Profile Injection Attack Detection in Recommender System

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Software Engineering

Submitted By

Ashish Kumar

(Roll No. 801331004)

Under the supervision of

Dr. Deepak Garg

Associate Professor and Head

(CSED)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Profile Injection Attack Detection in Recommender System*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala; is an authentic record of my own work carried out under the supervision of *Dr. Deepak Garg* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Ashish Kumar)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Deepak Garg)
Associate Professor
Thapar University
Patiala

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. S. Bhatia)
Dean (Academic Affairs)
Thapar University
Patiala

i

ACKNOWLEDGMENT

First of all, I am extremely thankful to my respective guide ***Dr. Deepak Garg*** Associate professor, CSED, Thapar University for his valuable guidance, advice, motivation, encouragement, moral support, sincere effort and positive attitude with which he solved my queries and provide delightful ambiance for learning, exploring and making this thesis possible. He always set high goals for me and help me to find the right direction to achieve those goals. It has been a great experience to work under his guidance.

I am also heartily thankful to ***Dr. Prashant Singh Rana***, Assistant Professor, CSED, Thapar University for motivation and guidance that was very helpful to achieve my goals.

I would like to thank my family members and my friends who are dearest and precious to me for their love, encouragement, blessings and support in all respects. Most importantly, none of this would have been possible without the love and patience of my family. To my brother and friends for showing me right direction. They are constant source of love, concern, support and strength for me all these years.

Finally I would like to thank management of Thapar University for proving a great platform for learning, not just for academics but also for sports and many other creative things.

Ashish Kumar

801331004

ME (SE)

ABSTRACT

Recommender systems are backbone for ecommerce website today. It is not restricted to ecommerce websites; social networking websites are also highly dependent on recommender systems. These are using recommender systems for providing personalized services to their users or customers.

Recommender systems are desired to attain a high level of accuracy while making the predictions which are relevant to the customer, as it becomes a very tedious task to explore the thousands of relevant items from the huge database. Different ecommerce companies use different types of recommender systems based on their requirement. But a threat comes in frame in last one decade i.e. profile injection in the system to affect the accuracy of the collaborative recommender system. These attacks degrade the accuracy of recommendations.

In this thesis, we focused on the behavior of attacks in the system. When attacks are inserted some change comes in the recommendations, we measure these changes in the form of prediction shift. This prediction shift may be positive or negative. In the next phase we focused on the detection of these attacks so that these attacks no longer affect the accuracy of the system. For this purpose, we use machine learning models and detection attributes. In the supervised profile classification approach, we compare six models and compare their accuracy. Although, none of the model gives 100% accuracy but some models gives good accuracy. We pick best performer models and proposed our ensemble model by using voting technique of ensembling. Although our proposed model doesn’t give best accuracy but it will never give worst accuracy in any case. Next we give a comparative study of unsupervised models.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This chapter introduces recommender system, its history, business aspects of a recommender system, and different types of ratings used by different service providers.

1.1 Recommender System and its History

From the beginning of web services the data on the web has been increasing at rapid rate. Retrieval of useful information is an important issue. A recommender system is a system that suggests items to the user in which he might be interested. It is often essential for peoples to consult with others and ask about their past experiences when making choices [1].This suggestion could be of any type like which movie to watch, what item to buy or which book to read for machine learning. To recommend an item to user a recommender system focuses on the similarity of the user with other users in the system and description of item. A recommender system is very beneficial for a user who has lack of knowledge for finding appropriate item on the web. In the e-commerce context the role of recommender system is same as the sales man in retail showroom.

A recommender system can be personalized or non-personalized. In personalized recommender system different users receive diverse suggestions according to the users interaction with the system whereas a non-personalized recommender system are often very easy to understand and usually they include top 10 books or CDs etc. But these non-personalized recommender system are not focused by researchers that much.

A personalized recommender system offers a list of items based on their ranking. To calculate this ranking recommender system predicts the most suitable item based on user's preferences. Recommender system can also consider the navigation of user to a particular web page and items shown on that page plays role to create a list of recommended items.

As e-commerce web sites began to develop, it become very important to filter the whole range of alternate items. But it is very difficult for users to find most appropriate choice

from the immerse variety of items that the website is offering. Although large number of options available for users, but because of no clear information, user often take poor decision. And in this case instead of giving benefit to user, it started to decrease the user's well beings.

Recommender system is a result of very simple observation that individuals often relies on recommendations provided by others in making daily decisions [3]. For example when selecting a movie to watch, user tends to rely on the movie review from the critics.

