitt en essensiell del av brukeropplevelsen online, og hjelper brukere til å navigere seg gjennom en overflod av tilgjengelig informasjon, produkter og tjenester. I denne sammenheng har en rekke akademiske artikler nylig blitt publisert, som viser at det er mulig å utlede privat informasjon om brukeren fra brukerens anbefalinger og tidligere interaksjoner med systemet. Denne rapporten ønsker derfor å bygge videre på dette arbeidet, ved å undersøke utleding av privat informasjon kun fra brukerens anbefalinger, hvor et spesielt fokus blir gitt til brukerens lokasjon.

I denne forbindelse ble en rekke eksperimenter ble utført, hvor brukerens lokasjon, samt kjønn, alder, og yrke, ble forsøkt utledet ved hjelp populære klassifiseringsmetoder. Anbefalingene ble generert ved bruk av en Factorization Machine med Bayesian Personalized Ranking tap, samt to kjente datasett: MovieLens 100K og BookCrossing.

Klassifiseringsmetodene var i stand til å utlede om en bruker bor i USA med en AUC score på 82.7 %. De strevde derimot mer når et annet datasett ble brukt, hvor målet heller var å utlede hvilken region i USA en bruker bodde. Det kan derfor konkluderes at det er mulig å utlede informasjon om brukerens lokasjon fra anbefalingene deres, men at det vil avhenge på anbefalingenes egenskaper og på hvilket geografiske nivå lokasjonen er forsøkt utledet.

Preface

This dissertation was written as a master thesis at Norwegian University of Science and Technology (NTNU). The research and experiments which formed the foundation for this thesis were conducted at the Department of Computer Science (IDI), under the supervision of Associate Professor Özlem Özgöbek.

First, I would like to thank my supervisor for the guidance and support she provided throughout the process, and for facilitating collaboration with other researchers in the field. Next, I would like to thank Martha Larson, Professor at Radboud University, for valuable insights and contributions to the project. I would also like to thank Manel Slokom, PhD student at Delft University of Technology(TU Delft), for the valuable contributions to the research, and for answering any question I had on the topic. Finally, I would like to thank my family and friends for their loving support during my studies, and for proofreading my thesis.

Contents

Figures

Tables

Chapter 1

Introduction

This chapter provides an introduction to the main aspects of the thesis, including motivation, project goal, and approach. At the end, an outline of the thesis is provided.

1.1 Motivation

With the growth of the Internet, an abundance of online stores and services has emerged, requesting users to provide private information in exchange for personalized experiences. Recommender systems have a number of various use cases; from eCommerce, through content recommendation and Social Networks, to health care applications. The aim is to assist users in navigating the overwhelming number of options available to them and make informed decisions, especially in situations where users lack sufficient experience or knowledge[1]. Although these systems have proved to be highly beneficial, they do not come without risk. In order to provide recommendations tailored to the users' preferences and needs, personal information is usually a necessity and is collected, processed and stored by the online service. The quality of recommendations depends on the amount of provided information, as well as its richness, reliability and recentness. However, the risk of personal data being exposed to third-parties and the damage it can cause, also develops in accordance with these factors. This is known as the privacy-personalization trade-off[2].

Recommender systems collect data from a huge amount of users, which allows users to learn personal information about each other, even if the users keep the information private. The threat is most evident in cases where two or more users have a shared account or service. As the recommendations are generated from the users combined interactions with the system, the recommendations intended for one user will give clues and insights about the other user's activities. Another situation where this threat can arise is in the case of a shoulder surfing attack,

where another person uses observation techniques to obtain information about the target user, e.g. by looking over the user's shoulder at their screen. An attacker can also obtain the users' recommendations through listening to the network traffic between the user and the service.

Literature mentions two types of information disclosure which can emerge from recommender system output. The first type is identity disclosure, which can be caused by a re-identification attack where the existence of a user is found in a public dataset[3]. The second type is attribute disclosure, which happens when users' private-attributes are inferred from public data. This is also referred to as an inference attack. Private-attributes contain information the user might not want to disclose, such as gender, age, and location, and can be used as auxiliary information to enhance recommendations. However, strengthening the relationship between private-attributes and system output can also enhance inference attacks. Works by Beigi et al. [4] and Zhang et al. [5] shows that private-attributes can be accurately inferred if the adversary has access to the user's recommendations and interaction history. A problem yet to be addressed is private-attribute inference attacks only based on a user's recommendations. The danger of this potential threat is how easily accessible recommendations are compared to other types of data, such as a user's interaction history.

Location disclosure attack is a subcategory of private-attribute inference attacks, which focuses on inferring location information of a user. The location of a user is considered sensitive information, as it can give information about the user's whereabouts and can be used to derive new information, such as the user's movement patterns, information about individuals or relationships, or identifying points of interest or sensitive places[3]. To infer this type of information it is common to either utilize the location of the target's friends and social ties [6–8], text-based information [9–11], or a combination of both [12, 13]. In addition, there are works focusing on inferring location as a private-attribute, based on users' publicly available item ratings [14] or public interests[15–17]. However, there seems to be no work focusing on inferring a users' location from their recommendations alone.

1.2 Project Goal and Research Questions

The goal of this thesis is to investigate whether users' location information can be inferred from personalized recommendations and how different aspects impact the accuracy level of the inference.

The research questions for this thesis are

RQ1 Information leakage: To what extent does information about the user's location leak through their recommendations?

RQ2 Dataset characteristics: How does the dataset's characteristic impact location inference?

RQ3 Data splitting strategy: How does the partition of location information affect the inference results?

RQ4 Classification models: Which classification model performs best for inferring the location information?

1.3 Approach

The goal of the experiments is to infer a user's location attribute from their personalized recommendations using multi-class classification. To generate recommendations, a Factorization Machine(FM) model[1] is used with Bayesian Personalized Ranking (BPR). For this classification task, the model is fed the user's top-50 generated recommendations and a set of user features, and will try to predict the user's location attribute. The covered classification models are Logistic Regression, Support Vector Machine (SVM), Random Forest, and ZeroR. The code used in this project can be found on GitHub[2].

To look at how the characteristics of a dataset affects the inference, two popular real-world datasets are used, namely MovieLens 100K[3] and Book-Crossing[4]. The characteristics include dataset sparsity, domain differences, and the scope the of location information. In the experiments, the user's location information is treated as a private-attribute, where three different splits were utilized for the MovieLens dataset; the *two-way split*, *five-way split*, and *ten-way split*, while the *USA vs. rest split* was utilized for BookCrossing.

1.4 Thesis Outline

This section gives an overview of all chapters, with a short description.

Chapter 1 - Introduction The current chapter intends to give an introduction to aspects of the thesis, including motivation, project goal, and approach.

Chapter 2 - Background A theoretical introduction to relevant theory and techniques for the following experiments.

Chapter 3 - Related Work An overview of published literature and important work related to recommender system privacy, private-attribute inference and location inference.

Chapter 4 - Data A description of the datasets used in the experiments.

Chapter 5 - Method and Experiments An overview of the different methods used, including data preprocessing, model implementation details and evaluation

[1] https://github.com/etlundquist/rankfm
[2] https://github.com/MiriamFi/rank-rec-infer.git
[3] https://grouplens.org/datasets/movielens/100k/
[4] http://www2.informatik.uni-freiburg.de/ cziegler/BX/

metrics.

Chapter 6 - Results A representation of the results from all experiments.

Chapter 7 - Discussion Comparison and discussion of the results, as well as their implications.

Chapter 8 - Conclusion A summary of the project, a conclusion of the discussion, and suggestions for future work.

Chapter 2

Background

This chapter provides an overview of the theory and techniques relevant to this thesis. First, an introduction to recommender systems and privacy aspect is provided. Next, a description of the recommender model is included, and at last the relevant classificaiton models are presented.

2.1 Threat Model

A threat model is a commonly used tool to discuss threats, adversaries, as well as suitable countermeasures. In this paper, a hypothetical attack scenario will be presented using a threat model. The following threat model is proposed by Salter et al. in [18], defined by a set of terms.

The first step of the threat model is to understand the motivation and opportunities of a potential adversary. First, the adversary's *resources* and *access* will be defined. Collectively, they describe the involved entities and resources which determine what an adversary is capable of doing in a situation. The *objectives* of the adversary are also described, e.g. the adversary wants to infer the location of a user. Once the adversary has become clear, the *vulnerabilities* of the system can be mapped. Lastly, the *countermeasures* in place to hinder the attack are presented. Usually, there is a correlation between vulnerabilities and countermeasures: when the vulnerability becomes more serious, the countermeasures must become stronger.

2.2 Recommender Systems

In the digital landscape overflowing with information and entities trying to get your attention, recommender systems are more popular than ever. Recommender systems strive to provide users with personalized recommendations, supporting the users by navigating and filtering content to fit the users' interests and needs,

and have become an integrated part of users' lives. For example, in 2015 Netflix revealed that approximately 80 % of the streaming hours on their platform were influenced by the system's recommendation algorithm, estimating the annual value of the platform to be more than $1B[19].

The recommender system process contains two main entities; the recommender system, which generates and provides recommendations, and the user, which interacts with the system. In order for the system to generate recommendations suitable for the user, it must have sufficient data about the user's preferences. This data is collected through the user's interactions with the system, such as clicks, likes, and reviews, and stored in the service. The process is displayed in Figure 2.1.

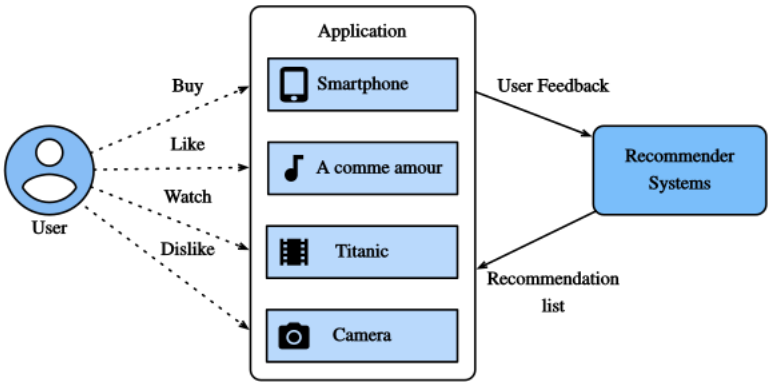


Figure 2.1: Recommender system and user interaction[20].

2.3 Recommendation Strategies

There are three most common recommendation strategies, namely Collaborative-filtering, Content-based filtering, and hybrid filtering, which is a combination of both[1]. Figure 2.2 displays an overview of recommendation techniques.

2.3.1 Content-based filtering

Content-based filtering generates recommendations based on analysis of item attributes, and proves to be the most successful when the items to be recommended are documents or other content types which are easy to describe, such as web pages and news. Both users and items are represented using attribute or feature vectors, e.g. user's gender, age, item category, and genre. The disadvantage of CBF is that it is a domain-specific algorithm, which makes business knowledge and feature engineering necessary.

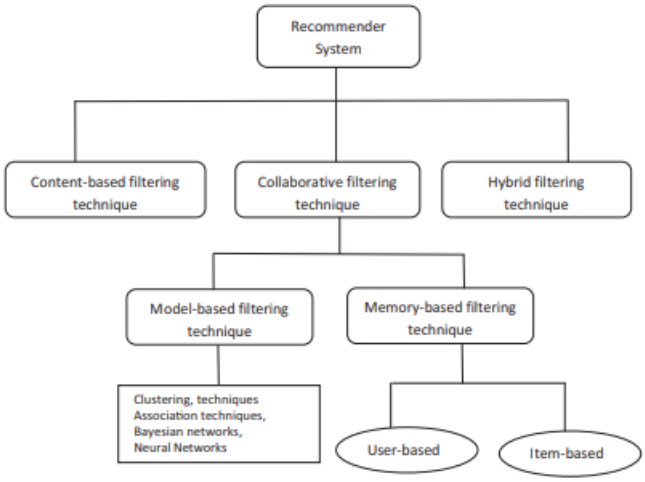


Figure 2.2: Recommendation techniques[21]

2.3.2 Collaborative filtering

The main idea of collaborative filtering is to provide users with recommendations based on what other users with similar tastes have liked in the past. The system identifies users with similar tastes by comparing their rating history, and finds items which might be of interest to the user, but they have yet to interact with. This is an efficient approach for recommendation systems in domains where content is not easy to describe or create meta-data for, such as music and movies. However, common problems like cold start and sparse data can have a negative impact on prediction accuracy. Collaborative filtering is currently highly popular and widely implemented recommendation strategy, and is used in big services like Amazon[22].

2.3.3 Context-Aware Recommendation

A regular recommender system generates recommendations based on the users' interactions with the system[1]. In this process, two types of entities are considered, namely users and items. Then, the rating function R can be described as the following for (*user*, *item*) pairs yet to be rated:

$$R : \text{User} \times \text{Item} \rightarrow \text{Rating}$$

where Rating is a totally ordered set, and User and Item are the user and item domains. First, ratings are predicted for all user-item pairs and then the recommender system can recommend the items with the highest ratings for each user. However, for some services and domains, contextual information can be utilized to enhance the recommendation process and provide recommendations more suitable to the user's current circumstances. For example, using a temporal context in online shopping. In the winter, the demand and which items are popular (ski

equipment, winter clothes, Christmas decorations, etc.) might be vastly different than in the summer (gardening tools, summer clothes, beach utilities etc.). Recommender systems which utilize contextual information in item prediction are called Context-Aware Recommender Systems (CARS). The rating-based representational approach to CARS makes two assumptions: First, the contextual information is known to the system and is defined by a set of contextual attributes, and second, these attributes will have an impact on the ratings. The rating function can be defined as the following:

$$R : \text{User} \times \text{Item} \times \text{Context} \rightarrow \text{Rating}$$

where Context is the contextual attributes defined for the system. In the past 20 years, academic researchers have applied contextual recommendations for different applications and domains, including movies[23], news[24], and travel guides[25]. Some businesses have also begun using contextual information into their recommendation systems, for example LastFM[1].

2.4 Privacy in Recommender Systems

In order to provide personalized recommendations, the recommender system needs to collect, store, and process information about the users [1, 26]. This information can include anything from the user's taste in music to their political view and health condition, and is used to create a user model, which will form the basis for the recommendation generation. The quality of the provided recommendations are dependent on some factors regarding the utilized user data. For instance, although some models thrive on sparse data, providing sufficient and rich data will increase the model's confidence to make suitable item predictions. In addition, it is of great importance that the system stays updated on the users' preferences, by receiving fresh new interaction data. High quality in terms of recommendations means that there is a high correlation between the users' preferences and the generated recommendations. This correlation can also be exploited by adversaries and increase the risk of a privacy breach. This is known as the privacy-personalization trade-off and and must be carefully considered when designing privacy solutions for recommender systems.

2.4.1 Privacy Risks

Privacy risk can arise in all phases of the recommender system's data management cycle, from data collection, through processing and storage, to dissemination. There are three main types of adversaries[1]. The first is the insider, which typically is a part of system management or has access to the system for other legal reasons. The intention of the insider is not necessarily to harm the user in any way, but rather due to other benefits. For example, by collecting excessive contextual

[1] https://www.last.fm/

Table 2.1: Privacy risks in recommender systems [1].

Adversary	Direct access to existing data	Inference of new data
Recommender system	Unsolicited data collection	Exposure of sensitive information
	Sharing data with third parties	Targeted advertising
	Unsolicited access by employees	Discrimination
Other users	Leaks through shared device or service	Inference from the recommender output
External entities	Lawful data disclosure	
	Hacking	Exposure of sensitive information
	Re-identification of anonymized data	

information about the user, or inferring new information about the users, with the goal to increase system performance or to sell it to third-parties, and thereby violating the users' expectation. For more information regarding privacy in context-aware recommender systems, see the work of Knijnenburg et al. [27]. The second type of adversary consists of other regular system users. As the system leverages information from an abundance of users, the users can learn personal information about each other through the system output, even if the information was meant to be kept private [28]. For example in the case of a shared account. Then, the interactions of both users will be used to generate recommendations, and as each user knows their past interactions, they will also be able to infer information about the past interactions of the other user. The third adversary refers to external entities. This can for example be hackers who gain unauthorized access to the system and commit data theft. Another scenario is if a third party received anonymized or pseudo-anonymized data from the system and is able to de-anonymize it. Table 2.1 displays an overview of potential risks in a recommender system.

2.4.2 Privacy Solutions

In order to mitigate the leakage of private information in recommender systems, some countermeasures must be considered. These techniques aims to protect the user's privacy and at the same time preserve system utility. There are three main categories of approaches. The first is refers to strategies related to system architecture and standards, which aims to minimize the threat of data leakage. This is done through protocols which serve as a guarantee to the users that the service providers follow practices to protect the users' privacy (e.g. distributed architectures) [29]. The second consists of different policies and regulations, which may hinder the service provider from any unsolicited collection, processing or sharing of private user-data. The third approach utilized algorithmic techniques for protection of the data. Some of these techniques which are relevant for the thesis will be presented in the next section. For more information on the topic of privacy in recommender systems, read the textbook by Ricci et al. [1].

2.4.3 Attribute Inference Defences

As the focus of this thesis is inference of private-attributes, some common mitigation strategies will be presented here. Currently, solutions for protecting the user against private-attribute attacks are yet to reach maturity. These approaches can be split into 4 different categories.

The first group [30–33] have implemented different solutions where cryptography algorithms are applied to the recommendation models. The approaches utilize secure multi-party communication protocols to enable user input to be kept confidential, while maintaining recommendations accuracy. However, these solutions are not suited for real-world deployment due to the high computational cost of encryption.

The second group [34, 35] evolves around obfuscation. Noise (i.e. perturbation) is added to input data, such as user ratings, before being utilized in recommendation generation in order to block inference of demographic information about the user. [34] propose a technique for perturbing users' movie ratings to mitigate inference of the users' gender, with insignificant impact on recommendation accuracy.

The third group [36–41] contains works focusing on the notion of Differential Privacy (DP), an approach which has become well-established in recent years. The essence of DP is that output of calculations should not allow inference about any of the original data, and, much like the works in the second group, it can be achieved by perturbing the input. However, these works fail to protect the user's sensitive information beyond the users' ratings under membership attacks. A recent work by Zhang et al. [5] proposes a differentially private convolutional network, but similar to the second group, it aims to protect demographic information of the user.

The last group [4, 14, 42] utilizes adversarial learning to defend against inference attacks. The work by Beigi et al. [4] proposes the Recommendation with Attribute Protection (RAP) model which aims to provide relevant recommendations while protecting against attribute inference attacks. The system is modelled as a min-max game between two components; a Bayesian personalized ranking recommender and a private-attribute inference attacker, with conflicting objectives. The attacker is utilized to optimize the recommendation process, such that demographic information not can be inferred accurately from the generated recommendations combined with the user's' item list.

2.5 Recommendation Models

There exists different models which can be used for recommendations. In this section, the models relevant for this thesis will be presented.

2.5.1 Factorization Machines

Factorization Machines(FM)[43] are generic supervised models which combine the advantages of Support Vector Machines(SVM)[44] with factorization methods . The models map real-value features into a low-dimensional latent factor space and are easily applied to different kinds of prediction tasks, such as classification, regression and ranking. Interactions between variables are represented using factorized parameters, enabling the model to make predictions about variable interactions even for highly sparse data. This is often the case for recommender systems, and is an advantage which separates Factorization Machines from SVM. The model equation can be calculated in linear time, making it scalable for large datasets and is defined according to Rendle [43]:

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

where x is the feature vector for user u, and $w_0 \in \mathbb{R}$, $w \in \mathbb{R}^n$, and $V \in \mathbb{R}^{n \times k}$ are the parameters to be estimated. The interaction between a user and an item is represented using binary vectors, where each vector has one non-zero value. The model allows the use of contextual information in the recommendation process, by adding relevant auxiliary user/item features to the training sample. Figure 2.3 displays an example of training data for Factorization Machines. The approach presented by Rendle aims to minimizing rating prediction error, and must therefore be adapted in order to use it for implicit feedback.

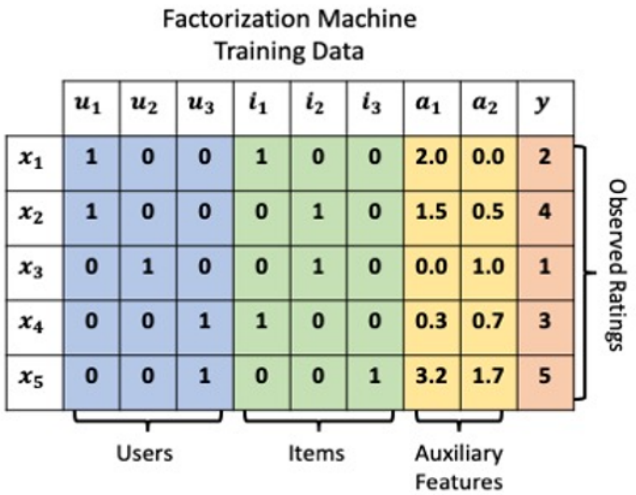


Figure 2.3: Training Data for Factorization Machines[45].