Before coming the recommendation system in existence, at that time search engines were not personalized and they do not take in to the account the special need of each individual user and consider only the keywords for performing search queries. To solve this problem first recommendation system was introduced by Goldberg and his team in 1992 [2].

Amazon.com is a well-known e-commerce website which uses recommender system. In real time it shows the most popular items to the user. It performs some mathematical computation as well as performs some business rules on the opinions collected from different users. Business rules can be defined as items belonging to specific category like electronics, kitchen etc. available items in the stock or finding the worthwhile product to display on their front page. When a customer searches an item, it performs some type of sorting and searching to find out the right product to be displayed on the top of the list. It may be the non-personalized recommender system that uses the ratings given by different users. Usually item having rating or rated by large number of users, get place in top of list. When a user selects an item from this list, it also shows first next recommender like "frequently bought together", it is called product association recommender. Based on the other user's interaction with the system it also shows "customer who bought this also bought this" or it shows "customer who viewed this item also viewed these items", but these items are not purchased by others. But people look at these items before purchasing a particular item. Even without selecting or searching an item, it shows the list of items based on the search or purchase history of user.

Figure 1.1: Recommendations example for online store.

1.2 Business Aspects of a Recommender System

A recommender system plays an important role for both the service providers and for users. Functions of a recommender system for service provider are [3]:

i. **Increase user satisfaction:** to improve the experience of user with the system it is important that recommender system is well designed. Easy to find relevant item, easy to use and accuracy of system increases the user's satisfaction.
ii. **Increase the sell:** the very first task of any online retailer is to sell maximum number of items. Here recommender system play very important role. Its work is not to show those items that the user only know, its work is to recommend those item which the user want to buy but user is not able to find those items on immense data. It also recommends additional items to the user that he may buy.
iii. **Unique and personalized service to the customer:** a good recommender system gathers user's information and rather than recommending him top rated items, it will recommend his items of his own interests. Every user will get list of different recommending items.

A recommender system is valuable not only for service providers. It is equally valuable for users also. It helps user to find right item in real time. Functions of a recommender system for users are [3]:

i. **Find items of interest:** it finds items based on their rankings, and then predicts items to a particular user's interest. Without the help of recommender it would be very difficult to search for some items from thousands of available items. It not only filters out items but also shows a list in sequence or in the form package, in which a user may buy those items.
ii. **Narrow down the set of choices:** sometimes user got confused which item to buy from the large number of available options. At this point recommender system helps to narrow down the choices and recommend only most suited items to the user.
iii. **Discover new things:** recommender system also helps to users to find new best alternatives available. Many times it happens that user is not aware of alternative or better items for a particular need. Here recommender system help user to discover new interesting items.
iv. **Just exploring items:** many times user just wants to know new items, then instead of showing irrelevant items to user, recommender system show those items in which user has interest, so that his browsing can be converted in purchase.

1.3 Types of Ratings

For the working of recommender systems most important is ratings. Ratings collection can be done by either implicitly or explicitly. In implicit form of rating collection, user interacts with system with some keywords. For example if user search "machine learning" on amazon then it will give a list of books on machine learning to user. When user click on any of the books for more detail, at this stage system assume that this particular user is interested in this book. On the other hand in explicit collection of rating system ask user to give ratings, these ratings are of different types:

i. **Ordinal rating**: e.g. "strongly disagree, disagree, neutral, agree, strongly agree", user can choose any of these based on his opinion.
ii. **Numerical rating:** e.g. 1-5 stars, numbers of stars indicated how much you agree with the item.
iii. **Binary rating:** e.g. thumb up or thumb down symbol, thumb up indicates liked and thumb down indicates disliked.

CHAPTER 2

RECOMMENDER SYSTEMS

This chapter introduces various types of recommender systems, how to calculate recommendations, advantages and disadvantages of different- different recommender systems.

In last two decades several different types of recommendation systems have been introduced. Difference between different types of recommendation system is how they predict the best item to the customer. Some recommendation systems are using the properties of items to predict the recommended item where some other recommender systems are using the customer's interaction or past behavior. Over the time recommendation system is getting smarter and smarter.

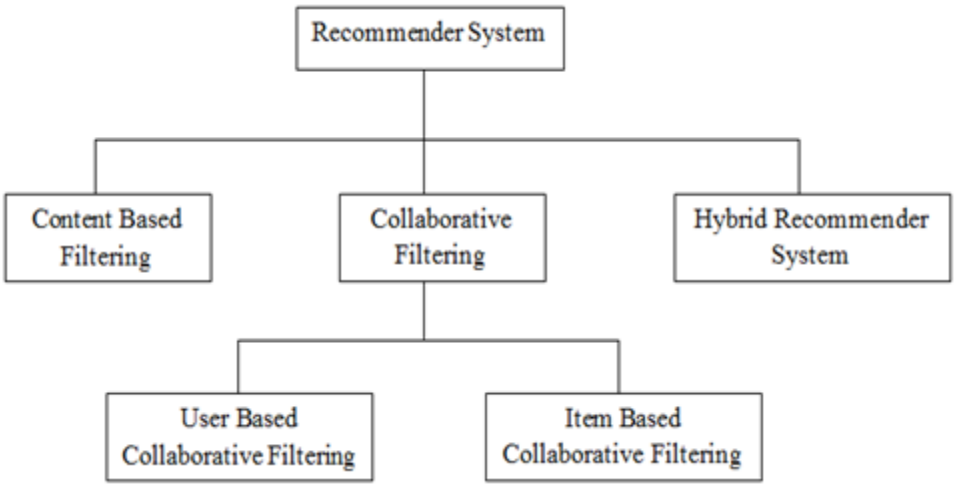

Figure 2.1: Types of Recommender System

2.1 Content Based Recommender System

The biggest challenge of the internet is that information available on the web is dynamic and heterogeneous in nature. It is very difficult for a user to find the accurate information from such a huge data that meets his requirement. Nobody is interested in spending hours on web just to search small information. It is the responsibility of web service provider that user gets right information in less time. So it is important to personalize the

information access according to the user's interests and tastes. Content based recommender system use the user's interests or tastes as input and generate a list of recommended items as output. At amazon.com, recommendation algorithm personalizes their store for each individual customer separately [5].

Content based recommender system is mainly used for recommending text based products such as web pages, news, messages etc. from which you can find a textual description. Each item to be recommended is described by some of their associated features, often these feature are called keywords. Some of keyword examples are shown in Table 2.1.

Table 2.1: An example of keywords used in content based RS.

Category	**Keywords**			
News	Research	Movie review	Cricket	New mobile
Clothing	Cotton	Blue	Low-price	Casual
Movie	Aamir	Anuskha Sharma	Raj Kumar Hirani	Comedy

Content based recommender systems recommend items to a user based on these keywords and user's interest. Content based filtering does not rely on social information in the form of other user's rating [4]. In this recommender system, we need some information /keywords about the set of available items and another thing that we need is user profiles describing what the user likes or dislikes. Now the recommender system learns the user's preference and recommends items that are similar to user preference.

In recommendation process attributes of items are compared against the user profile, and item that shows maximum similarity or preference with the user profile will be shown on the top of recommended items. The purpose of this comparison is to filter out items that are not relevant to user. Basic architecture of content based recommender system is shown in Figure 2.2.

The personal information (personal description, ratings, posts etc.) posted by users on social networking websites like Facebook, LinkedIn can be exploited by a recommender

system. [7] Defines recommender system as software that uses the interest or preferences of an individual user for products and makes recommendations accordingly.