2.5.2 Learning to Rank

Personalized ranking refers to the task of providing a user with a ranked list of items. An example is a streaming service, which provides its users with a personalized list of movie recommendations. Some of these systems generate recommendations based on implicit feedback. In the case of a streaming service, an example can be the list of movies the user has seen in the past. There are several recommender models which allow the use of implicit feedback, such as Matrix Factorization[46] and adaptive-KNN[47], but are not optimized for personalized ranking.

Machine learning techniques which are used to train the model in a ranking task are called Learning to Rank(LTR). These are optimization methods which attempt to learn the items' rank-order directly instead of minimizing prediction errors[48]. First, a local ranking model $f(q,d)$ is trained on a pair of user-item interactions q and d, which will output a score regarding the items' relative ordering. A global ranking model $f(q,D)$ can be considered more generally, where D is a list of items, and the output will be a list of scores. Then, the aim is to find a suitable permutation(ranking list) out of all possible permutations, given the user and their associated items, by using the scores provided by the ranking model.

2.5.3 Bayesian Personalized Ranking

Bayesian Personalized Ranking (BPR) is a personalized ranking loss which can be derived from the maximum posterior estimator. The technique is widely used in collaborative filtering. Pairs of positive (observed) and negative (unobserved) items are included in the training data, where the observed items are assumed to be preferred over unobserved items. The training data is represented by tuples (u,i,j), where user u prefers item i over item j. Next, BPR aims to maximize the posterior probability, according to the Bayesian formulation in the work by Rendle et al.[49]:

$$p(\Theta|>_u) \propto p(>_u|\Theta)p(\Theta) \tag{2.2}$$

Where Θ represents the recommendation model's parameters and $>_u$ is the desired personalized total ranking of a user's (u) items. $p(\Theta|>_u)$ can be defined as the difference between item i and item j's predicted utility scores, mapped into a latent space $[0,1]$ using a sigmoid function. Then the generic optimization criterion can be defined as the following for a personalized ranking task[20]:

$$\text{BPR-OPT} := \sum_{(u,i,j\in D)} \ln \sigma(\widehat{y}_{ui} - \widehat{y}_{uj}) - \lambda_\Theta \parallel \Theta \parallel^2 \tag{2.3}$$

where $D := \{(u,i,j)|i \in I_u^+ \wedge j \in I \setminus I_u^+\}$ is the training set. I denotes all items, while I_u^+ refers to the items a user u liked and $I \setminus I_u^+$ the unobserved items. The predicted scores for user u to item i and item j, are $\widehat{y}_{ui}$ and $\widehat{y}_{uj}$ respectively. σ is a sigmoid function, shown in Figure 2.5, and λ_Θ are model specific regularization parameters.

2.6 Classification Models

There are a variety of different classification algorithms available, all which differ in complexity, and have their own set of advantages and disadvantages. This thesis makes use of the algorithms; Random Forest, Logistic Regression, Support Vector Classification (SVC), and Most frequent, which are presented in the following sections.

2.6.1 Random Forest

Random forest is a popularly used algorithm, which is both easy to use and flexible[50]. It can be used for both classification and regression tasks. The algorithm is an ensemble of random decision tree classifiers, which combines the prediction of decision trees to make stable and more accurate predictions. This technique has roots in ensemble learning, which is the process of generating and combining several classifiers to solve a complex problem and improve model performance [51]. The technique mitigates the problem of overfitting and generally provides results with lower variance.

A decision tree is a common tool used for decision making, representing decisions in the form of a tree-like model [52]. The term also refers to the method of constructing such a model automatically from data. This is done by classifying a population into segments similar to branches, resulting in a tree with root nodes, internal nodes, and leaf nodes. The root node and internal nodes represent a question or an attribute, and the emerging branches represent either possible attribute values or answers to the question. Leaf nodes contain the resulting class or category. Figure 2.4 shows an example of a decision tree. The model also includes the probability of different outcomes, as well as costs and utility of resources. Decision trees have a disadvantage, namely inaccuracy [53]. They work great with the training data, but are not that flexible when classifying new samples.

Random forests combine the simplicity of decision trees with flexibility by introducing randomness into the model, causing a great increase in accuracy. There are many methods which can be used to achieve this, such as the random subspaces method[54], random feature selection[55] and Bagging[56]. For this thesis, the Bagging method is used, where each tree is built from a bootstrap sample of the

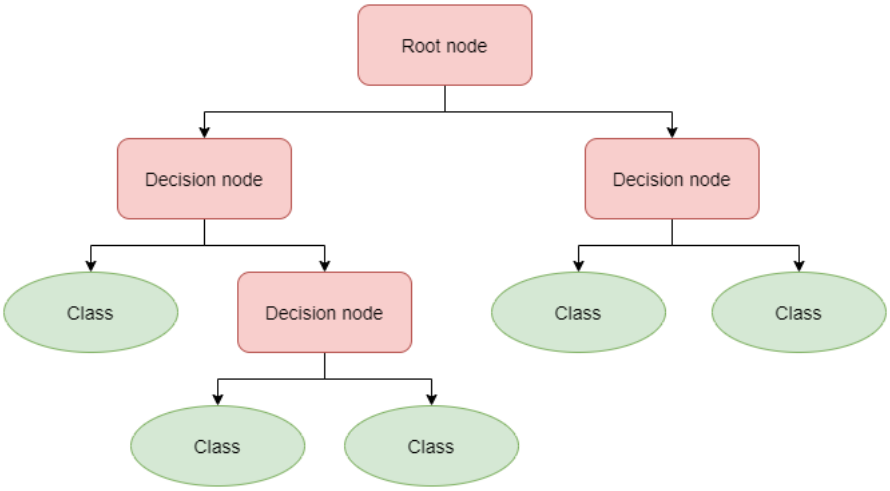


Figure 2.4: Illustration of a decision tree.

training set, and the aggregate is used to make a decision.

The first step of the algorithm is to create a bootstrapped dataset from the original dataset. The bootstrapped dataset must be of the same size as the original dataset, and can include the same values from the dataset several times. Then, a decision tree is built, where only a subset of randomly selected variables from the bootstrapped dataset is considered at each step. Once the decision tree construction is finished, the process is repeated, building a forest of decision trees. The introduced randomness increases variety and is what causes the random forests to be more effective than single decision trees.

When using random forest for classification, the input data x is fed to all the decision trees in the forest, and each tree will assign the sample a predicted class. Then, the class which received most votes is concluded to be the sample's predicted class. This strategy is called majority vote[57] and is the most commonly used strategy for deciding on a predicted class. Another method is to use the average of the classifier predictions, which is most commonly used for regression tasks, but can also be applied to classification tasks. In this thesis, the final prediction is made by calculating the average of the predictions of the ensemble classifiers.

There are typically a part of the samples which are not included in a bootstrap dataset, which is called an out-of-the-bag dataset. These out-of-the-bag samples can be used to evaluate the model, by running them through the decision trees which were created without them and see if they are classified correctly. The accuracy of the model will depend on the proportion of the out-of-bag samples which were classified correctly by the random forest. The proportion which was incorrectly classified is called the out-of-bag error. Once the accuracy of the model is calculated, the random forest construction process can be repeated where the number of variables randomly selected at each step is changed. This is done a

number of times and then the most accurate model is chosen.

2.6.2 Logistic Regression

Logistic regression is an efficient and simple algorithm for linear classification problems, which thrives when using both dense and sparse data [58, 59]. The algorithm is also unlikely to overfit and is scalable for larger datasets. An disadvantage is that it is highly dependent on regularization, and requires data preprocessing such as normalization and scaling of the dataset.

In order to estimate the probability for X belonging to a certain class, a hypothesis function is used. The probability represents the model's confidence in that $y = 1$, given an input X and parameters Θ. A property of logistic regression is that the hypothesis lies within the range of 0 and 1. This is achieved by using a Sigmoid function (logistic function). As shown in Figure 2.5, the Sigmoid function converges towards 1 when an input z approaches infinity and 0 when z approaches negative infinity.

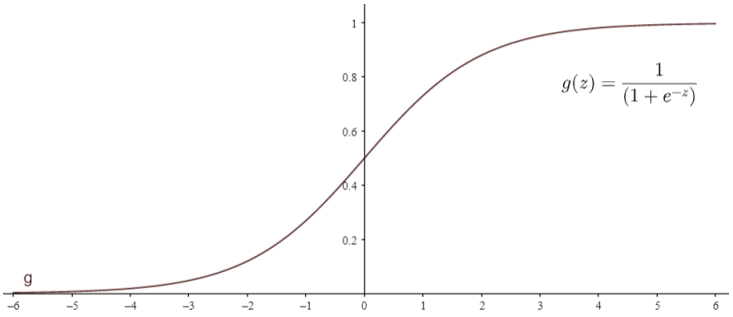


Figure 2.5: Example of a sigmoid function.

To make predictions from the probability estimate, a decision boundary is needed. The decision boundary is a function which draws a clear line between when y should be predicted as 1 and when y should be predicted to be 0. This can be a simple threshold, such as $h_\theta(x) \geq 0.5$, or be a more complex function, which can be both linear or nonlinear.

Equation 2.4 displays the algorithm's cost function. If the prediction and the truth value are the same, the cost is zero, and if the prediction is zero, the cost is infinity. An important aspect of the cost function is that it is convex. In order to find parameters Θ, the cost function is minimized. This can be done using an optimization algorithm, such as Gradient Descent. Gradient descent will only converge at the global minimum if the cost function is convex. It is possible to use other optimization algorithms, such as L-BFGS [60], to increase the speed of Logistic regression.

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if} \quad y = 1 \\ -\log(1 - h_\theta(x)) & \text{if} \quad y = 0 \end{cases} \tag{2.4}$$

To use logistic regression for multiclass problems, a strategy called one vs all is used. For each class i, a logistic classifier $h_\theta^{(i)}(x)$ is trained to predict the probability that $y = i$. To make a prediction on a new input x, choose the class i that maximizes $\max_i h_\theta^{(i)}(x)$.

2.6.3 Support Vector Machine

Support Vector Machine (SVM) is another common algorithm, which can be used for both classification and regression tasks [44, 61, 62]. SVMs are often preferred due to its ability to produce fairly high accuracy with less computational power. The goal of a support vector machine is to find a hyperplane which represents the separation of two classes, also called a margin. This is displayed in Figure 2.6. By seeking to enlarge the distance between the training samples from each class, the algorithm becomes more confident and robust in its decision making. In this process, support vectors have a role to play. Support vectors are data points which are close to the margin, and their position will therefore impact the hyperplane's position and orientation.

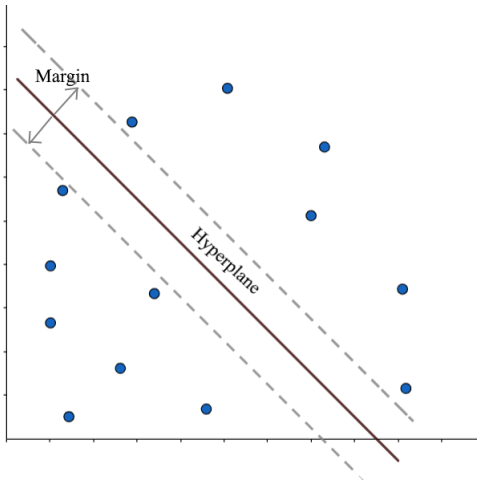


Figure 2.6: The large margin principle.

The cost function for support vector machines is quite similar to the cost function of logistic regression, but has a different property. For y=1, the cost will be 0 when the $h_\theta(x) > 1$, and for y=0, the cost is 0 when $h_\theta(x) < -1$. This gives rise to the margin between the classes.

It is important to note that the usefulness of the margin depends on a parameter C. C controls whether the cost function or the regularization terms should weigh

more, and has a similar role as λ has for logistic regression. A large margin classifier is only useful when C is large and the model's variation is high. A reasonably small C makes the decision boundary more robust by ignoring some outliers, and is caused by giving more weight to the regularization term. Unlike classifiers such as logistic regression, the hypothesis of SVM does not output probabilities, but the class directly.

The margin hyperplane is N-dimensional, where N corresponds to the number of features. A set of polynomial features is an alternative way to represent data samples, and is helpful when wanting to find nonlinear decision boundaries. Each feature uses a kernel function $k(x, y)$, which is a similarity function between an input x and a target value y [63]. Equation 2.5 displays an example of a Gaussian kernel function. If an input x is close to the target value y, the kernel function will output 1, and if the distance is great, the function will output 0. Once a set of features for a given input x has been calculated, the class for x can be predicted.

$$k(x, y) = exp\left(- \frac{||x - y||^2}{2\sigma^2} \right) \tag{2.5}$$

The kernel function has a parameter σ, which can give different effects when changing the value. By decreasing σ, the value of the feature drops rapidly as x moves away from y, resulting in a lower bias and higher variance. Naturally, the opposite effect is achieved by increasing σ.

In order to use SVM for mutliclass problem, the problem is reduced into multiple binary classification problems, using either the one-vs-all or the one-vs-one strategy.

2.6.4 ZeroR classifer

When evaluating a classifier model, the results need to be interpreted and made sense of. By using a baseline classifier to compare numbers with, it will become easier to understand the significance and meaning behind the results. One of the most common baseline classifiers is the ZeroR classifier[64]. The algorithm finds the class which occurred most frequently under the training process, and simply chooses this class for all new inputs x. In the case of a binary classification problem, the algorithm will prove to be right more often than not simply by following the odds. In order to deem the classifier up for evaluation useful, it needs to outperform the ZeroR classifier. If not, it will not be considered better than guessing. When dealing with classification problems with high class imbalance, a result just a bit higher than the ZeroR algorithm might be significant.

2.7 Data Splitting Strategies

It is usual to split the data used for recommendation and classification tasks into a training set and a test set. The training set is commonly used for the system to develop a model and feature sets, while the test set is used to give an unbiased assessment of the model's performance. It is therefore important for the test set to not be included in the model development process, as it will cause the results to be misleading. An important topic is how big the portion of dataset aside for the test set should be. A large training set is favorable to enhance the model's prediction performance, but if the test set becomes to small, the results might not give an accurate impression of the model's general performance. The number of samples should ideally be much bigger than the number of the model's predictors. In addition, it is common to use an extra set, the validation set, if hyperparameter tuning is performed for the model. In the next sections, popular data splitting strategies for recommender models and classification models are presented.

2.7.1 Splitting Strategies for Recommendation Models

In order to be able to evaluate a recommender system, it is usual to split the data into a training set and a test set. There are many strategies which can be applied when splitting the data, each which will have different effects. A brief overview of some common strategies according to Meng et al. [65] is provided below.

Leave One Last

Leave One Last is one of the most commonly used splitting strategies for item-based recommendation systems. For each user, the last interaction between the user and the system is used for validation and the rest for training the system. There are two variations of this strategy, based on the format of the interactions. The first one is called Leave One Last Item, where an interaction corresponds to a user-item pairing. The second is Leave One Last Basket, where an interaction represents a pairing between a user and several items. It is commonly applied for systems where users can buy a group of items. The advantage of leave one last is the abundance of training data. The downside is that the performance results may not accurately represent the system's ability to make suitable recommendations for the user over time, as only one interaction is used for valuation for each user. Another potential problem is 'leakage' which will be described in the next paragraph.

Temporal Splits

The main idea of the strategy is to split up the interactions by time, and have been used in works like [66] and [67]. There are two versions of this splitting. The first one is called Temporal User, which is quite similar to the leave out last item strategy, except that a chosen percentage of the last interactions are kept

for validation. As the training/test boundary might vary greatly between users, evidence regarding future interactions might leak into the model during training. This is information which should not be expected to be available to the model at the time of recommendation generation, might lead to an overestimation of the model's prediction ability. The second version is the Temporal Global split, which instead uses a fixed timestamp common for all users when partitioning the interactions into training and test sets. The temporal global split is considered to provide a more realistic setting than the temporal user strategy. As the fixed partition timestamp is not tailored to each user's interaction history, there is no guarantee that all users will have interactions in both training and test set. Those users who do not have interactions in both sets are removed from the dataset, causing a bit of data loss.

Random Split

The strategy randomly chooses a set of interactions for each user to be in the test set, and the rest to be in the training set. Rendle et al. used one version of this strategy in [49]. The strategy is simple to use, but is not inherently realistic or reproducible, unless the author specifies the splits.

User Split

The last strategy splits interaction data by users, and is one of the less common ones. By letting the model train on one set of users and then validate using another set of users, the model is assumed to have the ability to make recommendations for new users, which many models do not have. This approach does also suffer from the leakage problem.

2.7.2 Splitting strategies for Classification Models

Just like for the recommender system, there exists several ways to partition the data for a classification task. The most common strategies according to Kuhn et al. [68] are presented below.

Simple Random Sampling

Random sampling, with its many variations, is the most widely used strategy for data partition for classification models. Using simple randomized sampling means that each sample has the same probability to be chosen for a set, making it an unbiased representation of the sample population and its inherent characteristics. The splitting strategy is simple to use, but can cause problems if the class distribution between the sets becomes imbalanced, or if classes are completely excluded from a set.

Stratified Random Sampling

The stratified random sampling approach alleviates the problem by selecting samples randomly from each class instead of the entire population. This results in a more balanced distribution of samples between the different sets, and gives a more accurate representation of the population. The strategy can run into trouble in cases where there are samples not belonging to a specific class, or if a sample falls into several classes.

Non-Random Sampling

Non-random sampling is a strategy where samples are selected based on factors instead of randomness. In some cases a non-random approach might be the best choice. For example if time is an important aspect, it is beneficial to test the model on the newest data. However, by removing the random aspect it becomes much harder to know hold well the general population is represented.

2.8 Resampling Methods for Classification Models

Whereas sampling refers to approaches for selecting data which will be used to evaluate model performance, resampling refers to splitting methods for improving model accuracy and quantifying the uncertainty of the performance results [68]. In addition, the methodology provides support for model selection and parameter tuning. In the next paragraphs, the resampling approach Cross-Validation is explained.

2.8.1 K-fold Cross-Validation

Cross-Validation (CV) is a popular approach for model evaluation when there is limited data available. The approach consists of several iterations, where a new model is generated and a different part of the dataset is used as the test set for each round. This way the model is evaluated by using each part of the dataset for both training and testing, resulting in a generally less biased and less optimistic prediction of the model's skills than the training/test split.

The first step of K-fold Cross-Validation is to shuffle the data and partition it into K number of groups (*folds*), where each fold contains $\frac{1}{K}$ of the samples. Then, a fold will be chosen as the test set (*hold out set*), and the model will be trained on the $K-1$ remaining folds. This process will be repeated for K iterations, where each fold will be used as a test set exactly once, and a new model must be trained and evaluated for each iteration. At last, the model's final evaluation score can be derived from the iteration scores, for example by averaging them. Figure 2.7 shows a diagram of the data partition in 3-fold Cross-validation.

It is important to choose a suitable value for K [69]. Some common values are $K = 5$ and $K = 10$, but there is no explicit rule. Generally, a larger K means a

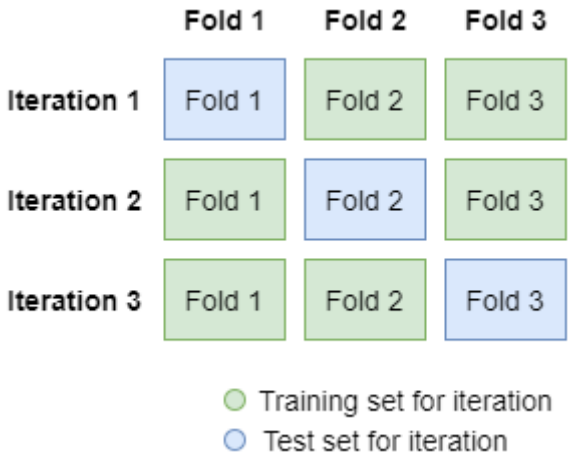


Figure 2.7: 3-fold Cross-Validation

smaller size difference between training and test set, and therefore results in less bias. Choosing a high value for K will result in a high number of iterations and a high computational cost. It is therefore not a suitable choice in some cases when large datasets are involved. In addition, a bigger test set might lead to less training data and a less confident model, increasing the risk of higher variance. However, it is also important to choose a K big enough for the training set to be representative for the whole dataset.

Hyperparameter tuning can be performed for each iteration and validated using the test set. Then, the best performing model can be chosen at the end. However, by tuning the model parameters to fit the same training set used for model evaluation, there is a risk of overfitting and overestimating model performance.