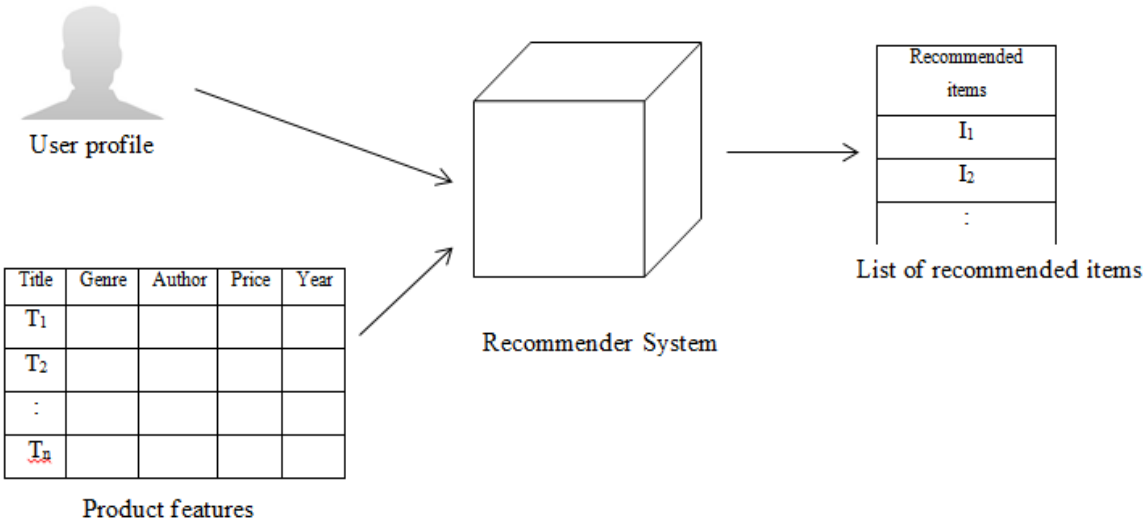

Figure 2.2: Architecture of content based RS.

[6] Presents an online social network based recommender system that extracts user's interest for job and then make recommendation to them accordingly. He uses the data set extracted from work4. Efficiency of a recommender system depends on how users and items have been represented in the system.

[1] Uses tag based recommender system for social book marking websites. Tagging activities in social bookmarking sites are used to identify user's need and hence to personalize recommendation. Tags assigned to a web page are highly related to the content of web page. Architecture of his proposed recommender system consist three steps: user modeling, web page modeling, and then finally present new web pages to each user based on the user profile and web page features.

[9] Presents a recommender system for music data that combines two methodologies, content based filtering techniques and then interactive genetic algorithms. He use feature extraction tool i.e. CALM to analyze the properties of items. As the interest of a user may change, to handle these changing interests he combines filtering technique with genetic algorithm.

In content based recommender system, algorithm extract the textual description from the unstructured web pages, news articles etc. the extraction of features is very difficult task because algorithm searches for words or strings in text, if exact word or string matched, it will pick that word but it will not figure out synonyms. Semantic analysis and its integration in personalization model is one of the most innovative approach proposed in literature to solve this problem.

2.2 Collaborative Filtering

It is a process of filtering information using large datasets to create a pattern that can predict the future interest of a specific user. It works on the assumption that user who had similar taste in the past will have similar taste in future also.

Collaborative filtering approach overcomes the limitations of the content based recommender system. Suppose items for which the content is not easily available or difficult to obtain can still be recommended to a user based on the feedback received from other users. Unlike content based recommender system, a collaborative filter can recommend items with very different content or keywords that are not used normally as long as other users have already shown interest for these different items.

In the early 1990s, collaborative filtering came into picture as a solution for dealing with the overload in online information space. Tabestry [2] was a manual collaborative filtering system which was used by users to query an item such as corporate mails based on other users action. It was an experimental mail system developed at the Xerox Palo Alto research center to handle huge amount of incoming documents via electronic emails.

Basically collaborative recommender system is a two steps process. At the first step it computes the similarity between the targeted user for which we want to find recommended item and all other users in the system. In the second step it develops the prediction for the targeted user based in the first step [8].

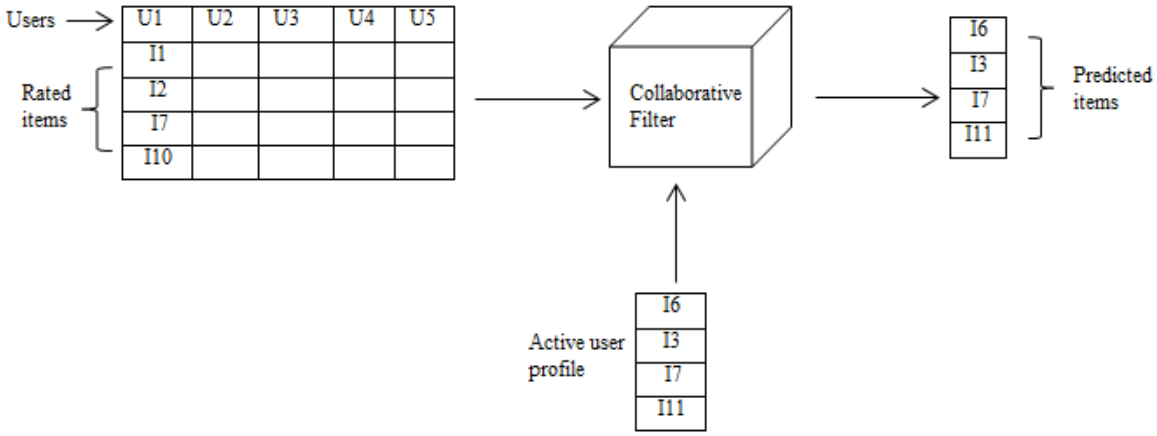

Figure 2.3: Basic structure of collaborative filtering.

2.2.1 User-User Collaborative Filtering

It is also popularly known as kNN (k nearest neighbors) collaborative filtering. It was the first of the automated collaborative method. User-user collaborative filtering us a straightforward approach based on the concept of concept of collaborative filtering. Many user based systems such as GroupLens [11], Ringo [10] and BellCore video [13] evaluate the interest of a user u for an item i using the ratings for this item by other users called neighbors that have similar rating pattern. GroupLens [12] used this approach to identify usenet articles that are likely to be interesting to a particular user. Users are required to provide ratings and system combines these ratings with the ratings provided by other users to provide personalized results. And users need not know the opinion of the other users.

2.2.1.1 Steps to Perform User-User Collaborative Filtering

i. **Calculating similarity:** To compute the similarity between users, several methods have been proposed and evaluated in the literature. Some of these are as follows:

   a) **Pearson correlation:** it can be calculated by Equation 2.1.

$$S_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})}{\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r_u})^2}\sqrt{\sum_{i \in I}(r_{v,i} - \overline{r_v})^2}} \quad (2.1)$$

   Where $S_{u,v}$ is the similarity between user u and v, $u, v \in U$. $i \in I$ is the subset of items rated by both users u and v. $r_{u,i}$ Denotes the rating given by the user u to

item i and $\overline{r_u}$ is the average of all ratings given by user u [14]. Both GroupLens and BellCore used Pearson correlation to compute the similarity in their projects [12, 13].

Constrained Pearson Correlation: it is a variation of Pearson correlation that uses the midpoint instead of mean rate. On a rating scale of 5, rating 3 can be fixed as neutral provided that user neither like nor dislike item. It was proposed by Shardanand and Maes [16] and it is given by Equation 2.2.

$$S_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - r_z)(r_{v,i} - r_z)}{\sqrt{\sum_{i \in I}(r_{u,i} - r_z)^2}\sqrt{\sum_{i \in I}(r_{v,i} - r_z)^2}} \quad (2.2)$$

r_z is the mid neutral rating.

b) Cosine Similarity: It is a vector space approach based on the linear algebra. It is given by Equation 2.3.

$$S_{u,v} = \frac{\sum_{i \in I} r_{u,i} \cdot r_{v,i}}{\sqrt{\sum_{i \in I} {r_{u,i}}^2 \cdot {r_{u,i}}^2}} \quad (2.3)$$

Basically we have two techniques for selecting neighbors. First one is threshold based selection in which we choose users whose similarity exceeds a certain threshold value. Second technique is kNN in which k is the input provided to the system and we select top k nearest neighbors [15].

ii. Computing Prediction

Similarity $S_{u,v}$ that we have computed by Equations 2.1, 2.2 and 2.3 is used to generate predictions. By picking the item with the highest predictions, recommendations are generated. It can be computed by Equation 2.4.

$$P_{u,i} = \frac{\sum_{n \in Neighbors}(r_{n,i} - \overline{r_n}) S_{u,n}}{\sum_{n \in Neighbors} |S_{u,n}|} + \overline{r_n} \quad (2.4)$$

2.2.1.2 Problems of User-User based CF

i. Issue of sparsity: Typically a data set has large number of users and items. But rarely it happens that every item get good number of ratings, generally a user gives ratings to only a few numbers of items and only popular items get ratings and many items remain unrated. Rarely there is situation where no recommendation can be made for

an item [RS-15]. Many solutions have been proposed for this issue such as dimensionality reduction and item-item based collaborative filtering.

ii. **Computational performance:** Computing all pair of correlation is expensive. In worst case it takes $O(MN)$ where M is the number of users and N is the number of items in the product catalog, since it examine M customer for N items each. As the average customer vector is sparse so in worst case performance of algorithm will be close to $O(M+N)$ because every customer contains a small number of items [5]. Second factor that affect performance is user profile could change quickly. It is needed to compute correlation and make recommendation in real time to keep user happy.

iii. **Scalability:** collaborative algorithms require large computation to perform and this computation increases as the number of users and items increase. With millions of users and items as most of the ecommerce portals has that much, a typical recommender system running traditional algorithm suffer serious scalability problem.

2.2.2 Item-Item based Collaborative Filtering

To address the scalability issue of user-user based recommendation algorithm, model based also popularly known as item-item based recommendation technique have been developed in [18, 19]. It is based on the principle that if two items have same users like or dislike then those set of items are considered to be similar. It was first time described by Sarwar et al. [17] and Karypis [20].