2.8.2 Nested k-fold Cross-validation

In order to reduce the chance of overfitting, nested cross-validation can be used [70]. As the name states, a new round of cross-validation is performed within each iteration of the cross-validation. This means that for each iteration, the training data is split into a new training and test set, which are used for hyperparameter tuning. Once each model has been evaluated, the best model is chosen to participate in the outer cross-validation for model evaluation. Then, the final score is found by averaging the model performance scores. By only using a subset of the dataset for parameter tuning, the model is less likely to overfit under model evaluation. As nested cross-validation can be computally expensive it is common to choose a smaller K for the inner CV, such as $K = 10$, $K = 5$ or $K = 3$.

2.8.3 Other Variations of Cross-Validation

There exists many variations of Cross-validation. Leave One Out Cross-Validation is a version of the K-fold strategy, where $K = n$, where n is the number of samples in the dataset. In other words, there will be n iterations, where the test set will always consist of one sample. The method has the same strengths and weaknesses as the Leave one last splitting strategy for recommender models. Another approach is the Stratified k-fold cross-validation, where samples are selected randomly from each class to be included in each fold, creating a more balanced class distribution between each training and test set. The Monte Carlo Cross-Validation approach (*Random Subsampling*), is quite similar to k-fold cross-validation, except that it allows overlap between the splits. In other words, for each iteration, a new test set is chosen completely randomly from the dataset, allowing samples to be included in the test set more than once.

2.9 Evaluation Metrics

2.9.1 Evaluation of Recommendations

In order to make accurate inferences, the recommender system's ability to generate relevant recommendations for the user is crucial. For evaluating the recommendations, the standard metrics P@K and R@K [71] are used, as well as hit rate [1]. These metrics are widely used in related works, including the the work by Beigi et al.[4]. Precision and recall are both well-known metrics used in information retrieval, and measure how relevant a list of ranked recommendations is for the user.

P@K, or "precision at K", is the ratio of test cases which has been successfully recommended in a position of the top-K ranking list to the value of K. For each user, P@K is calculated as:

$$P@K = \frac{|\{\text{test items}\} \cap \{\text{top-K returned items}\}|}{K} \tag{2.6}$$

R@K, or "recall at K", represents the ratio of recommended items in the top-K ranking list, which also exist in the test set, to the number of recommended items in the test set. For each user, R@K is measured as:

$$R@K = \frac{|\{\text{test items}\} \cap \{\text{top-K returned items}\}|}{|\{\text{test items}\}|} \tag{2.7}$$

After P@K and R@K are measured for all users in the dataset, the average P@K and R@K is found and is the finalized measurement.

Hit rate measures how well the model is able to recommend removed items, i.e. items which exist in the test set. A hit has occured when an item in the user's top-k

recommendations also exists in the user's test set. A low hit rate means that the model does not have sufficient information about the user's preferences. In this thesis, hit rate is measured as an average of all users' hits, defined as

$$\text{HR} = \frac{|\{\text{test items}\} \cap \{\text{top-K returned items}\}|}{|\{\text{users}\}|} \tag{2.8}$$

2.9.2 Evaluation of Classifier

ROC curve
A Receiver Operating Characteristic (ROC) curve is a graphical plot which displays the performance of a classification model at all classification thresholds. The curve plots the two parameters; False Positive Rate (FPR) on the x-axis and True Positive Rate (TPR) on the y-axis. By decreasing the classification threshold, the classifier will classify more items as positive, and thereby cause the number of True Positives (TP) and False Positives (FP) to increase accordingly. An example of a ROC curve is provided in Figure 2.8.

False Positive Rate (FPR) is defined as:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{2.9}$$

where "FP" means "False Positive" and "TN" refers to "True Negative".

True Positive rate (TPR) has the following definition:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.10}$$

where "TP" refers to "True Positive" and "FN" means "False Negative". TPR is a synonym for the recall metric.

AUC
While ROC displays the classifier's probability of accurate classification, Area Under the ROC (AUC) curve measures the classifier's ability to distinguish between classes. More specifically, AUC is the probability of a classifier ranking a random positive instance higher than a random negative instance. the two-dimensional area under the ROC curve. An example of AUC is provided in Figure 2.9.

AUC is given by measuring the entire area under the ROC curve from (0,0) to

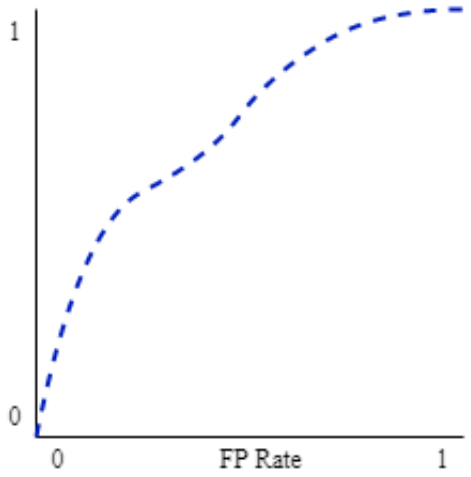


Figure 2.8: Example of a ROC curve.

(1,0), and is defined as:

$$\begin{aligned} &TPR(T) : T \rightarrow y(x) \\ &FPR(T) : T \rightarrow x \\ &A = \int_{x=0}^{1} TPR(FPR^{(}-1)(x))\,dx \end{aligned} \tag{2.11}$$

.

Where T is the threshold, TPR is the True Positive Rate, FPR is the False Positive Rate, X1 is the score for a positive instance, and X0 is the score for a negative instance.

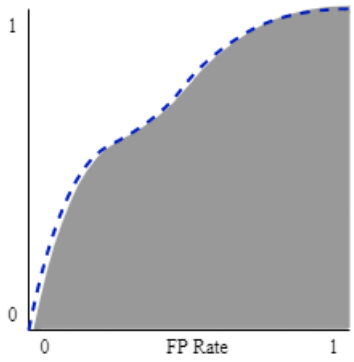


Figure 2.9: An example of AUC.

Micro-AUC

In order to evaluate the adversary classifier in regards to inference of the location attribute, the well-known metric Micro-AUC [72] is used. Micro-AUC is often

chosen when the data distribution is suspected to be imbalanced, such as the case is for the dataset used in this thesis. Unlike the Macro-AUC metric, Micro-AUC takes the contributions of each class into account, and therefore gives a more accurate assessment. Higher AUC implies more accurate location inference, and thereby also a greater threat to the individual's privacy in regards to private attributes.

Chapter 3

Related work

There has been quite a bit of research on the topic of user privacy related to the use of Web services and social networks, including recommender systems. The focus of this thesis is private-attribute inference attacks, with an emphasis on the inference of location. This chapter should not be considered a complete guide to the field, but rather an introduction to the field of study which this thesis builds upon.

3.1 Location Inference Attacks

Research[73, 74] shows that relationships in social media are strong indicators of location. Based on this principle, several works[6, 12, 13, 75–79] uses information about the user's friends and social circle to infer location. Backstrom et al.[6] published a work which utilizes the users' home address and Facebook's social network for inferring the user's geolocation. According to research, the users' geolocations at given points of time can be inferred, or their pseudonymous location traces can be re-identified, if an adversary has access to a collection of the users' disclosed locations[80–82]. Based on this principle, Backstrom utilized the geographical distribution of the user's friends to maximize the likelihood of where the target user's location was. Since then, there have been many developments on the topic. For example, Kong et al.[7] extends the work of Backstrom et al.[6] by including strategies for weighting and estimating which of the user's friends are likely to be the most predictive of the user's location. Some works, such as one by Gong et al.[16, 83], utilize both information about the user's social circle and additional public information to infer location, and were able to correctly infer the city the users lived in for 57 % of the users. If a user has many people from an area in their social network, or publishes many posts or other information related to an area, it is likely that the user has a strong connection to the city as well. These works created the foundation for AttriInfer[17], a more efficient approach which uses Markov Random Fields to infer the location of the users. Other works, such as the

works[7, 84, 85] utilize mentions (e.g. check-ins). Jurgens et al.[84] proposed an approach which used spatial propagation of location assignments through a social network. The method only needed a small amount of initial locations and were able to accurately infer the location of nearly all users. For the most active users, the median location error was estimated to be 10 km.

Other methods are text-based. These works often extract location related words, topics and features to predict the user's location. People in the same area often talk about popular events and topics currently happening around them. Based on this idea, in the work of Eisenstein et al.[86] a model is created which is capable of identifying geographical-dependent terms and linguistic communities. Then, the model is applied to infer users' location just from Twitter text. Han et al.[87] propose a similar approach, where the relationship between the distribution of location, topics, and user characteristics, is modelled in order to predict the user's location. Another work, proposed by Han et al.[88] infers the user's location by extracting location-indicative words and looking at the geographical bias of the used words. Ryoo et al.[11] uses the same principle, but focuses on inferring the user's main location of activities. They were able to infer the location of 60 % of Korean Twitter users within 10 km of their location. The works of Wing et al.[89] and Tang et al.[90] also train their classification models based on extracted features(mentions, entity names, etc.).

Some approaches[91–96] combine text-based methods with relationship-based methods to infer the user's location, and thereby balance out the weaknesses from both types of approaches. Rahimi et al.[94] based their inference on utilizing a user relationship graph. However, there are many isolated users in the graph which cannot have their location predicted accurately from other users. Therefore, a text-based approach was also utilized to infer the location of these users. A later work of Rahimi[93] builds upon this idea and proposes a multiview geolocation model, rooted in Graph Convolution Network, which uses both network and text context. Tian et al.[96] proposes a location inference method which gives a special consideration to the relationship between users and indicative words. The method utilizes representation learning and label propagation to filter out relationships which are not relevant to the location attributes and accurately represent characteristics of the users' location attributes.

It appears to be some gaps in the state-of-the-art. Many works utilize social circle information, text-based approaches or a hybrid, but few works are leveraging other kinds of behavioural information. In fact, there seems to be no works which attempt to infer location solemnly from a user's generated recommendations. As recommendations can be more accessible than other types of behavioral information, it is an important topic to investigate. The majority of works are also rooted in either social networks (e.g. Twitter and Facebook) or location-based recommender systems (e.g. Foursquare and Yelp), both promoting user location sharing as a part of the experience. Other domains where the user's location appears to be less related to generated recommendations, such as movies and books, seem to

be less explored in the context of location inference. This is a topic which can be interesting to investigate, as location information is not assumed to leak through the recommendations in these domains. If the users' location information can be inferred without the users' knowledge or consent, it will be considered a breach of privacy.

3.2 Attribute Inference Attacks

The aim of an adversary executing a private-attribute inference attack, is to infer private attributes of users given their interaction history and public available information. Such attacks could be categorized into three groups.

The first group leverages behavioral information about users, such as public information, interaction history, and recommendations. Publishing trivial information about oneself in social networks might seem harmless, however Kosinki et al.[97] shows that users' likes on Facebook can be used to infer a range of sensitive attributes, such as gender, sexual orientation, ethnicity and political view, by using logistic regression. Similarly, Chaabane et al.[15] attempts to infer private attributes such as gender, age, and country, from facebook interests, using public information from users with similar interests, extracting their semantics from Wikipedia and applying statistical learning for comparison. Other works, such as BlurMe[34] and BlurM(or)e[35] uses classification models, such as logistic regression and SVM to accurately infer the users' gender from users' movie ratings. Unlike Facebook likes, the users' ratings and interaction history are usually not considered public, and can cause a different type of threat, e.g. through unauthorized access, insider threat, or published datasets. Similarly, the work of Resheff et al.[98] shows that gender and age can be inferred from the system's user-representations when combined with external information.

Some works infer private information from the recommender system output. Calandrino et al.[28] demonstrates that the user's past transactions can be inferred based on the user's recommendations and a subset of their transaction history. This is done by using an algorithm which monitors the changes in the recommender system public output over a period of time. Then, the recommendations are combined with the subset of the user's transaction history which is available to the attacker in order to infer the unknown interaction history. Another work, authored by Beigi et al.[4], infers demographic information, such as gender, age and occupation, from the users' recommendations and previously rated items. First, a Bayesian personalized recommender generates recommendations, by extracting users and latent embeddings from interaction data and utilizing a learning to rank method. Then, the users' recommendations and interaction history is fed to a Recurrent Neural Network(RNN) for the attributes to be inferred. Similarly, Zhang et al.[5] infers demographic information from recommendations and interaction history using a two-layer deep neural network model. Although, an important notion is that the main focus of these two approaches is protection against attribute infer-

ence. There are seemingly no works which infer private attributes only from a user's location.

The second group utilizes information about the user's social circle and community memberships. These inference attacks are based on the principle of social influence, which states that connected users are likely to have similar attributes[99]. Several works[77–79, 100–104] uses the user's friends and social relations to infer private user attributes. He et al.[100] shows that this is possible to achieve with high accuracy, especially when users are connected through strong relationships. In addition, the research shows that it is still possible to infer private attributes in a community where most people hide their information. Dey et al.[102] designed an iterative algorithm which utilizes the social network structure to predict Facebook users' age, based on the age of their friends and friend-of-friends. The algorithm is even able to infer the age within a few years' margin of users who did not publish their friend lists. The work of Gong et al.[105] infers private information by using social relationships, and takes it a step further by predicting hidden relations between users based on the inferred information. In addition, some works[106, 107] uses information about users' community memberships. Users who are grouped together often have some similar attributes. Even if a user hides their attributes, they might be predicted based on the other users' attributes in that group.

The last group combines the two types of information; behavioural and social, to launch powerful attribute inference attacks. The works of Gong et al.[16, 83] propose a Social-Behaviour-Attribute (SBA) network model which ties social structures, user behaviours, and user attributes together in a unified framework. Thereafter, attribute inference is performed through a vote distribution attack on the users in the SBA network. Another approach is presented by Jia et al.[17], which also leverages users who are missing certain attributes (negative training users), in the inference process.

This work intends to address a gap in the state-of-the-art. As mentioned, the inference approach typically utilizes either behavioural information, information about the user's social circle, or a hybrid. This work falls under the first category and investigates the topic of attribute inference from recommendations. The thesis shares similarities with the works by Beigi et al.[4] and Zhang et al.[5], as they also infer private-attributes from recommendations, but unlike these works, the thesis does not utilize the interaction history of the users. In addition, neither of these works focus on inference of private information from context-aware recommendation. Therefore, this thesis investigates the use of context-aware recommendations in the context of private-attribute inference. The user's demographic information (gender, age, occupation, and location) is utilized under recommendation generation. This information is also used as auxiliary features in the inference process. Another difference is that the works of Beigi et al.[4] and Zhang et al.[5] attempts to infer the gender, age, and occupation of the users, while the main focus in this thesis is inference of the user's location. This work is therefore unique, and is the

first work to attempt to infer the users' location from context-aware recommendations without any auxiliary behavioral information.

Chapter 4

Data

For the experiments in this thesis, two datasets are used: The MovieLens 100K (ML) dataset and the BookCrossing (BX) dataset. Both datasets are well established and widely used in related works. The next sections give an presentation of each dataset, their characteristics, attribute descriptions, as well as potential data splitting strategies for the recommender and classification models.

4.1 MovieLens 100K Dataset

GroupLens[1] is a project driven by the University of Minnesota, which publishes research, projects, and datasets related to the topic of social computing. MovieLens[2] is a service managed by GroupLens, which provides the users with non-commercial, personalized movie recommendations, and has enabled the publication of several datasets. For this thesis, the MovieLens 100K Dataset[108] was chosen. The dataset contains 100,000 ratings for 1682 movies given by 943 different users. The ratings are explicit, ranging from 1 to 5, and each user has rated at least 20 movies. In addition, some demographic information is included, namely the user's gender, age, occupation, and location.

The data was collected from September 19th, 1997 to April 22nd, 1998, a period of seven months, making it one of the older datasets. The dataset has been popular and can be found in several related research papers, including the wrok of Beigi et al. [4] and Strucks et al. [35]. Therefore, one advantage of using MovieLens 100k is the possibility of comparing experiment results to other works. The dataset is quite small, which proves to be both a downside and an upside. On one hand, the smallness and simplicity of the dataset makes it easy and suitable to test concepts. On the other hand, it can lead to sparsity and imbalance problems, as well as not being completely comparable to the many enormous real-world datasets that are

[1] https://grouplens.org/
[2] https://movielens.org/

Table 4.1: ML Attribute Description for Recommender Model

Attribute Name	# Classes	Classes
Gender	2	Male, Female
Age	3	Under 35, 35-45, over 46
Occupation	21	Administrator, Artist, Doctor, Educator, Engineer, Entertainment, Executive, Healthcare, Homemaker, Lawyer, Librarian, Marketing, None, Other, Programmer, Retired, Salesman, Scientist, Student, Technician, Writer
Location	52	US states

in use today. Another notion is that the dataset only includes users from the United States. Therefore, an aspect which might have an impact on the results is how much the users' movie preferences will vary across the country. If there is little variation in user preferences, it will be harder for the inference model to predict the location of the users.

The domain of MovieLens is movie recommendation. What makes this domain interesting is the general assumption that movie recommendations do not leak location information. If this assumption proves to be wrong, it would be a problem noteworthy to discuss.

4.1.1 Attribute description

As mentioned, the four attributes included in each user's demographic information, are gender, age, occupation, and location. A brief description of each attribute will be presented, as well as the distribution between different values and any applied preprocessing. In this thesis, the experiments will consist of a recommendation task and a classification task. In order to avoid classes with sparse data under classification, new splits will be defined for some attributes. The contextual information will be split into a different set of classes for the recommender model, displayed in Table 4.1, and the classification models, displayed in Table 4.2. There are several splits which can be used for the location information in the classification task, and these are therefore displayed separately.

Gender Attribute

The gender attribute has two classes, male or female, and is represented by a single letter. The distribution between the two classes are presented in Figure 4.1, and it is clear that there is a majority of male users, causing the dataset to be imbalanced. This might have an unfavorable effect when used for classification.

Table 4.2: ML Attribute Description for Classification Models

Attribute Name	# Classes	Classes
Gender	2	Male, Female
Age	3	Under 35, 35-45, over 46
Occupation	16	Administrator, Artist, Doctor/Healthcare, Educator, Engineer/Technician, Entertainment, Executive, Homemaker/Retired, Lawyer, Librarian, Other/None, Programmer, Salesman/Marketing, Scientist, Student, Writer

Age Attribute

The age attribute represents the user's age in years, but is split into three classes for this experiments. This is done as it would be favorable to reduce the number of classes under classification. The new classes are; below 35, 35 to 45, and above 45, split according to Beigi et al. [4] and Hovy et al. [109]. The new class distribution is displayed in Figure 4.2.

Occupation Attribute

The dataset includes 21 different occupations, some which are rather sparse. The imbalance may cause problems when the dataset is used for classification. In order to alleviate this, classes with some similarity are grouped together to form new classes for the classification task, displayed in Table 4.2. As shown in Figure 4.3, the distribution between different classes has mellowed out a bit.

Location Attribute

The users' location are represented by a US zip code. Some of the values are not valid zip codes, and would usually be beneficial to remove. However, this will have an impact on the inference of other user attributes, and will make the results less comparable to other related works. Therefore, the noisy zip codes are kept and

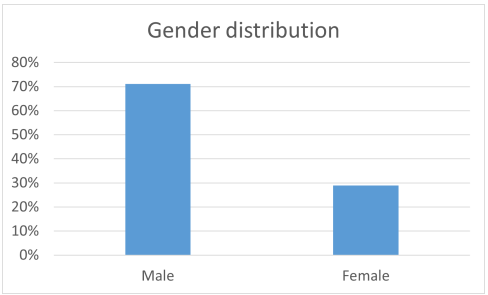


Figure 4.1: MovieLens Gender Distribution

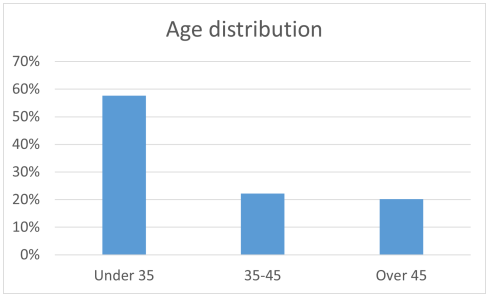


Figure 4.2: MovieLens Age Distribution

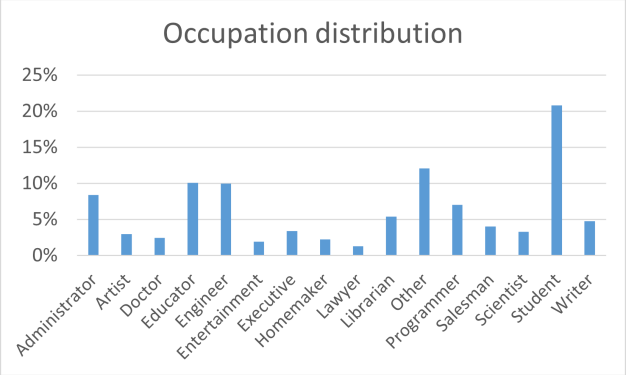


Figure 4.3: MovieLens Occupation Distribution

grouped together in a separate *none* class. As there are nearly as many users as zipcodes, there is a need for a new partition of the location data.

4.1.2 Location Data Partitions

In order to hinder classes with sparse data under classification, it is beneficial to reduce the number of classes. There are many strategies which can be applied when partition the location data. The partitions explored in this thesis are presented in the following paragraphs.

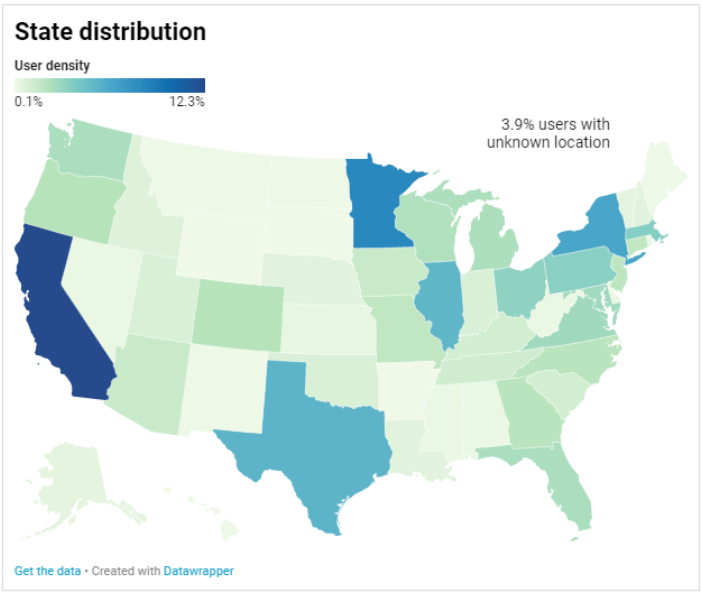


Figure 4.4: User Density in States

State, County and City Partition

The first explored partition which was based on the idea of mapping the zip codes to the belonging state, county and city, and analyze how the granularity impacted the classification results. However, this partition quickly proved to be unsuitable. The mapping of zip codes to cities resulted in 509 new classes, and as there are only 943 users, there would not be enough samples of each class to include it in both a training set and a test set for a classifier. Even though the state mapping resulted in fewer classes, namely 52, the problem remained. Figure 4.4 shows the distribution of the state classes, and it is revealed that there is a high imbalance in the dataset with a few dominating locations. As there are some states with only one user sample, it became impossible to include it in both a training and test set, once again. It was clear that a new partition strategy was needed.

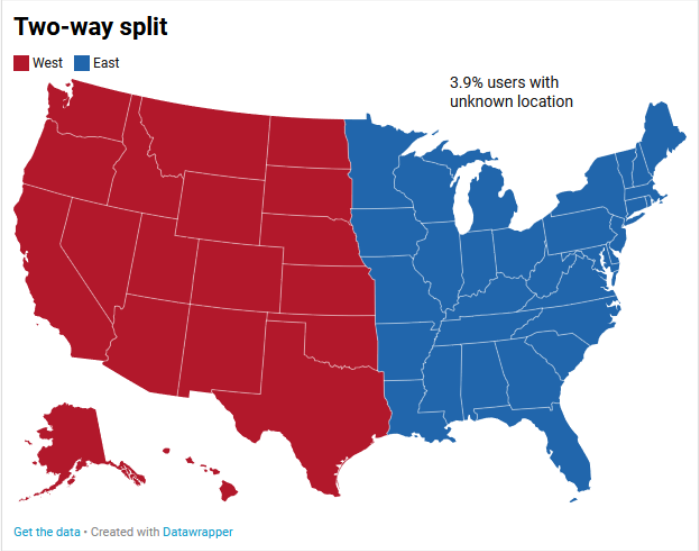


Figure 4.5: Two-way Split

2-way and 5-way Partition

The focus in the next strategy was to reduce the number of classes. This was done by grouping the states into regions. The partition strategy is simply based on the location of the states without any regard to the similarities or differences between the states. The 5-way partition allows for a bit more granularity and splits the country into five regions, namely West, Midwest, Southwest, Southeast, and Northeast. The split is shown in Figure 4.6. The 2-way split is displayed in Figure 4.5 and parts the country into a West and an East region. The distribution for both splits are presented in Figure 4.7 and Figure 4.8, and shows that the splits contain some imbalance. In addition to the regions, there is a dedicated *none* class for noise values.

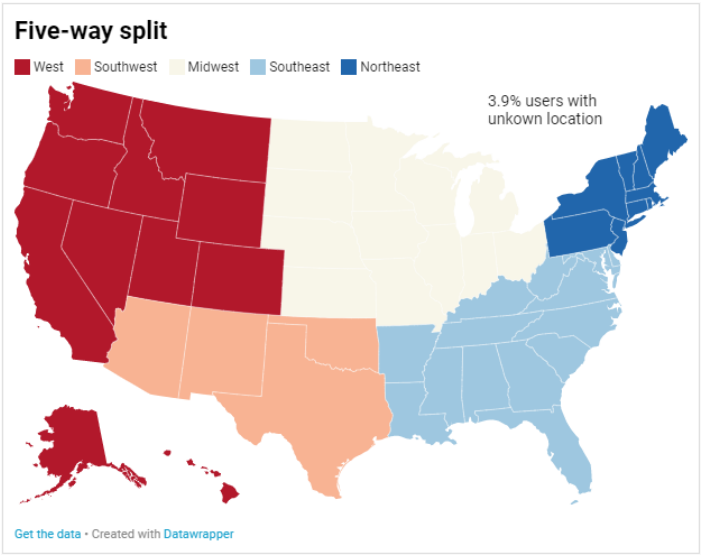


Figure 4.6: Five-way Split

Ten-way Partition

The ten-way partition was designed after the execution of the experiments using the 2-way and 5-way splits, and is a bit exploratory in nature. The experiments indicated that an increase in the number of classes might prove to have a positive effect. Therefore the number of classes in this partition is 10, including a *none*-class. The idea is that grouping states based on location or simply balancing the dataset may not prove to be the most meaningful way to partition the data, as the users in those states might not behave similarly at all. Instead, this partition is based on the users' similarity in movie preferences. This is done by first grouping users based on the similarity of their received recommendations, using a K-means clustering algorithm[110]. Then, the states are grouped based on each state's distribution of these user similarity classes, once again by using K-mean clustering. By doing this, the states grouped together are more likely to have something in

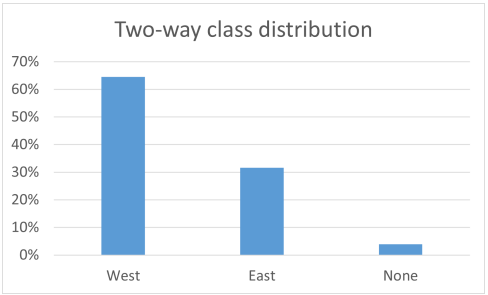


Figure 4.7: Class Distribution for ML Two-way split

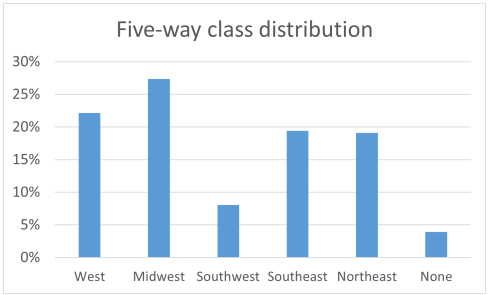


Figure 4.8: Class Distribution for ML Five-way split

common and might improve inference results. The auxiliary attributes were also utilized in this process. An important note is that these classes were generated by only using users and recommendations from the training set. In other words, just information available to a potential attacker. However, this strategy has some drawbacks.

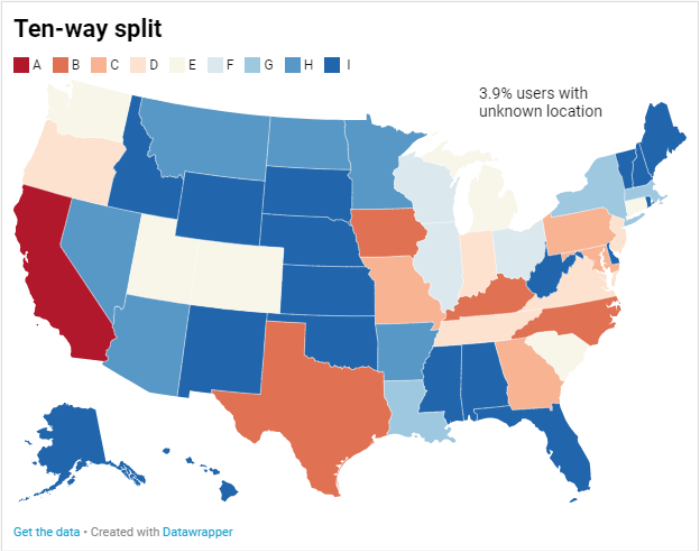


Figure 4.9: Ten-way Split

First of all, the information used to create the partition is specific for this dataset and will depend highly on the preferences of the users included in the training set, and might not be representative for new users. The other potential drawback is due to the imbalance of the dataset. The user distribution for states in Figure 4.4 shows that some states have just one or a few users, while other states have an abundance of users. This might cause states with many users to have a more similar distribution between the user similarity classes, simply because there is the possibility for there to be some users in each user class. As for the states with one or a few users, the distribution may end up highly skewed. Therefore, some alterations is made manually to create more balanced classes, as shown in the class distribution in Figure 4.10. The motivation behind choosing this partition is not to find a reasonable strategy for a potential attacker, but rather to explore the possibility of location information leaking through recommendations, by using a partition which conveniently makes sense to the inference model.

Other Splits

There are many partitions which would be interesting to explore, but due to the time limit and the limitations of the dataset, this was not explored in this thesis. For example, one strategy which was considered was to split the country into the top 10 cities with the highest population (according to Wikipedia), and compare

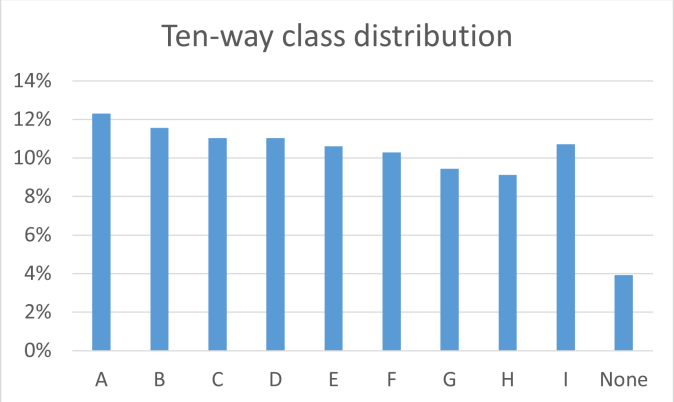


Figure 4.10: Class Distribution for ML Two-way split

them to the rest of the country. Unfortunately, the users from these cities combined did only make out 6.5 % of all users, and the partition strategy had to be dropped. Another potential partition strategy was to split the country into coast states versus country states, as these states might be culturally similar. The division of states which are traditionally republic or democratic speaks in favor of this. However, once again the distribution was highly imbalanced with 81 % of users belonging in a coast state.

4.2 BookCrossing Dataset

The second dataset chosen for this project is the BookCrossing dataset[71]. BookCrossing[3] is an online community which connects people through their love for books. The webpage has become an international library where users can share books that they love with other interested users. This way, the books can travel between users all across the world. It is also possible to meet other users with similar preferences, keep track of the books' journey, and leave ratings and reviews. The BookCrossing dataset was collected by Cai-Nicolas Ziegler, a Professor in Computer Science at the University of Freiburg, through a 4-week crawl that lasted from August to September 2004. The dataset consists of 1,149,780 ratings provided by 271,858 users regarding 271,379 books, making it a considerably bigger dataset than MovieLens 100K. For each user, age and location data is provided if available, and for each book a set of additional information is included, such as

[3]https://www.bookcrossing.com/

Table 4.3: Characteristics for ML and BX subset.

Dataset	# Users	# Items	# Ratings	Sparsity
MovieLens	943	1682	100,000	93.7 %
BookCrossing subset	3837	113,308	465,431	99.9 %

Table 4.4: Description of attributes from BookCrossing dataset

Split for	Attribute Name	# Classes	Values
Recommender model	Location	46	Countries
Classification model	Location	2	USA, remaining countries

book title, year of publication etc. The ratings are either explicit, ranging from 1 to 10, or implicit. The domain of the dataset, namely book reviews, is quite similar to MovieLens' domain, especially in regard to the lack of expectation for location information to leak through recommendations. The dataset was chosen as it has a similar domain to the MovieLens dataset, but contains more samples and an international userbase, which would be interesting to compare to the American userbase.

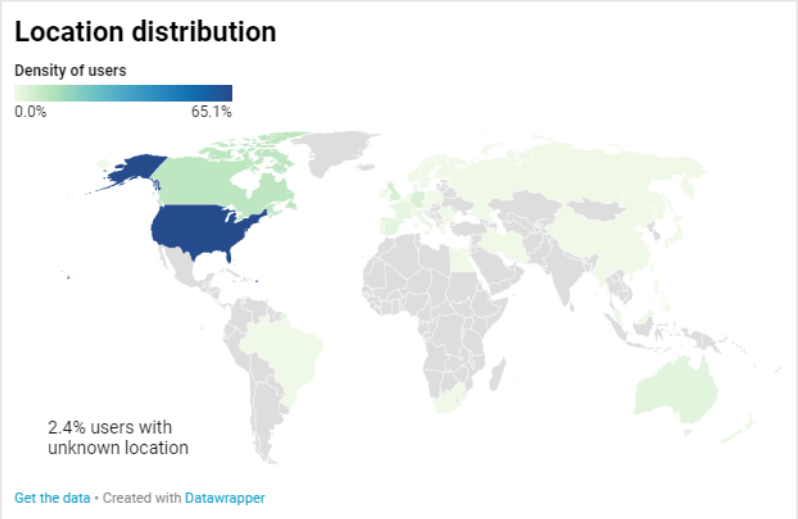


Figure 4.11: User Density in Countries

4.2.1 Subsetting the Dataset

In order to reduce time and computational costs, as well as avoid any memory problems, only a subset of the dataset is used in the experiments. The subsetting process consists of a set of filters, which each reduces the number of entries. The first step is to remove any entries with invalid values, such as books and ratings with an invalid isbn code, or users with missing location information. Then, explicit feedback is filtered out, as the recommender system model used in this thesis utilizes implicit feedback. As there are nearly as many books as users, there will be a large number of books with only one or a few ratings. The next step is therefore to remove the 50 % least popular books. By doing this, the the dimensionality of the dataset is considerably reduced. Due to the MovieLens dataset guaranteeing each user to have at least 20 ratings, the BookCrossing dataset must have the same

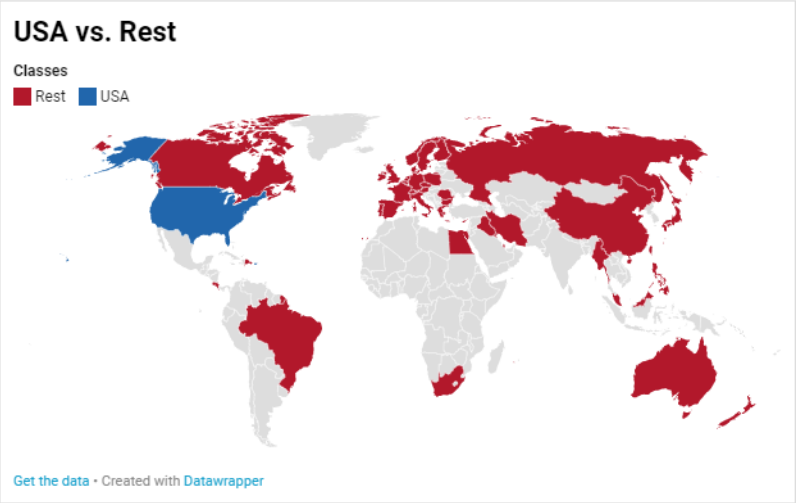


Figure 4.12: USA vs. Rest Split

constraint for the sake of comparability. Any user who has less than 20 ratings is filtered out. After completing the subsetting process, the result is a reduced dataset which consists of 184,640 ratings. The characteristics of the datasets which are used in this thesis are summarized in Table 4.3.

4.2.2 Attribute Description

The dataset provides two demographic attributes for the users, namely age and location. Unfortunately, a high number of users are missing a value for the age attribute, and therefore the attribute is not included in the experiments. The location attribute contains the city, state, and country of a user. Due to many users missing values for city or state, and for the sake of simplicity, only the country value is used. Thereby, the only contextual information included in recommendation generation is the user's location, as displayed in Table 4.4.

The class distribution between the countries in Figure 4.11 reveals great imbalance and indicate that new classes must be generated for the classification task. As the USA-class consists of 65 % of the samples, the location information will be split into two classes; the first one contains samples from the USA, and the other contains samples from the rest of the world. The classes are visualized in Figure 4.12. It would be interesting to look into different splits for this dataset in further work.

Chapter 5

Method and Experiments

This chapter begins by providing an overview over the different components and techniques which comprise the experiments. First, the utilized tools are described, then the threat model of these experiments, and next the recommender and classification models are presented. The different data splitting strategies are also included, as well as the hypotheses for the experiments. At last, a description of the different experiments is provided.

5.1 Tools and Libraries

NumPy[1]**:** NumPy is a library which adds support for large, multi-dimensional arrays and matrices, including a large collection of mathematical functions to operate on them.

Pandas[2]**:** Pandas is another package which offers tools for data manipulation and analysis. While NumPy focuses on arrays and matrices, Pandas provides support for numerical tables and time series, offering data structures and a set of manipulation operations.

Matplotlib[3]**:** The library extends Python and NumPy by adding plotting and visualization options. It provides the user with an object-oriented API for integrating plots into applications.

RankFM[4]**:** RankFM is an implementation of the Factorization Machine model class described by Rendle [43], which has been adapted for collaborative filtering recommendation with implicit feedback. It has options for using both Bayesian Personalized Ranking and Weighted Approximate-Rank Pairwise loss to learn model weights via Stochastic Gradient Descent (SGD). The

[1] https://numpy.org/
[2] https://pandas.pydata.org/
[3] https://matplotlib.org/
[4] https://github.com/etlundquist/rankfm

model accepts NumPy arrays and Panda DataFrames as input formats for interaction data, as well as additional user/item features to enhance the recommendation generation process. RankFM provides common recommendation methods, as well as a set of utility functions and metrics, like precision and recall.

Scikit-learn[5]**:** Scikit-learn is a machine learning library which presents a set of tools for predictive data analysis. It provides widely used algorithms for classification, regression, and clustering, including Random Forest, Logistic Regression, Support Vector Machines and ZeroR. The package also has support for preprocessing, such as standardization and normalization. Another feature is model selection, which enables the user to easily compare, validate and choose parameters and models, including the algorithms grid search and cross-validation, and different metrics, such as F1 score, accuracy, and ROC-AUC.

Uszipcode[6]**:** The package is a zip code database, which makes it possible to do zip code lookups for the United States. The package includes a set of different features and allows the user to customize their search. Additional information about the zip code is provided, making it a handy tool whether the user just wants to check the existence of a zip code, or wants to find out which city or state the zip code belongs to.

5.2 Threat Model

Using the terms described in Section 2.1, the following threat model is presented for a real-world scenario. Note that some assumptions are made for the experimental setup, which deviate from this threat model.

Resources: An attacker encounters a recommender system with a substantial number of users. At hand is a pre-trained classification model, which can be used to infer users' private attributes.

Access: The attacker has access to the recommendations of a subset of system users, gained by listening to network traffic, as well as the users' location information published on their social media account.

Objectives: The objective of the attacker is to learn the location of a targeted recommender system user, only given their recommendations.

Vulnerability: The attacker can breach the user's privacy by inferring the location of the user without their knowledge.

Countermeasures: There are no countermeasures assumed for this attack.

[5]https://scikit-learn.org/stable/

[6]https://pypi.org/project/uszipcode/

5.2.1 The Attack

For this attack, the goal of the adversary is to infer the location information of a recommender system user without the user's knowledge. The attack consists of two phases. First, a training phase, carried out from time t_0 to t_1, as displayed in Figure 5.1. Then, there is an inference phase at time t_{attack}, as shown in Figure 5.2. Prior to these phases, the users are interacting with the recommender system frequently, providing new information and facilitating the system in learning the users' preferences. In order to train a location inference model, the attacker needs users' recommendations and location information. The recommendations are gained by intercepting them from the transaction between system and users at the time interval t_0 to t_1. During this time the recommender system can have been retrained one or several times. The system users also own a social media account, where their location information is published. The attacker fetches the location information and uses it as ground truth to train the inference model.

At time t_{attack}, the training of the location inference model is complete and the attacker has chosen a target user. Unlike the previously mentioned users, this user has not published their location information at their social media account. Once the user interacts with the system, the attacker intercepts the user's recommendations in the transaction. Then, the recommendations are fed into the location inference model, and the user's location is predicted.

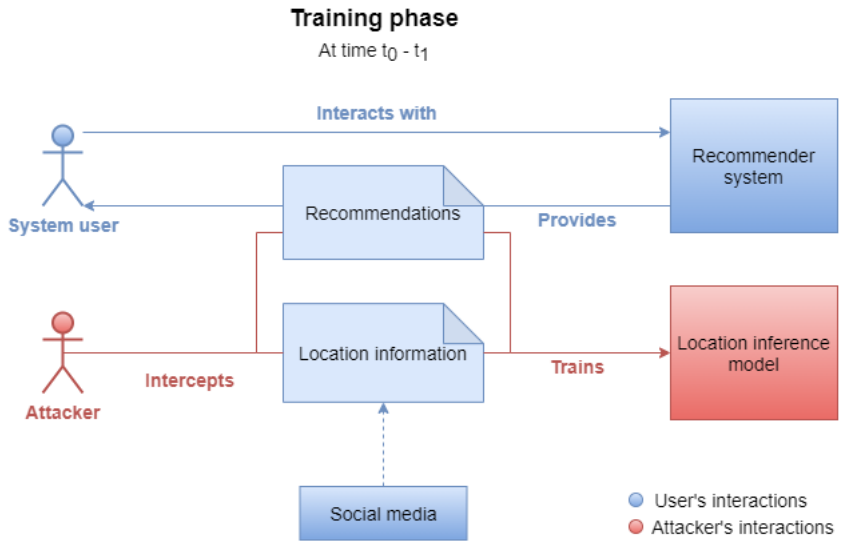


Figure 5.1: Training phase of attack

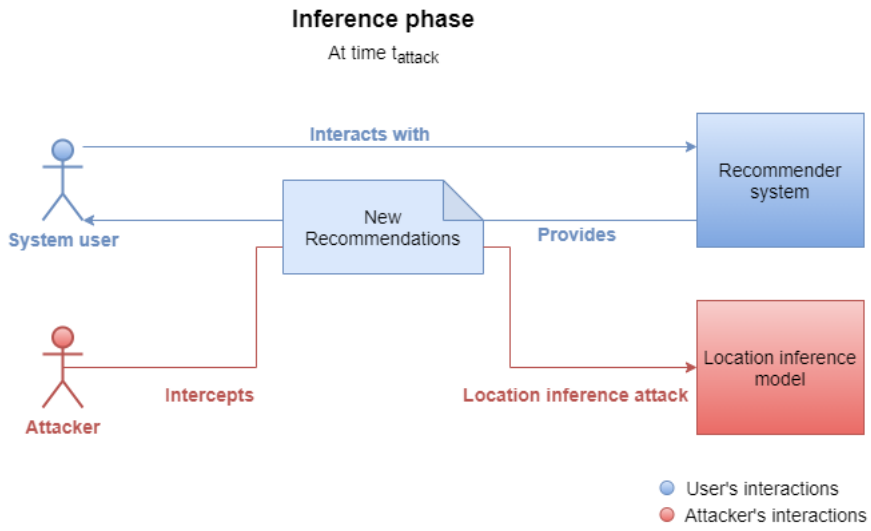


Figure 5.2: Inference phase of attack

5.2.2 Deviations from Threat Model

In order to keep the experiments simple and feasible to execute, there are some deviations from the threat model. First, in the model, the recommendations are generated and collected by the attacker over a time interval t_0 to t_1. The recommender system can have been retrained one or several times in this time period. In the experiments, the recommender system is trained once at t_0, and then the recommendations are generated and collected once for all users. Second, in the attack model, the inference attack is carried out for a target user at an arbitrary time t_{attack} after the training phase is complete. For the experiments, the recommender system is retrained once at t_1 to include new interaction data. Then, new recommendations are generated and collected for all users, which will be fed to the inference model. In order to stay true to the attack model in terms of separating users for training and testing the inference model and still infer the location of all users, k-fold cross-validation is used. A full description of the experiments is provided in Section 5.8.

5.3 Recommendation Model

The first step in the experiments is to generate some context-aware recommendations, which will be used to both train and evaluate the classification models. In order to achieve this, a Factorization Machine (FM) model is used, which has been adopted for collaborative filtering with implicit interaction data. Bayesian Personalized Ranking (BPR) is used to learn model weights via Stochastic Gradient Descent (SGD). The model was chosen as it is simple to use and achieves decent results even on sparse data. In addition, the model allows auxiliary user-

/item features, which is essential in context-aware recommendation. The python package RankFM is used for the implementation of the model, with the parameters *loss=bpr* and *factors=15*. All other parameters can be assumed to use the default value. The recommender model was run several times using different values for all parameters, and these turned out to give the best results. Read more about the model and settings in the RankFM documentation[7].

The interaction data is fed to the model in the format of *<user, item>* pairs. Additional user features are included as one hot encodings. For the MovieLens experiments, the included attributes are; gender, age, occupation and location, while the BookCrossing experiment only includes the location attribute. These attributes are included as user features under the assumption that for a user attribute to be inferred from recommendations, the recommendations must have been generated using that attribute. In other words, if recommendations are generated using contextual information, there is a chance for a trace to exist in the recommendations. This causes the potential threat to be investigated in the thesis. After the model is trained, the recommender system generates the top 50 recommendations for each user, based on a new set of user interactions.

5.4 Classification Models

To execute the attack in the threat model, any suitable multiclass classification algorithm can be used. This is one of the reasons why it is hard to create a robust defence against inference attacks. Therefore, one of the topics to be investigated in this thesis is how the inference results will vary between models, as it will give a broader perspective on the potential threat. Four different classification models are used in the experiments, namely Random Forest, Logistic Regression, Support Vector Machine, and ZeroR. The algorithms were chosen due to their popularity and suitability for multiclass tasks. They can be found in related works, such as BlurMe [34] and AttriGuard [14].

For each classifier, a set of hyperparameters are tuned. There is no formal rule of which parameters to tune, or the exact values to use. For this thesis, common parameters were chosen for each algorithm. First, test experiments were run with different parameters and a broad set of values, then the subsets which seemed to make the greatest impact were chosen. For the experiment for the BookCrossing dataset, uses a subset of the hyperparameters which are used for the MovieLens experiments. This is due to time limitations, as the BookCrossing dataset is quite a bit larger and causes a much longer running time. The hyperparameters and settings for each classification model are listed below.

[7] https://rankfm.readthedocs.io/en/latest/rankfm.html

Table 5.1: Settings for random forest

Parameter	Values for Ex1	Values for Ex2
n_estimators	100, 250, 500, 1000, 1500	100, 250
max_features	10, 15, 20	10, 20
min_samples_leaf	1, 2, 4	2, 4
min_samples_split	2, 10, 15, 20	10, 20
random_state	16	16

5.4.1 Random Forest

Random Forest is a highly popular algorithm for multiclass problems. A description is given in Section 2.6.1, and it will therefore not be repeated here. The parameters used for hyperparameter tuning are presented in Table 5.1. N_estimators refers to the number of trees to be built in the forest. In general, an increase in trees means more models which can make predictions, and will often lead to a more confident model as a whole. In this thesis, the final prediction is calculated by averaging the predictions of all trees, instead of majority vote. Max_features refers to the number of features which can be considered each time a tree splits. The min_sample_split and the min_sample_leaf represent the minimum number of samples required to split a decision node and to become a leaf node, respectively. All these values were chosen based on the results from test experiments. In addition, the random_state is set for reproducibility. For more details regarding parameters, see Scikit-learn's documentation[8]. Any other parameter can be assumed to have the default value stated in the documentation.

5.4.2 Logistic Regression

Logistic regression is described in detail in Section 2.6.2. The algorithm was originally meant for binary classification tasks, but can be used for multiclass tasks by applying the one versus all strategy. In Table 5.2 is an overview over the parameters to be tuned. Solver refers to the algorithm which will be used in the optimization problem. These solvers were chosen as they were suitable for multiclass problems and showed promising results in the test experiments. Max_iter is the maximum number of iterations the solver can use to converge, and after running test experiments, it was clear that a high number is needed. C controls the regularization strength, where smaller values of C will increase regularization strength and reduce the chance of overfitting. Penalty describes the norm used for penalization. Both the 'newton-cg' and the 'sag' solver only support the 'l2' norm. For more information regarding the parameters, visit Scikit-learn's documentation[9].

[8]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[9]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Table 5.2: Settings for logistic regression

Parameter	Values for Ex1	Values for Ex2
solver	newton-cg, sag, saga	newton-cg
c	0.1, 0.01, 0.001	0.1, 0.01, 0.001
penalty	l2	l2
max_iter	10000	10000
random_state	16	16

Table 5.3: Settings for SVM

Parameter	Values for Ex1	Values for Ex2
kernel	linear, poly, rbf, sigmoid	poly, rbf
c	100, 10, 1.0, 0.1, 0.001	1.0, 0.1, 0.01
probability	True	True
random_state	16	16

5.4.3 Support Vector Machine (SVM)

The description of SVM can be found in Section 2.6.3. Like logistic regression, SVM is a binary classification algorithm, but can be used for multiclass classification using the one vs all strategy. The settings are displayed in Table 5.3. Several different kernels are tested for these experiments. These are similarity functions used to measure the difference between an input and a target value. The c-parameter has the same function as in logistic regression; to control the regularization strength. In order to measure the AUC-score for the classification models, probability estimates are needed. These are not included by default for SVM, and must therefore be explicitly stated. Scikit-learn gives an overview of the parameters[10].

5.4.4 ZeroR

The ZeroR classifier is used as a baseline in the experiments, to give intuition about the significance of the classification models' results. The model is very fast and simple. If a model does not perform much better than the ZeroR classifier, the results are not great. Why use a complex model and spend time on hyperparameter tuning, if a simple algorithm will give better results? The ZeroR classification model is described in Section 2.6.4. In the Scikit documentation, ZeroR is referred to as the Dummy classifier, with the strategy 'most_frequent'. More information can be found in the documentation [11].

[10] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[11] https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html

5.5 Data Splitting Strategies

As the experiments consist of two parts; one recommendation task and one classification task, there is a need for two different data splitting strategies. The chosen strategies are presented in the following sections.

5.5.1 Splits for Recommendation Model

There are many aspects to consider when choosing a splitting strategy. Each strategy has its own strengths and weaknesses, and it is important to find one which fits with the scenario of the project. For the threat model described in the thesis, the ideal choice is the temporal global splitting strategy. In order for the experiments to provide any evidence related to a real-world threat, a relatively realistic splitting strategy is needed. In addition, time is an important aspect in the threat model, as users interact with the system over time and the attacker patiently collects data from these interactions. Unfortunately, there are some limitations due to the chosen datasets and experimental setup. The second part of the experiments is a classification task, and in order to make accurate predictions, the model needs a lot of training data. As the MovieLens 100K dataset is relatively small, the removal of users under the splitting process can have a significant impact on the classification results. Therefore, the temporal user splitting strategy is chosen instead. Although the strategy is a tad less realistic and can suffer leakage problems, it still gives the experiment a temporal aspect and ensures all users to be included in both training and test set. There is also an experiment utilizing the BookCrossing dataset. The dataset does not have any temporal data, and therefore each user's items will be split using a random splitting strategy instead. A more detailed description of both strategies can be found in Section 2.7.1.

Two rounds of recommendation generation are required in these experiments: one set of recommendations used to train the inference model, and another set used to infer the location information. To train and test the recommender, a user-item interaction matrix is used, where items are sorted on time. The matrix will be split on the users' items, based on the temporal user splitting strategy. This means that a point t in time does not represent an actual timestamp, but rather the point where each user individually has interacted with a certain percentage of their items. The strategy ensures a sufficient number of item interactions at each point t. Another important notion, is that the test sets need to be of a certain size, as they are not only used for model evaluation, but also to generate data for the classification model. Unfortunately, this also means less data for the recommender system to train on and might have an impact on model performance. The BookCrossing experiment follows the same approach, but chooses a number of users randomly for each set, instead of using the temporal aspect.

For round one, the recommender system is trained at time t_0, using a training set consisting of 50 % of the users' first item interactions. A test set consisting of

25 % of the users' next item interactions is used to generate recommendations for the users. Then, round two of recommendation generation begins at time t_1. The recommender system's second training set consists of the 75 % of the users' first item interactions, i.e. all the users' item interactions up until time t_1. The test set consists of the users' 25 % newest items, and is used to generate the new recommendations used for location inference. The splittings for both rounds are included in Figure 5.3.

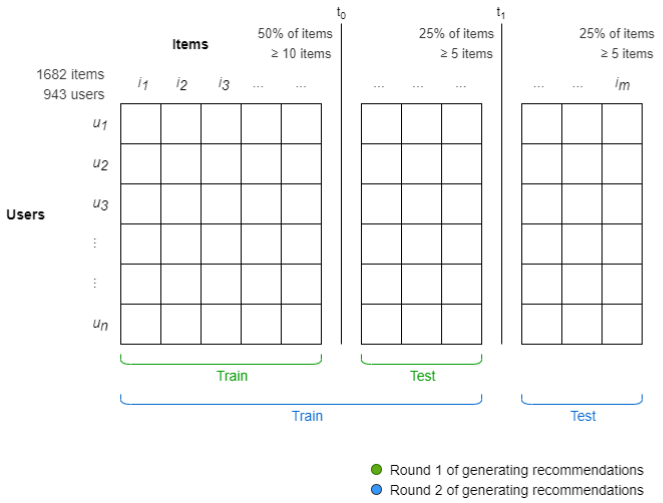


Figure 5.3: Splits for Recommender Model

5.5.2 Splits for Classification Model

The same splitting strategies used for the recommendation task were chosen for the classification task, as it creates consistency throughout the experiment. Non-random sampling with a temporal aspect is also the strategy which most accurately corresponds to the attack scenario, as the attacker will have to train the inference model on data collected before the actual attack. The recommender system has produced two sets of Top-K recommended items for each user. These can be considered a natural partition for the classifier's training and test set. Notice that the training set also includes the users' location attribute, as it is needed to train the classification model. In addition, some auxiliary attributes are included. Figure 5.4 displays the partition.

5.6 Resampling strategy

For this thesis, Nested Cross-Validation is the chosen resampling method. The approach is chosen due to its popularity and ability to reduce bias under model evaluation, when limited data is available. In addition, it is a suitable strategy for hyperparameter tuning, as it will decrease the chance of overfitting. Although a

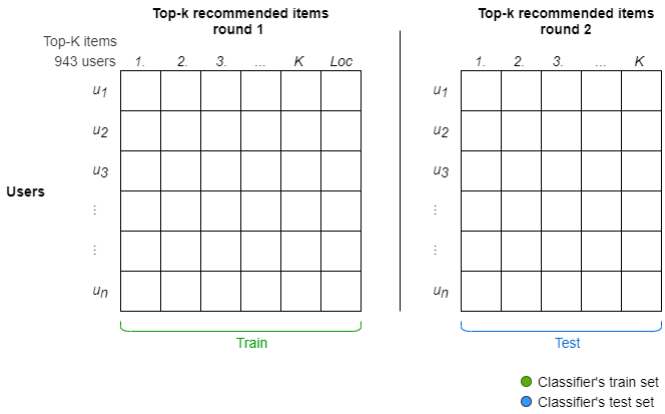


Figure 5.4: Splits for Classifier Model

stratified CV approach would be tactical for the experiment, it does not fit the attack scenario. It is not realistic for the attacker to be able to strategically choose what classes to include in a test set, as that would imply that the attacker already knows the class values they were trying to predict. The method is described in Section 2.8.1.

The outer round of CV is used for model evaluation and will have k=5 folds, while the inner CV is used for parameter tuning with k=3 folds. They are displayed in Figure 5.5 and Figure 5.6, respectively. There are two reasons for this. First, to ensure that both the training and test set is large enough to contain all classes, as it is important for model evaluation. Second, to lower the computational cost of the experiments. Unlike the splits for the recommender system, the folds used for classifier cross-validation will be split on users, who are chosen randomly. For each iteration, four folds of users will be used for training the classifier and one fold for testing. Notice that the users in the training folds in each iteration correspond to the top-k recommendations from the classifier's training set, while the users in the test fold correspond to the top-k recommendations from the classifier's test set. This is illustrated in Figure 5.7.

The process of optimizing hyperparameters is nested under the model skill evaluation process. The training set for each iteration in the outer CV is split up into three folds, constituting a new training set and a validation set. These are used for hyperparameter tuning, where Grid Search is used to find an optimal set of hyperparameters for the model. Although Random Search is faster and has proven to generally perform decently, grid search is guaranteed to find the best set of parameters from the options available. In addition, the number of folds and the dataset used for the experiments are relatively small, making the computational cost reasonable.

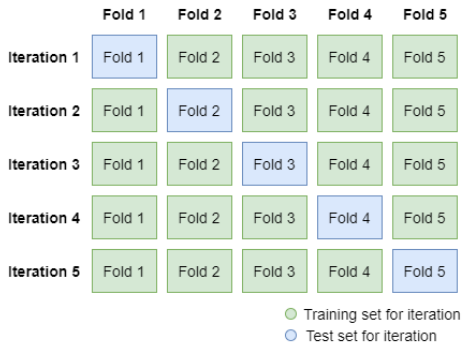


Figure 5.5: 5-fold Cross-Validation for Model Evaluation.

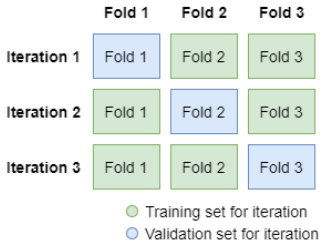


Figure 5.6: 3-fold Cross-Validation for Hyperparameter Tuning.

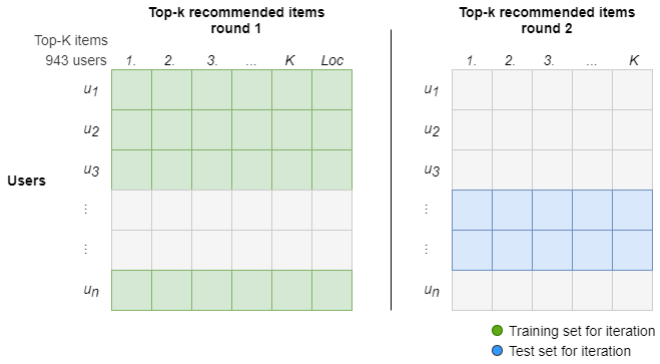


Figure 5.7: Example of Data Splitting for an Iteration in Outer Cross-Validation.

5.7 Hypotheses

These are the hypotheses derived from the research questions for the experiments in this thesis.

H1 Information leakage: The personalized recommendations will leak information about the user's location attribute, which a trained classifier will be able to infer.

H2 Dataset characteristics: The dataset's characteristics will have a significant impact on the accuracy of location inference.

H3 Data splitting strategy: The data splitting strategy for the location attribute will have an impact on the inference results.

H4 Classification models: The success of location inference will vary between the different classification models.

The hypotheses summarizes the expectations regarding the outcome of the experiments. Recommendations are expected to leak location information as there

have been other works which have been able to infer private attributes from recommendations, using auxiliary information. The dataset characteristics is likely have an impact on the inference results, as the importance and role of the location attribute will vary between datasets. In this project, both the MovieLens 100K dataset and the BookCrossing dataset are utilized, where BookCrossing is predicted to give the best results, due to the international scope of the location information. As there are endless ways to split the location attribute, it makes sense that some are better than others. One important factor is the sample distribution across classes, as it will be harder to infer information from sparse classes. In the experiments for this thesis, the two-way split, the five-way split, and the ten-way split are used under classification. As the ten-way split groups location information based on similarity in user behaviour, it is predicted to give the best results. The five-way split may give the second best performance at it allows more granularity and distinctions between the classes. It is also likely that there are some variations between the different classification models, as they build upon different techniques which can be more or less suitable for this particular task.

5.8 Experiments

This thesis aims to explore the leakage of location information in personalized recommendations. Based on the threat model presented in Section 5.2 and the hypotheses stated in the previous section, a series of experiments are performed to investigate the topic. There are two main experiments. Experiment 1 uses the MovieLens dataset and consists of four sub-experiments. First, two lists of recommended items are generated using a FM model adapted for collaborative-filtering with BPR loss, as described in Section, and the temporal splitting strategy from Section 5.5.1. Both interaction data and contextual information are used as input for the recommender model. These lists of recommended items will be used in the inference process. The evaluation of the recommender model for both rounds of recommendations are presented in experiment 1.0. Then, three inference experiments; experiment 1.1, 1.2, and 1.3, are carried out using three different splits for the location attribute: The two-way split, the five-way split and the ten-way split. In addition to inference of the location attribute, the user's gender, age, and occupation attribute is also attempted inferred. The recommendations generated in experiment 1.0 are partitioned into training and test set according to the strategy mentioned in Section 5.5.2. For the inference task, four different classification models are used, namely random forest, SVM, logistic regression, and ZeroR. To enhance attribute inference, the remaining attributes which are not currently being inferred are used as auxiliary features for the other attributes. Nested 5-fold cross validation is used for model evaluation and hyperparameter tuning, as described in Section 5.6. The main differences between the sub-experiments are summarized in Table 5.4.

In order to investigate how the characteristics of a dataset impacts the inference

Table 5.4: Description of sub-experiments

Experiment	Task	Location Split
Experiment 1.0	Recommendation	State
Experiment 1.1	Classification	Two-way
Experiment 1.2	Classification	Five-way
Experiment 1.3	Classification	Ten-way
Experiment 2.0	Recommendation	Country
Experiment 2.1	Classification	USA vs. the Rest

Table 5.5: Description of experiments

Experiment	Dataset	Recommendation Split	Attributes
Experiment 1	MovieLens	Temporal user split	Gender, age, occupation, location
Experiment 2	BookCrossing	Random split	Location

results, the same experiment is performed with a second dataset, namely the BookCrossing dataset. This experiment is referred to as experiment 2 and has a few differences from experiment 1. These differences are summarized in Table 5.5. First, like in experiment 1, the recommendations are generated and evaluated in experiment 2.0. As the BookCrossing dataset does not have a temporal aspect, the interaction data is instead split using the random splitting strategy for the recommendation model. In addition, no other contextual information is utilized except the user's location. Then, in experiment 2.1, the location attribute is attempted inferred using the USA vs. rest split. The classification models and the inference process is the same as described in experiment 1, except that no contextual information is utilized under classification.

For both experiment 1 and 2, the recommender model is evaluated using the precision at-k, recall-at-k, and hit rate metrics. The same set of recommendations are used for all experiments performed on the movielens dataset, in order to make the results comparable between the experiments. In addition, the classification models were given the same random state to reduce randomness between the results. This way, the only difference between the MovieLens experiments, is the partition strategy of the location attribute used for the classification model. The classification models are evaluated using 5-fold cross-validation, and the ROC-AUC and F1 metrics. The recommender system is not evaluated using cross-validation, as a fixed dataset is used as input and therefore are not expecting too much variance in its performance.

Chapter 6

Results

In this thesis, two main experiments are performed, which are described in Section 5.8. Experiment 1 consists of four sub-experiments and utilizes the MovieLens dataset. In experiment 1.0, a recommender model will be used to generate two lists of recommended items for the inference process. Then, three inference experiments (1.1, 1.2, and 1.3) are carried out using three different splits for the location attribute under classification: The *two-way* split, the *five-way* split and the *ten-way* split. In experiment 2, the BookCrossing dataset is utilized, and unlike experiment 1, the experiment only consists of two parts. First, the recommendations are generated and evaluated in experiment 2.0. Then, the location attribute is attempted inferred in experiment 2.1 using the *USA vs. rest* split. The same inference models are used in both experiments, namely zeroR, logistic regression, SVM, and random forest. This chapter will present the results from the performed experiments. A more detailed description of the methods, splitting strategies, and metrics, can be found in Chapter 5. Visit Chapter 2 for theory related to the recommender model and classification models. Chapter 4 describes the datasets used in the experiments, as well as different attribute representations used for the recommender model and the classifier models.

6.1 Experiment 1.0: Recommendation Generation

Table 6.1 displays the results of the recommender model in Experiment 1.0. As shown, P@K is 8.7 % for round 1 of recommendation generation, and a bit lower for round 2. When interpreting these results, it is important to notice that it is not feasible to get a P@K score close to 100 % for this experimental setup. P@K is calculated by dividing the number of items existing in both the test set and the top-K recommendations by the number of K. In the case of the MovieLens 100K dataset, each user is only guaranteed to have at least 20 ratings. For a user that only has rated 20 items, the test set will consist of 5 items. K=50 in this experiment, meaning that the highest P@K which can be achieved for such users

Table 6.1: Recommender results for Experiment 1.0

	P@K	R@K	HR
Round 1	0.08653	0.2611	0.9088
Round 2	0.07340	0.2136	0.8897

is 10 %. The users who have rated more items can achieve a higher score. As for the R@K score, 100 % is the highest achievable score for all users. The results from this experiment says that on average, 1 out of 4 items in the test set is ranked in the top-50 recommendations in round 1, and 1 out of 5 items is ranked in the top-50 recommendations in round 2. The hit rate brings another perspective, and states that 1 item appeared in both the test set and the top-50 recommendations on average for each user. It is important to notice that there are variations in the sizes of the users' training and test set, which can cause the scores between the users to vary. Another interesting notion is that the results are lower for the second round. This happens even though the training set is bigger for round 2 (75%) than for round 1 (40%), and the test size remains the same (25 %).

6.2 Experiment 1.1: Two-way Split

The results of experiment 1.1, where the two-way split is used for the location attribute under classification, can be found in Table 6.2. Notice the difference in scores between AUC and F1. For example, the F1 score of the location attribute is 62.7 % for SVM, which states that the classification model infers the right class more often than not. However, the AUC score for SVM is only 51.4 %. The reason behind this is the class imbalance in the dataset. If the F1 score (62.7 %) is compared to the most location class with the most samples for the two-way split (west: 64.5 %), a high correlation is revealed. In other words, the SVM performs just as good as the ZeroR algorithm, which only chooses the most frequent class from the training set for all new input (the west class in this case). The AUC metric takes the class imbalance into account and provides the score of 51.4 %. . When a model has an AUC score at 50%, it means that the model has no ability to discriminate between the different classes. In other words, the model is not able to detect a strong signal regarding the given attribute from the recommendations. By looking at the table, it is clear that the statement holds for logistic regression and random forest as well, which disagrees with hypothesis H1 regarding location information leaking from recommendations.

There are some differences in performance between the models. The SVM model performs worse than the ZeroR model for all attributes except location, where they have about the same score. Logistic regression seems to be slightly better than ZeroR at discriminating between classes for both location and occupation. Random forest is the best performing model in this experiment, both for the location attribute and for the experiment as a whole. It is also the model where the

Table 6.2: Inference Results for 2-way Location Split

	AUC				F1			
Model	Gen	Age	Occ	Loc	Gen	Age	Occ	Loc
ZeroR	0.5000	0.5000	0.5000	0.5000	0.7105	0.5769	0.2079	0.6447
Logistic Regression	0.4884	0.4643	0.4215	0.5249	0.7020	0.5238	0.1951	0.6225
SVM	0.4314	0.4589	0.5201	0.5137	0.6956	0.5482	0.2079	0.6267
Random Forest	0.6162	0.6124	0.4095	0.5288	0.7095	0.5801	0.2418	0.6352

Table 6.3: Inference Results for 5-way Location Split

	AUC				F1			
Model	Gen	Age	Occ	Loc	Gen	Age	Occ	Loc
ZeroR	0.5000	0.5000	0.5000	0.5000	0.7105	0.5769	0.2079	0.2735
Logistic Regression	0.4885	0.4621	0.4210	0.5362	0.7010	0.5228	0.1930	0.2513
SVM	0.4322	0.4710	0.5391	0.5074	0.6967	0.5492	0.2079	0.2725
Random Forest	0.5490	0.5677	0.4494	0.5814	0.7137	0.5748	0.2153	0.2820

scores vary the most between the attributes, with AUC scores which range from 41.0 % for the occupation attribute to 61.6 % for the gender attribute.

6.3 Experiment 1.2: Five-way Split

In experiment 1.2 the location classes represent five regions of the United States. The results can be found in Table 6.3. When looking at the AUC score for location inference, it is revealed that all models perform better than zeroR when inferring the location attribute once again. The AUC score of SVM decreases a bit from experiment 1.1, while logistic regession and random forest both increase. In fact, random forest achieves the highest location inference score in experiment 1, with an increase in the AUC score from 52.9 % to 58.1 %. Another observation is that the F1 score has dropped significantly for the location attribute from experiment 1.1 to experiment 1.2. This is due to the five-way split including more location classes which the users can be distributed between. By comparing the F1 scores (25.1 %-28.2 %) to the distribution of the most dominant class, namely the 'west' class (27.4 %), it is clear that there still is a strong correlation between the F1 score and class distribution.

An unexpected observation is that although the only difference between experiment 1.1 and experiment 1.2 is the partition of the location information, there is a difference in the inference results of the other attributes. This is particularly visible when comparing the results from random forest for the gender and age attribute, where the AUC scores decrease from 61.6 % and 61.2 % to 54.9 % and 56.8 %, respectively. The difference for logistic regression (≤ 0.0022) and SVC (≤ 0.019) are minor.

Table 6.4: Inference Results for 10-way Location Split Based on User Similarity

	AUC				F1			
Model	Gen	Age	Occ	Loc	Gen	Age	Occ	Loc
ZeroR	0.5000	0.5000	0.5000	0.5000	0.7105	0.5769	0.2079	0.1018
Logistic Regression	0.4887	0.4705	0.4217	0.4895	0.7041	0.5249	0.1930	0.1050
SVM	0.4563	0.4765	0.4795	0.5022	0.7009	0.5589	0.2079	0.1008
Random Forest	0.5665	0.5275	0.5011	0.5400	0.7084	0.5769	0.2132	0.1241

6.4 Experiment 1.3: The Ten-way Split

Experiment 1.3 uses a ten-way location split based on user similarity, and is a bit more exploratory than the others. As the states were grouped based on users with the most similar behaviours, the split is expected to give the best results. The results presented in Table 6.4 indicate that there is no improvement. In fact, by studying Figure 6.2, which expresses how the location inference performance changes across the experiments, it is revealed that both logistic regression and SVM achieved their lowest scores at 49.0 % and 50.2 %. Random forest gained its next highest score at 54.0 % These results agrees with hypothesis H3, stating that the splitting of the location attribute will impact the inference results. In addition, the standard deviation for the models are rather high, not only in this experiment, but in experiment 1.1 and 1.2 as well. The results seem to vary quite a bit between the iterations as well. Read more about the standard deviation in these results in Section 6.7. Although the classification models performed better than zeroR in all experiments, except for logistic regression in experiment 1.3, the results of experiment 1 disagree with hypothesis H1, stating there being a strong signal from the users' location attribute in the recommendations.

As the other attributes were included when looking at user similarity, it would be possible that the new location split could be more indicative of the other attributes. Then, their results could be increased when using the location attribute as an auxiliary attribute, although a big effect cannot be expected. When looking at Figure 6.1, which displays how the inference of the auxiliary attributes changes between the experiments, it is clear that there is no significant improvement. Even so, it is still interesting that the inference score for the other attributes continue to change between the experiments. Logistic regression and SVM received relatively stable scores across the experiments, with the exception of SVM in experiment 1.3 where it dropped a bit. Random forest's performance varied more. In experiment 1.3, random forest achieved its best score for the occupation attribute (50.1 %) and its worst score for the age attribute (52.8 %). Across the experiments, random forest gave the best performance for the gender and age attribute, being the only model to outperform zeroR. SVM achieved the best scores for the occupation attribute, and the worst scores for the gender attribute. The worst performance for age and occupation was given by logistic regression.

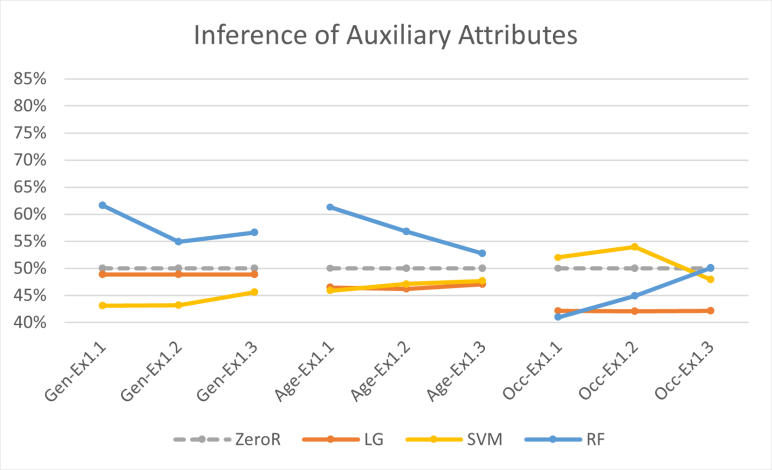


Figure 6.1: Inference Results for Auxiliary Attributes

Table 6.5: Recommender results for Ex 2

	P@K	R@K	HR
Round 1	0.008262	0.01709	0.2604
Round 2	0.009711	0.01888	0.2924

6.5 Experiment 2.0: Recommendation Generation

Experiment 2 uses the BookCrossing dataset and a slightly different strategy when splitting the training and test set for the recommender system. While the MovieLens experiment splits each user's items based on a temporal aspect, the BookCrossing experiment splits the items randomly for each user. When looking at the evaluation of the recommender model in Table 6.5, the main observation is that the scores are much lower than for MovieLens. The P@K score is below 1.0 % for both rounds and corresponds to roughly $\frac{1}{10}$ of the P@K score from experiment 1.0. R@K has dropped even lower, where the score of round 1 is less than 7.0 % of the R@K score from experiment 1.0. The HR has dropped to about $\frac{1}{3}$ of the score. Another observation is that the second round performed better than the first round, which is the opposite of what was observed in experiment 1.0.

6.6 Experiment 2.1: USA vs. the Rest

For experiment 2, only one class partition is used for the location attribute, namely *USA vs. the rest*. Unlike the MovieLens dataset, BookCrossing includes location information on an international level. As there is assumed to be bigger cultural differences between countries than inside one single country, this experiment is expected to achieve higher inference results. The location attribute is the only attribute attempted inferred, meaning that no other contextual information is used

under the classification. There are also less parameter values included in the hyperparameter tuning in this experiment.

The inference results for experiment 2.1 are presented in Table 6.6. In this experiment, logistic regression, SVM and random forest all perform better than the zeroR model, which means that the models are able to discriminate between different classes. Unlike the findings in experiment 1, this agrees with hypothesis H1, claiming that location information can leak through recommendations. In this experiment, SVM seems to be the best performing model, although there is not a big performance difference between the models in terms of AUC scores. For the F1 scores, logistic regression and SVM have significantly higher scores than zeroR. Notice that random forest has a AUC score similar to logistic regression and SVM, even though the F1 score is the same as for zeroR. When comparing the results of experiment 1 and experiment 2, it is clear that experiment 2 achieves much better inference results. Even the worst AUC score from experiment 2 (79.4 %) is higher than the best location inference results from experiment 1 (58.1 %). This observation agrees with hypothesis H2, which stated that the characteristics of a dataset will have an impact on the inference results. As only one inference experiment is executed using the BookCrossing dataset, it will not be possible to compare how different splitting strategies for the location attribute impacts the results. However, as the location attribute is split with the class distributions taken into account and the experiment achieves decent results, it can be reasonable to assume that the chosen splitting strategy is suitable in this setting.

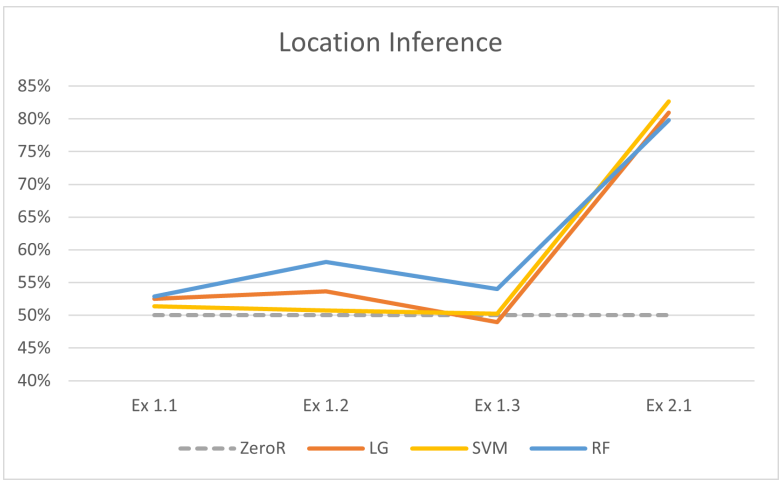


Figure 6.2: Inference Results for Location Attribute

6.7 Standard Deviation

Table 6.3 displays the the standard deviation throughout the experiments, and how it differs between attributes and classification models. The standard deviation

Table 6.6: Inference Results for USA vs. Rest Split

	Loc	
Model	AUC	F1
ZeroR	0.5000	0.6505
Logistic Regression	0.8097	0.7907
SVM	0.8266	0.7284
Random Forest	0.7984	0.6505

changes between the attributes. The gender and age attributes, which achieves comparable inference results, have standard deviation scores within the same range 2 %-5 %. For the age attribute, a big increase is visible from experiment 1.1 to 1.3 for both SVM and random forest. The occupation attribute has the overall highest variance, with the range of 4 %-9 %, and also the biggest difference between the models. As for the location attribute, the standard deviation stays around 3 % in experiment 1, with the exception of random forest in experiment 1.2 (6.7 %) and SVM in experiment 1.3 (5.8 %). As both random forest and SVM's score vary across the experiments and attributes, it is not easy to conclude on which model has the lowest variation. A noteworthy observation is that the standard deviation decreases significantly for experiment 2.1, where the Book-Crossing dataset is used and the inference results are the highest. In general, the figure indicate that there are some variance in the results between cross-validation iterations, especially in experiment 1.

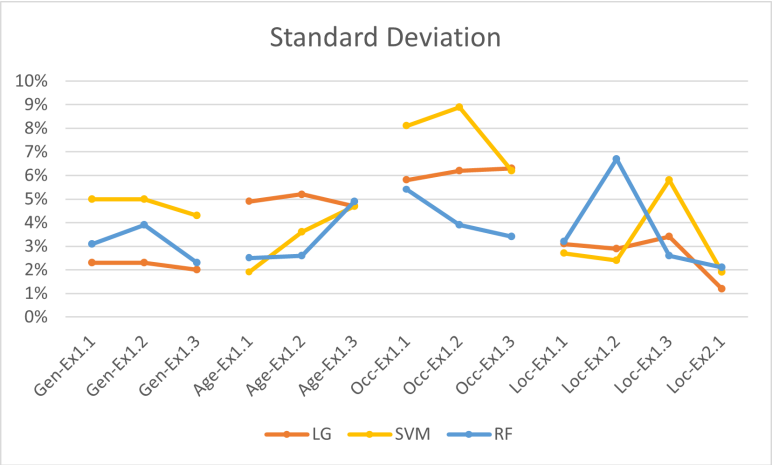


Figure 6.3: Standard Deviation for Inference Results

Chapter 7

Discussion

In this chapter, the experiment results presented in Chapter 6 are discussed. First, the role of recommendations in the inference process is discussed, and then the focus shifts to attempt to answer the research questions, one by one. The topic of location leakage in recommendations will be investigated, and then the impact of the dataset characteristics and splitting strategies are discussed. At last, an evaluation of the classification models and the implications of the findings will be included.

7.1 Recommendations

Based on the results presented in Section 6.1, the recommender model performance proves to be reasonable for the MovieLens dataset. The second round of recommendation generation has lower scores than the first round, even though the training set consists of 75 % of the ratings for the second round, and only 40 % for the first round. Both the dataset and splits are fixed and will be the same for each run of the experiment, so this observation is not likely to be arbitrary and change if the experiment is rerun. Each user's ratings are split into training and test sets based on time. Therefore, the training and test set represent two different points in time (for each user), causing there to be new items in the test sets for each round. The reason behind the observation may simply be that there is a greater overlap between the items in the training and test set in the first round than the second round, as there are fewer items in circulation. Another aspect to consider is if there is a data leak. As the point in time where the ratings are split varies between users, an item might exist in the test set of one user but in the training set of others, causing information about future items to potentially leak into the present for that user. However, this will usually cause overly optimistic performance scores, and by looking at the results in Table 6.1, this does not seem to be the case.

The results of experiment 2.0, found in Section 6.5, shows that the recommender

model has a lower performance when using the BookCrossing dataset. As the dataset does not include a timestamp for the ratings, the data is split into training and test sets using a random splitting strategy. But the splitting strategy should not cause such low scores by itself. One possible explanation can be found by looking at the sparsity levels of the training and test sets across the experiments. The sparsity level of both training and test sets for experiment 2.0 are quite high, with 99.9 % sparsity, while they range from 95.3 % to 98.0 % for experiment 1.0. The recommender model would probably achieve a higher performance score for both dataset if sparsity is increased. As for the BookCrossing dataset, the sparsity is partially due to the subsetting of the dataset, where only implicit ratings are included. In order to alleviate this problem, it would be possible to simply translate the explicit ratings into implicit ratings and include them in the reduced dataset.

As the recommender model performs worse when using the BookCrossing dataset, one might assume the inference results would fall too. However, the opposite is observed, with the inference model in experiment 2.1 performing far better than in the inference models in experiment 1. The evaluation scores represent the system's ability to recommend items which exist in the user's test set. As the ratings are implicit, it is only known that the user interacted with the items originally, not whether they actually liked them or not. In addition, if the user has no interaction with an item, their opinion toward it is unknown as well and there is a possibility that the user will like it a lot. In other words, it would be possible for a recommender system to produce recommendations fit to a user's preferences without recommending items from the test set. On the other hand, it would be assuring to have a high performance score for the recommender model, as the recommendations would give a more accurate representation of the user's preferences. Then, it would be safer to conclude something about the relationship between a user's recommendations and location inference. However, there exist real-world recommender systems of all quality levels. One might argue the potential threat is greater if a signal which can be used for location inference, can be found in recommendations generated by a somewhat inaccurate recommender system. As this topic is not covered by the research questions, it will be left as an afterthought.

7.2 Location Leakage in Recommendations

The main aspect of this thesis is to investigate whether personalized recommendations leak the user's location information. Hypothesis H1 expressed the expectation of a leak. The inference results presented in Chapter 6 indicate that recommendations generated from the MovieLens dataset did not leak a strong signal about the user's location, while the recommendations generated using the BookCrossing dataset did. One reason as to why the inference results from experiment 1 were not great, might be that a user's recommendations are not indicative enough of the user's location. The classification models were not able to find a strong sig-

nal for the auxiliary attributes either. Random forest was able to pick up a signal from gender and age, but no classification model was able to infer the occupation of the user. There are seemingly no related works which try to infer user attributes from only recommendations. Although there might be many reasons for this, it may indicate that there is not a high expectation of recommendations leaking strong signals about the user's attributes. For example, the work by Beigi et al.[4] uses recommendations for inference, but also includes other data, such as the user's interaction history. However, this argument cannot be the sole reason for experiment 1's poor results, as the experiment 2 using the BookCrossing dataset gave decent results.

Another factor is the use of contextual information in experiment 1. Both location and the auxiliary attributes were utilized by the recommender model when generating the recommendations. How much each attribute impacted the recommendations, and whether the signal of one attribute might have enhanced or obfuscated the signal of another attribute, is not certain. It would be reasonable that a recommender model can find some attributes to be more indicative of the users' preferences than others, and therefore let it impact the recommendations more. It is not given that these attributes will enhance each other's signal, as maybe a dominant attribute will end up weakening the signal of a less indicative attribute. To investigate this further, it would be possible to rerun the experiment and explore different uses of attributes in the recommendation generation process. For example, only feed the attribute to be inferred to the recommender system and see if the classification models find a stronger signal.

The classification models are also using the contextual information under training. When looking at the results for experiment 1.1, 1.2, and 1.3, the inference scores change for all attributes, even though the only difference is the splitting strategy for the location attribute under classification. This is especially visible for the random forest algorithm in Figure 6.1. This indicate that including an auxiliary attribute under classification, and the chosen data splitting strategy for the attribute, can impact the results. As to whether simply including the location attribute enhanced or obfuscated the signal of the other attributes is hard to conclude as no experiment was run without including it. Experiment 2 did not use any auxiliary attributes for the recommender model nor the classification models and achieved greater results. Although there are bigger differences between these experiments than the use of contextual information, it is worth taking into consideration. If the recommendations only give a weak indication towards the location attribute, it is possible that an attribute which is less indicative of the location might end up becoming more dominant and obfuscate the signal to some degree. Section 7.4 includes a discussion regarding the impact location splitting strategies can have on other attributes.

The impact of recommendations and contextual information have now been discussed, but there are other factors which needs to be considered in regard to this research question. This includes the datasets' characteristics, data splitting

strategies, and the classification models. These factors will be addressed in the discussion of the other research questions, as they are highly related to the main research question.

7.3 The Impact of Dataset Characteristics

When investigating the impact of the datasets' characteristics on the inference results in regard to RQ2, four aspects will be discussed. These are the dataset's size and sparsity level, imbalance in the data, the domain, and the characteristics of the location attribute, which each will be presented in the next paragraphs.

Table 4.3 displays the size and sparsity levels of the datasets, and indicates that the subset of the BookCrossing dataset contains more users and items level than the MovieLens dataset. Having a high number of users can be advantageous because it means more samples for the classification model to train on. In the case of location inference, one might wish to split the attribute into many classes to increase granularity and close in on the user's location. However, the number of classes will be limited if there are not enough users to populate these classes. Due to this, and class imbalance, which is discussed further down, some location attribute splits had to be disregarded for both datasets. The BookCrossing dataset do also have a higher level of sparsity in the interaction data, which is partially due to the high number of items and the subsetting strategy. The sparsity level of the interaction data will only impact the inference results through the generated recommendations. Despite the BookCrossing dataset's higher sparsity level, experiment 2.1 provided much better inference results, indicating that the sparsity level is not the main factor which impacts the results.

Another difference between the datasets are the domains, as the MovieLens contains movie reviews and BookCrossing contains book reviews. The movie and book domains are quite similar in nature, both providing their consumers with media for entertainment and educational purposes. There is also often an overlap in content due to books becoming movies and vice versa. There might be one relevant difference though. Even though a movie is recorded in a specific language, the use of subtitles makes it possible for consumers who live in different countries and speak different languages to enjoy the same movie. In order for a book to appeal to a consumer with a different language, the book must be translated and published in the given language. This can be both time consuming and tedious work, meaning that the book probably must have a certain level of demand before a translation is considered. In addition, when a book is published in a new language, it receives a new ISBN code[1] and will not be considered the same as the original book in the dataset. In other words, the language barrier might be stronger in the book domain and cause bigger distinctions in user preferences between countries and regions. Even so, the datasets cover different geographical

[1] https://www.isbn.org/faqs_general_questions

scopes, and the impact of this effect is therefore unclear. There is also a possibility that the users' location and their interaction history are related for the BookCrossing dataset. As one aspect of the BookCrossing application is to send and receive books, and there is a chance that users will tend to make these exchanges more often with users who are nearby or have the same language.

The user base in the two datasets have different geographical scopes; the MovieLens dataset only contains users living in the USA, while the BookCrossing dataset includes users from many different countries. This is probably one of the main factors which caused the great difference in inference results. The BookCrossing experiment achieved far better inference results than MovieLens experiments. It seems logical that there in general would be greater distinctions (language, culture, preferences etc.) between countries than between regions inside one single country, although some countries have diverse subcultures. This makes especially sense in the movie/book domains, due to the argument mention in the previous paragraph regarding language barriers. It is reasonable that the users across USA tends to have roughly the same preferences in movies.

7.4 The Impact of The Location Splitting Strategy

Hypothesis H3 states that how the location attribute is divided into classes for classification will impact the inference results. Three different splits were used for the MovieLens experiment, namely the two-way split, the five-way split, and the ten-way split, which partition the United States into several regions. The expectations prior to the experiments were that the ten-way split would achieve the highest location inference score, as the split is created using user similarity rather than just a geographical basis. However, as observed in the results, the split actually gave lower scores. The experiment using the five-way split was also expected to outperform the two-way split experiment, as the west vs. east in USA is not known to have great cultural differences. Instead, there seem to be a bigger difference between coast and country states. The five-way split is based on existing regions of the USA, includes more granularity and can therefore be expected to capture the distinctions a bit better. Both logistic regression and random forest achieved a higher AUC score for the five-way split, while SVM's score dropped. It is not a striking improvement. In general, as the location inference scores ranges from 49.0 % to 58.1 %, it is safe to conclude that none of the splits provides a strong signal. Although, the main reason for the weak signal in these experiments is suspected the lack of distinction in movie tastes across the USA, there probably exists ways to split the location classes which will enhance the signal. For example, by designing splits which has clearer cultural distinctions, such as urban vs. rural areas, or use the political differences across states as a baseline for the splits. One limitation for the choice of splitting strategy in this experiment is the location distribution in the dataset. For a further discussion on this topic, see Section 4.

Experiment 2.1, performed using the BookCrossing dataset, provided higher loca-

tion inference scores. The split provided two classes, where one contains the users living in the USA and the other contains users from other countries. Given the decent AUC scores and a reasonable class distribution (USA: 65 % vs. world: 45 %), the splitting strategy can be considered to be decent. On the other hand, as only one splitting strategy was utilized for this experiment, it is not possible to compare against other splits. Other splits were not considered due to time and limitations of the class distribution in the subset. Although the majority of users live in the USA for the subset, it would probably be possible to choose another subsetting strategy which provides a more balanced class distribution. One possibility is to turn explicit ratings into implicit ratings, which will lead to more users having at least 20 ratings and thereby be included in the subset. Another possibility is to not filter out books based on popularity, but rather on another criteria. As there seems that the majority of users are Americans, and American books often are read by an international audience, these books are probably dominating the list of most popular items in the dataset. This means that books from other countries have a bigger chance of being filtered out, causing the international users' number of ratings to drop, and thereby also decreasing the amount of international users in the subset. It would certainly be interesting to look into different splits for this dataset.

Something which might have impacted the inference results in experiment 1 is the use of a noise class. A noise class will probably not be able to capture the essence of users without a location value, because the users' real-world locations belongs under one of the other classes. In other words, such a noise class will only distract the model, which might end up predicting valid location values as the noise class. There were a certain amount of users in the MovieLens which did not have a valid location value. The location classes referred to specific geographical areas, and therefore it did not feel natural to include the noise in any of them, as it could end up obfuscating the class signal. It was not preferable to remove the users with missing values either as it would impact the inference of the other attributes and lessen comparability to other works. BookCrossing dataset included users without a specified location value as well. As the location was split on USA vs. the rest of the world, the noise class could be included in the rest class. In hindsight, a better solution for experiment 1 would be to either design the splits like in experiment 2 with a rest class, or use some strategy to distribute the noise between the classes. The noise percentage of the MovieLens dataset (4.1 %) and the BookCrossing dataset (2.4 %) are fortunately relatively low and should not have cause a great impact on the results.

7.5 Classification Model Evaluation

Research question R4 asks which classification model performs the best for location inference. By looking at Figure 6.2, it can be concluded that random forest gave the best results across the experiments, while SVM achieved the highest in-

ference score at 82.7 %. Random forest also gave the best over-all performance for the auxiliary attributes. As nested cross-validation was used for hyperparameter tuning and evaluation, and the results looks reasonable, it appears that the problem of overfitting is avoided. However, the variance was rather high. For instance, random forest achieved an AUC score of 58.1 % for the location attribute in experiment 1.2 with a standard deviation at 6.7 %. The AUC score ranged from 46.5 % to 65.3 % between the 5 cross-validation iterations. These two are dramatically different scores, as one proves to perform worse than the zeroR classifier, while the other indicate that there is a significant signal leaking from the location attribute. Although most location inference scores have a lower standard deviation ($\leq$ 4%), it is still noteworthy, and high variation can also be found for the other attributes.

There can be several reasons for the high variance in experiment 1. It appears that the inference score depends highly on which users, and thereby which recommendations, are included in the training and the test set. There might be a difference in how indicative a recommendation is to users' location, causing some iteration training sets to have a more indicative set of recommendations than others. High variance can often appear when there is a lack of data. The top-50 recommendations were included for each user in the inference process. If there is not a strong signal from the location attribute in the recommendations, increasing the number of recommendations may help. However, as the recommendations are generated from a fixed set of training data, the confidence score of any additional recommendations will be lower than the ones in the top-50 list. In other words, it is not given that including more recommendations will not have an obfuscating effect instead, as more of the inaccurate recommendations will be included for each user.

The hyperparameter tuning process might also have an impact. If model performance highly depends on the recommendations in the training set in the outer cross-validation, then the chosen set of parameters might also depend on the training set for the inner cross-validation. This can potentially lead to more variation in the parameter sets across the iterations, creating bigger difference between the models, and causing more variation in the AUC scores.

Another interesting aspect in the discussion is the use of two different sets of recommendations as training and test sets in the cross-validation. In the case of experiment 1, the recommendations were generated based on a temporal aspect. This means that recommendation set used for evaluating the classification models will have a different distribution of items than the recommendation set used for training, both due to the popularity of items changing over time and the arrival of new items. Each users' two recommendation lists contains 47.8 % of the same items on average, supporting the argument. Although, experiment 1 achieved lower AUC scores and higher variation than experiment 2, which uses a random split instead of a time-based split, the impact of this aspect is not certain. It is also worth remembering that the model evaluation uses cross-validation and splits the

recommendation sets on users, causing the user's two recommendation lists to never be included as both a training set and test set in the same iteration.

7.6 Implication of Findings

The main goal of this thesis is to investigate if a user's personalized recommendations can leak information about the user's location, even if the domain of the recommendations is not associated with location (such as restaurant and trip recommendations). The results of experiment 2 which used the BookCrossing dataset proved this statement to be true, by inferring whether the user was from the United States or another country with an AUC score of 79.8 %-82.7 %. Recommendations are usually not considered to be sensitive, especially in domains such as movies and books. Recommendations are therefore rarely encrypted in the transport between system and users and can then be obtained through an eavesdrop attack. The user's interaction history and other private information are often less accessible and might have to be obtained by an insider or by linking published datasets to external datasets to reveal the users' identity. Therefore, even if an inference attack using recommendations has lower accuracy, the attack still can be considered a threat to users' privacy due to the recommendations' high availability. The domains of books and movies are not inherently associated with location, and therefore are most users not aware of the possibility of location information leaking through their recommendations. Inference of location from a user's recommendations can therefore happen without the user's knowledge, and thereby also consent, and be considered a breach of the user's privacy.

There are some differences between these experiments and a real-world attack scenario. First, the attacker will need a huge amount of data to train the inference model, in addition to contextual information about the users from their social media, which might take a considerable amount of time and effort to gather. The attack also has no guarantee that their model has been trained on the location value of the target user unless the recommender system provides predefined classes. Even then, the number of possible location classes can be excessive. The approach used in this experiment, inference by using a classification model, is therefore not suited for a real-world location inference attack, as the number of location classes will be limited for the model to achieve high confidence in its predictions. It will depend on the granularity level the attacker wants to achieve for the location information, and on the amount of data available. In addition, there is the assumption that recommendations need to be generated using location information, in order for it to leak through the recommendations. In real-world recommendation system usually do not explicitly state how their recommendations are generated nor which information is used in the process, making it hard for an attacker to know this beforehand. The success of the attack will also depend on how distinct the user behaviour and preferences are across the different location classes, as observed in this thesis by comparing the results of the MovieLens exper-

iment and the BookCrossing experiment. The domain is also a factor, as there are domains which will be more and less indicative of a user's location than books. To summarize, the feasibility and success of such an attack in a real-world scenario relies on many factors and will need more research to be determined.

Chapter 8

Conclusion

This research aimed to explore whether information about the user's location can leak from their personalized recommendations. The topic was investigated by executing a series of experiments, where users' location attributes were attempted to be inferred from a set of generated recommendations, using a set of trained classification models. One experiment, which utilized the BookCrossing dataset to generate the recommendations, proved that it was possible to infer if a user lived in the USA from their recommended items, with an AUC score of 82.7 %. In another experiment, where the MovieLens 100K dataset was utilized and the aim was to infer which part of the USA the user lived in, the highest results was 58.1 %. It is clear that the characteristics of the dataset impacts whether it is possible to derive a strong signal from the user's location attribute or not. The key difference between the datasets which seems to cause this gap in the results, is the scope of the location information. While the BookCrossing dataset contains users from many different countries, allowing inference of the home country of the user, the MovieLens dataset only contains users from the USA, allowing inference of the region of the user. This indicates that the location attribute in the BookCrossing dataset contains more cultural and linguistic diversity, which causes a greater distinction between the location classes utilized for inference.

The results also showed that location information can be derived from domains which are not strongly associated with location-based services. Although BookCrossing falls under such a domain, namely books, the service allows users to trade books with each other, and they are therefore maybe more likely to read books promoted and traded by other users nearby. In the MovieLens dataset on the other hand, there are seemingly no such elements which can enhance the correlation, and still a weaker, but not too low, signal was derived from the location attribute.

Another aspect which was investigated was how the split of the location attribute utilized under classification impacted the inference results. To explore this topic,

three different splits of the location attribute were utilized for the MovieLens experiment, which split the country into regions based on either geographical aspects or user similarity. Unexpectedly, the results displayed only small variations between the different splits, indicating that the choice of splits did not make a great impact. However, since the highest score achieved for these splits was 58.1 %, which is not very high compared to the results of experiments using BookCrossing, it is possible that the chosen splits simply were not a great fit for the objective, and that there can be found other splits which can achieve greater results.

Four different classification models were utilized for location inference, namely random forest, SVM, and logistic regression, and ZeroR (baseline). The random forest model gave the overall best performance across the experiments, while SVM achieved the highest AUC score of 82.7 % using the BookCrossing dataset. In general, for the experiments utilizing the MovieLens dataset, the models were just barely able to outperform the ZeroR algorithm, while for the BookCrossing dataset, they achieved a much better performance.

In addition to the location attribute, other private user-attributes (gender, age, occupation) were inferred in the experiments using the MovieLens dataset. These attributes, as well as the location attribute, were also utilized as contextual information under recommendation generation and classification. This led to an unexpected finding. When the split for the location attribute was changed, not only did it impact the location inference results, but the inference scores of the other attributes as well. The biggest difference is achieved under inference of the occupation attribute by random forest, where the AUC score increased from 41.0 % in experiment 1.1 to 50.1 % in experiment 1.3. This gives rise to the question of whether the use of one attribute can be exploited to enhance or obfuscate the signal of another attribute under classification, and whether there will be a similar effect if the contextual information under recommendation generation is changed.

This research is different from previous works, such as the work of Beigi et al. [4] and Zhang et al. [5]], as the attributes are only inferred from the user's recommended items and auxiliary demographic information. Although excluding other types of behavioural information, such as the user's interaction history, in the inference process may result in lower inference scores, it is still a risk worth considering due to recommendations' high availability compared to other sorts of information.

8.1 Further work

Based on the results and findings from this thesis, there are several different directions for further research on the topic, some which are listed below.

Exploration of datasets
Location inference from recommendations was explored in this thesis in the con-

text of two datasets. The experiments utilizing these two datasets, which although having similar domains, gave vastly different results. This implies that the success of a location inference attack will highly depend on the domain of the recommendations, and the role of location information in the application. Therefore, one direction of research is to further investigate how datasets with different characteristics and location attribute scopes impact the inference results, and the implications the findings will have for real-world applications.

Exploration of location splits
Another suggestion is to look deeper into how the splitting strategy for the location attribute affects the inference results. In the case of the MovieLens experiments, three different splits were utilized and none lead to a strong signal from the location attribute under inference. However, this does not exclude the possibility that there are other splits which can cause a stronger location leak from movie recommendations. The MovieLens 1M dataset could be considered instead, as it provides a larger set of samples and may alleviate the class imbalance experienced with the MovieLens 100K dataset. This can give more leeway to design more suitable splits. In addition, the BookCrossing experiment only utilized one split for the location attribute. It may therefore be interesting to see how the results will change when other splits are used.

Exploration of contextual information
The results of this thesis showed that the defined classes for the location attribute affects the inference scores of other attributes. Therefore, a different line of research could be to look into how the use of contextual attributes under classification can be exploited to impact the inference of other attributes. It would also be possible to look into whether an impact is visible if classes are redefined or attributes are changed under recommendation generation instead.

Bibliography

[1] L. Rokach, F. Ricci and B. Shapira, *Recommender systems handbook*. Springer, 2015.

[2] N. F. Awad and M. S. Krishnan, 'The personalization privacy paradox: An empirical evaluation of information transparency and the willingness to be profiled online for personalization,' *MIS quarterly*, pp. 13–28, 2006.

[3] G. Beigi and H. Liu, 'Privacy in social media: Identification, mitigation and applications,' *arXiv preprint arXiv:1808.02191*, 2018.

[4] G. Beigi, A. Mosallanezhad, R. Guo, H. Alvari, A. Nou and H. Liu, 'Privacy-aware recommendation with private-attribute protection using adversarial learning,' in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 34–42.

[5] S. Zhang, H. Yin, T. Chen, Z. Huang, L. Cui and X. Zhang, 'Graph embedding for recommendation against attribute inference attacks,' *arXiv preprint arXiv:2101.12549*, 2021.

[6] L. Backstrom, E. Sun and C. Marlow, 'Find me if you can: Improving geographical prediction with social and spatial proximity,' in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 61–70.

[7] L. Kong, Z. Liu and Y. Huang, 'Spot: Locating social media users based on social network context,' *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1681–1684, 2014.

[8] J. McGee, J. A. Caverlee and Z. Cheng, 'A geographic study of tie strength in social media,' in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 2333–2336.

[9] Z. Cheng, J. Caverlee and K. Lee, 'You are where you tweet: A content-based approach to geo-locating twitter users,' in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 759–768.

[10] B. Hecht, L. Hong, B. Suh and E. H. Chi, 'Tweets from justin bieber's heart: The dynamics of the location field in user profiles,' in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2011, pp. 237–246.

[11] K. Ryoo and S. Moon, 'Inferring twitter user locations with 10 km accuracy,' in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 643–648.

[12] R. Li, S. Wang, H. Deng, R. Wang and K. C.-C. Chang, 'Towards social user profiling: Unified and discriminative influence model for inferring home locations,' in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1023–1031.

[13] R. Li, S. Wang and K. C.-C. Chang, 'Multiple location profiling for users and relationships from social network and content,' *arXiv preprint arXiv:1208.0288*, 2012.

[14] J. Jia and N. Z. Gong, 'Attriguard: A practical defense against attribute inference attacks via adversarial machine learning,' in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 513–529.

[15] A. Chaabane, G. Acs, M. A. Kaafar *et al.*, 'You are what you like! information leakage through users' interests,' in *Proceedings of the 19th annual network & distributed system security symposium (NDSS)*, Citeseer, 2012.

[16] N. Z. Gong and B. Liu, 'You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors,' in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 979–995.

[17] J. Jia, B. Wang, L. Zhang and N. Z. Gong, 'Attriinfer: Inferring user attributes in online social networks using markov random fields,' in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 1561–1569.

[18] C. Salter, O. S. Saydjari, B. Schneier and J. Wallner, 'Toward a secure system engineering methodolgy,' in *Proceedings of the 1998 workshop on New security paradigms*, 1998, pp. 2–10.

[19] C. A. Gomez-Uribe and N. Hunt, 'The netflix recommender system: Algorithms, business value, and innovation,' *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, 2015.

[20] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, 'Dive into deep learning,' *arXiv preprint arXiv:2106.11342*, 2021.

[21] F. O. Isinkaye, Y. Folajimi and B. A. Ojokoh, 'Recommendation systems: Principles, methods and evaluation,' *Egyptian informatics journal*, vol. 16, no. 3, pp. 261–273, 2015.

[22] B. Smith and G. Linden, 'Two decades of recommender systems at amazon. com,' *Ieee internet computing*, vol. 21, no. 3, pp. 12–18, 2017.

[23] G. Adomavicius, R. Sankaranarayanan, S. Sen and A. Tuzhilin, 'Incorporating contextual information in recommender systems using a multidimensional approach,' *ACM Transactions on Information systems (TOIS)*, vol. 23, no. 1, pp. 103–145, 2005.

[24] H. J. Lee and S. J. Park, 'Moners: A news recommender for the mobile web,' *Expert Systems with Applications*, vol. 32, no. 1, pp. 143–150, 2007.

[25] L. Baltrunas, B. Ludwig, S. Peer and F. Ricci, 'Context-aware places of interest recommendations for mobile users,' in *International Conference of Design, User Experience, and Usability*, Springer, 2011, pp. 531–540.

[26] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama and G. Karypis, 'When being weak is brave: Privacy in recommender systems,' *arXiv preprint cs/0105028*, 2001.

[27] B. P. Knijnenburg and A. Kobsa, 'Making decisions about privacy: Information disclosure in context-aware recommender systems,' *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 3, no. 3, pp. 1–23, 2013.

[28] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten and V. Shmatikov, '" you might also like:" privacy risks of collaborative filtering,' in *2011 IEEE symposium on security and privacy*, IEEE, 2011, pp. 231–246.

[29] A. Pfitzmann and M. Hansen, *A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management*, 2010.

[30] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft and D. Boneh, 'Privacy-preserving matrix factorization,' in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 801–812.

[31] H. Polat and W. Du, 'Privacy-preserving collaborative filtering,' *International journal of electronic commerce*, vol. 9, no. 4, pp. 9–35, 2005.

[32] J. Canny, 'Collaborative filtering with privacy,' in *Proceedings 2002 IEEE Symposium on Security and Privacy*, IEEE, 2002, pp. 45–57.

[33] Z. Erkin, M. Beye, T. Veugen and R. L. Lagendijk, 'Privacy enhanced recommender system,' in *Thirty-first symposium on information theory in the Benelux*, 2010, pp. 35–42.

[34] U. Weinsberg, S. Bhagat, S. Ioannidis and N. Taft, 'Blurme: Inferring and obfuscating user gender based on ratings,' in *Proceedings of the sixth ACM conference on Recommender systems*, 2012, pp. 195–202.

[35] C. Strucks, M. Slokom and M. Larson, 'Blurm (or) e: Revisiting gender obfuscation in the user-item matrix,' 2019.

[36] A. Berlioz, A. Friedman, M. A. Kaafar, R. Boreli and S. Berkovsky, 'Applying differential privacy to matrix factorization,' in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015, pp. 107–114.

[37] Z. Liu, Y.-X. Wang and A. Smola, 'Fast differentially private matrix factorization,' in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015, pp. 171–178.

[38] D. Xu, S. Yuan, X. Wu and H. Phan, 'Dpne: Differentially private network embedding,' in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2018, pp. 235–246.

[39] D. Riboni and C. Bettini, 'Private context-aware recommendation of points of interest: An initial investigation,' in *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, IEEE, 2012, pp. 584–589.

[40] F. McSherry and I. Mironov, 'Differentially private recommender systems: Building privacy into the netflix prize contenders,' in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 627–636.

[41] A. Machanavajjhala, A. Korolova and A. D. Sarma, 'Personalized social recommendations-accurate or private?' *arXiv preprint arXiv:1105.4254*, 2011.

[42] V. W. Anelli, A. Bellogıén, Y. Deldjoo, T. Di Noia and F. A. Merra, 'Multi-step adversarial perturbations on recommender systems embeddings,' *arXiv preprint arXiv:2010.01329*, 2020.

[43] S. Rendle, 'Factorization machines,' in *2010 IEEE International conference on data mining*, IEEE, 2010, pp. 995–1000.

[44] W. S. Noble, 'What is a support vector machine?' *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[45] E. Lundquist. (2020). 'Factorization machines for item recommendation with implicit feedback data,' [Online]. Available: `https://towardsdatascience.com/factorization-machines-for-item-recommendation-with-implicit-feedback-data-5655a7c749db` (visited on 18/07/2020).

[46] A. Mnih and R. R. Salakhutdinov, 'Probabilistic matrix factorization,' in *Advances in neural information processing systems*, 2008, pp. 1257–1264.

[47] S. Sun and R. Huang, 'An adaptive k-nearest neighbor algorithm,' in *2010 seventh international conference on fuzzy systems and knowledge discovery*, IEEE, vol. 1, 2010, pp. 91–94.

[48] H. Li, 'A short introduction to learning to rank,' *IEICE TRANSACTIONS on Information and Systems*, vol. 94, no. 10, pp. 1854–1862, 2011.

[49] S. Rendle, C. Freudenthaler, Z. Gantner and L. Schmidt-Thieme, 'Bpr: Bayesian personalized ranking from implicit feedback,' *arXiv preprint arXiv:1205.2618*, 2012.

[50] L. Breiman, 'Random forests,' *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[51] R. Polikar, 'Ensemble learning,' in *Ensemble machine learning*, Springer, 2012, pp. 1–34.

[52] Y.-Y. Song and L. Ying, 'Decision tree methods: Applications for classification and prediction,' *Shanghai archives of psychiatry*, vol. 27, no. 2, p. 130, 2015.

[53] J. Friedman, T. Hastie, R. Tibshirani *et al.*, *The elements of statistical learning*, 10. Springer series in statistics New York, 2001, vol. 1.

[54] T. K. Ho, 'The random subspace method for constructing decision forests,' *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[55] Y. Amit and D. Geman, 'Shape quantization and recognition with randomized trees,' *Neural computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[56] L. Breiman, 'Bagging predictors,' *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[57] G. M. James, *Majority vote classifiers: theory and applications*. Stanford University, 1998.

[58] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein and M. Klein, *Logistic regression*. Springer, 2002.

[59] D. W. Hosmer Jr, S. Lemeshow and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[60] C. Zhu, R. H. Byrd, P. Lu and J. Nocedal, 'Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization,' *ACM Transactions on mathematical software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[61] S. Suthaharan, 'Support vector machine,' in *Machine learning models and algorithms for big data classification*, Springer, 2016, pp. 207–235.

[62] D. A. Pisner and D. M. Schnyer, 'Support vector machine,' in *Machine Learning*, Elsevier, 2020, pp. 101–121.

[63] S. S. Keerthi and C.-J. Lin, 'Asymptotic behaviors of support vector machines with gaussian kernel,' *Neural computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[64] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, 'The weka data mining software: An update,' *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[65] Z. Meng, R. McCreadie, C. Macdonald and I. Ounis, 'Exploring data splitting strategies for the evaluation of recommendation models,' in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 681–686.

[66] D. Liang, R. G. Krishnan, M. D. Hoffman and T. Jebara, 'Variational autoencoders for collaborative filtering,' in *Proceedings of the 2018 world wide web conference*, 2018, pp. 689–698.

[67] Z. Meng, R. McCreadie, C. Macdonald and I. Ounis, 'Variational bayesian context-aware representation for grocery recommendation,' *arXiv preprint arXiv:1909.07705*, 2019.

[68] M. Kuhn and K. Johnson, *Feature engineering and selection: A practical approach for predictive models*. CRC Press, 2019.

[69] M. Kuhn, K. Johnson *et al.*, *Applied predictive modeling*. Springer, 2013, vol. 26.

[70] G. C. Cawley and N. L. Talbot, 'On over-fitting in model selection and subsequent selection bias in performance evaluation,' *The Journal of Machine Learning Research*, vol. 11, pp. 2079–2107, 2010.

[71] C.-N. Ziegler, S. M. McNee, J. A. Konstan and G. Lausen, 'Improving recommendation lists through topic diversification,' in *Proceedings of the 14th international conference on World Wide Web*, 2005, pp. 22–32.

[72] T. Fawcett, 'An introduction to roc analysis,' *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[73] B. Wellman and M. Gulia, 'Virtual communities as communities,' *Communities in cyberspace*, pp. 167–194, 1999.

[74] M. Van Meeteren, A. Poorthuis and E. Dugundji, 'Mapping communities in large virtual social networks: Using twitter data to find the indie mac community,' in *2010 IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA)*, IEEE, 2010, pp. 1–8.

[75] J. McGee, J. Caverlee and Z. Cheng, 'Location prediction in social media based on tie strength,' in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 459–468.

[76] C. A. Davis Jr, G. L. Pappa, D. R. R. De Oliveira and F. de L. Arcanjo, 'Inferring the location of twitter messages based on user relationships,' *Transactions in GIS*, vol. 15, no. 6, pp. 735–751, 2011.

[77] M. Humbert, T. Studer, M. Grossglauser and J.-P. Hubaux, 'Nowhere to hide: Navigating around privacy in online social networks,' in *European Symposium on Research in Computer Security*, Springer, 2013, pp. 682–699.

[78] S. Labitzke, F. Werling, J. Mittag and H. Hartenstein, 'Do online social network friends still threaten my privacy?' In *Proceedings of the third ACM conference on Data and application security and privacy*, 2013, pp. 13–24.

[79] D. Jurgens, T. Finethy, J. McCorriston, Y. Xu and D. Ruths, 'Geolocation prediction in twitter using social networks: A critical analysis and review of current practice,' in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 9, 2015.

[80] R. Shokri, 'Quantifying and protecting location privacy,' *it-Information Technology*, vol. 57, no. 4, pp. 257–263, 2015.

[81] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec and J.-P. Hubaux, 'Quantifying location privacy,' in *2011 IEEE symposium on security and privacy*, IEEE, 2011, pp. 247–262.

[82] R. Shokri, G. Theodorakopoulos, G. Danezis, J.-P. Hubaux and J.-Y. Le Boudec, 'Quantifying location privacy: The case of sporadic location exposure,' in *International Symposium on Privacy Enhancing Technologies Symposium*, Springer, 2011, pp. 57–76.

[83] N. Z. Gong and B. Liu, 'Attribute inference attacks in online social networks,' *ACM Transactions on Privacy and Security (TOPS)*, vol. 21, no. 1, pp. 1–30, 2018.

[84] D. Jurgens, 'That's what friends are for: Inferring location in online social media platforms based on social relationships,' in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 7, 2013.

[85] R. Compton, D. Jurgens and D. Allen, 'Geotagging one hundred million twitter accounts with total variation minimization,' in *2014 IEEE international conference on Big data (big data)*, IEEE, 2014, pp. 393–401.

[86] J. Eisenstein, B. O'Connor, N. A. Smith and E. Xing, 'A latent variable model for geographic lexical variation,' in *Proceedings of the 2010 conference on empirical methods in natural language processing*, 2010, pp. 1277–1287.

[87] A. Ahmed, L. Hong and A. J. Smola, 'Hierarchical geographical modeling of user locations from social media posts,' in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 25–36.

[88] B. Han, P. Cook and T. Baldwin, 'Geolocation prediction in social media data by finding location indicative words,' in *Proceedings of COLING 2012*, 2012, pp. 1045–1062.

[89] B. Wing and J. Baldridge, 'Hierarchical discriminative classification for text-based geolocation,' in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 336–348.

[90] H. Tang, X. Zhao and Y. Ren, 'A multilayer recognition model for twitter user geolocation,' *Wireless Networks*, pp. 1–6, 2019.

[91] A. Rahimi, T. Cohn and T. Baldwin, 'Twitter user geolocation using a unified text and network prediction model,' *arXiv preprint arXiv:1506.08259*, 2015.

[92] A. Rahimi, T. Cohn and T. Baldwin, 'A neural model for user geolocation and lexical dialectology,' *arXiv preprint arXiv:1704.04008*, 2017.

[93] A. Rahimi, T. Cohn and T. Baldwin, 'Semi-supervised user geolocation via graph convolutional networks,' *arXiv preprint arXiv:1804.08049*, 2018.

[94] A. Rahimi, D. Vu, T. Cohn and T. Baldwin, 'Exploiting text and network context for geolocation of social media users,' *arXiv preprint arXiv:1506.04803*, 2015.

[95] P. P. Talukdar and K. Crammer, 'New regularized algorithms for transductive learning,' in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2009, pp. 442–457.

[96] H. Tian, M. Zhang, X. Luo, F. Liu and Y. Qiao, 'Twitter user location inference based on representation learning and label propagation,' in *Proceedings of The Web Conference 2020*, 2020, pp. 2648–2654.

[97] M. Kosinski, D. Stillwell and T. Graepel, 'Private traits and attributes are predictable from digital records of human behavior,' *Proceedings of the national academy of sciences*, vol. 110, no. 15, pp. 5802–5805, 2013.

[98] Y. S. Resheff, Y. Elazar, M. Shahar and O. S. Shalom, 'Privacy and fairness in recommender systems via adversarial training of user representations,' *arXiv preprint arXiv:1807.03521*, 2018.

[99] T. La Fond and J. Neville, 'Randomization tests for distinguishing social influence and homophily effects,' in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 601–610.

[100] J. He, W. W. Chu and Z. V. Liu, 'Inferring privacy information from social networks,' in *International Conference on Intelligence and Security Informatics*, Springer, 2006, pp. 154–165.

[101] J. Lindamood, R. Heatherly, M. Kantarcioglu and B. Thuraisingham, 'Inferring private information using social network data,' in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 1145–1146.

[102] R. Dey, C. Tang, K. Ross and N. Saxena, 'Estimating age privacy leakage in online social networks,' in *2012 proceedings ieee infocom*, IEEE, 2012, pp. 2836–2840.

[103] K. Thomas, C. Grier and D. M. Nicol, 'Unfriendly: Multi-party privacy risks in social networks,' in *International Symposium on Privacy Enhancing Technologies Symposium*, Springer, 2010, pp. 236–252.

[104] C. Jernigan and B. F. Mistree, 'Gaydar: Facebook friendships expose sexual orientation,' *First Monday*, 2009.

[105] N. Z. Gong, A. Talwalkar, L. Mackey, L. Huang, E. C. R. Shin, E. Stefanov, E. Shi and D. Song, 'Joint link prediction and attribute inference using a social-attribute network,' *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 2, pp. 1–20, 2014.

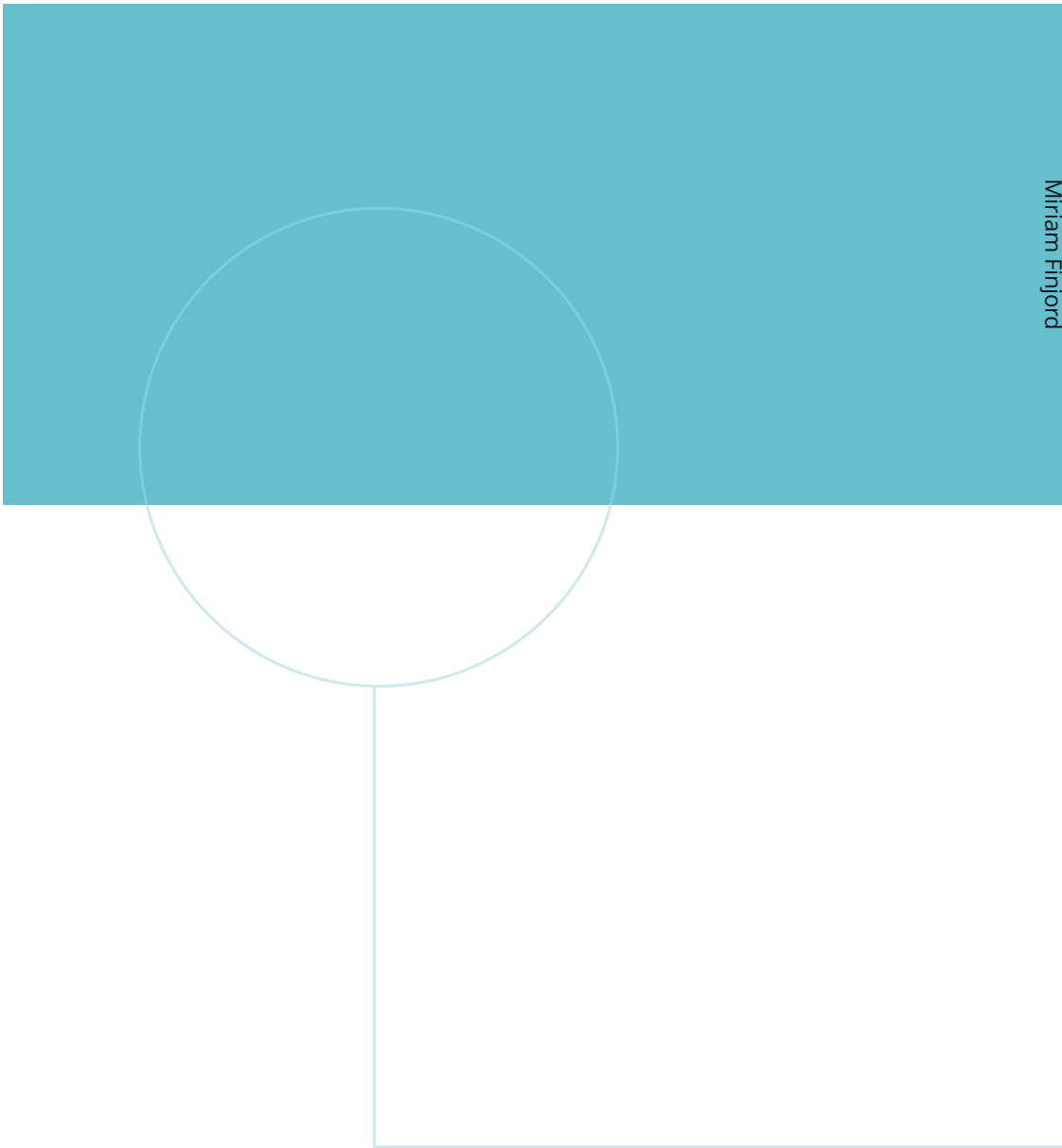
[106] E. Zheleva and L. Getoor, 'To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles,' in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 531–540.

[107] A. Mislove, B. Viswanath, K. P. Gummadi and P. Druschel, 'You are who you know: Inferring user profiles in online social networks,' in *Proceedings of the third ACM international conference on Web search and data mining*, 2010, pp. 251–260.

[108] F. M. Harper and J. A. Konstan, 'The movielens datasets: History and context,' *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[109] D. Hovy and A. Søgaard, 'Tagging performance correlates with author age,' in *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: Short papers)*, 2015, pp. 483–488.

[110] A. Likas, N. Vlassis and J. J. Verbeek, 'The global k-means clustering algorithm,' *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.



Miriam Finjord

NTNU
Norwegian University of
Science and Technology