Consider a user-item matrix in which rows indicates users and columns indicates items. User-user based collaborative filtering considers the relationship between rows whereas item based collaborative filtering considers relationship between columns.

Item-item based collaborative filtering is dependent on number of users in the system, if the number of users is fairly large as compare to number of items, in that case item-item recommendation technique is very efficient. In this technique of recommendation instead of matching the users to the other similar users, it matches each of the users purchase and rated item to the other similar items and then combine all these similar items to make a list of recommended items [22].

Amazon's [5] algorithms, item-item collaborative filtering is capable to scale a big massive data set and produce good quality recommendation in real time. It builds a similar item table by finding items that customer tends to purchase together. This approach takes less memory as well as processing time as compared to item-item table. Its computation is very fast because it depends only on the number of items user purchased or rated.

2.2.2.1 Steps to Perform Item-Item Collaborative Filtering

Calculation of item-item based recommendation is also two-step process, in the first step we compute the similarity between items and in the second step we perform prediction.

i. **Item similarity computation:** item-item similarity is fairly stable as average items have more ratings as compared to the average users and items do not generally changes rapidly. Different methods for computing the similarity between items are as follows:

   a) **Cosine based similarity:** in this case *n* items are considered as *n* vectors in *m* dimensional user space and similarity is measured by computing the cosine of angle between them. Similarity $S_{i,j}$ between two items i and j is given by Equation 2.5.

$$S_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|^2 * \|\vec{j}\|^2} \tag{2.5}$$

   "." Denotes the dot product of two vectors.

   b) **Pearson correlation based similarity:** similarity between item i and item j is computed by using Pearson correlation. For this purpose we first find the set of user who has rated both the items. Let U is the set of users who has rated both items i and j and this can be computed by given Equation 2.6.

$$S_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r_i})(r_{u,j} - \bar{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r_i})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r_j})^2}} \tag{2.6}$$

   Adjusted correlation based similarity: in this method of similarity computation instead of subtracting the average rating of item, we subtract average rating of user. It is given in the Equation 2.7.

$$S_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_u})(r_{u,j} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{r_u})^2}} \quad (2.7)$$

ii. **Prediction calculation:** similarity $S_{i,j}$ that we compute in the previous section is used for computing the rating prediction of target user. It is given by Equation 2.8.

$$P_{u,i} = \frac{\sum_{j \in S} S_{i,j} r_{u,j}}{\sum_{j \in S} |S_{i,j}|} \quad (2.8)$$

Problem with this equation is that similarity may become negative, it will bias the predicted values so that they no longer map to user ratings domain. Solution to this problem is that a threshold limit should be set to consider only non-negative similarities. So modified equation is given by Equation 2.9.

$$P_{u,i} = \frac{\sum_{j \in S}(S_{i,j} - b_{u,i})}{\sum_{j \in S} |S_{i,j}|} + b_{u,i} \quad (2.9)$$

2.3 Hybrid Recommender System

Many times it happens that one type of recommender system do not perform well in all the situations, to get performance sometimes it is required to combine several different algorithms to form a hybrid recommender system. For example collaborative filtering suffers from the problem cold start when user has not rated to any item or no rating to a specific item. But content based recommender system does not suffer with this problem. So a better approach is not to use these recommendation techniques separately but use a combination of several techniques.

A hybrid system could combine content based and collaborative filtering approach. It uses the description of items to find the its similarity with the existing items in the system based on their description and allow this new item to be recommended. Once the user give rating to an item then involvement of collaborative filtering increases.

In most of the hybrid recommender system content based approach is combined with collaborative filtering approach. Burke [21] divides the hybrid recommender system into seven class and give a thorough analysis:

i. **Weighted recommender system:** in this type of recommender system, score of a predicted item is computed from all of the available recommendation techniques applied in the system. Claypool et al. [23] combine content based and collaborative filtering approach in online newspaper.

ii. **Switching:** in this type, recommender switches between different recommendation algorithm and use the algorithms that is assumed to have the best result on a particular situation.

iii. **Mixed:** in this type of hybrid recommender system, several recommendation systems combine to form a list of recommended item. This approach avoids the new item start up problem.

iv. **Feature combination:** it uses the features from multiple recommendation system as an input to a single recommendation algorithm. The feature combination technique lets the system consider collaborative data without relying on it, hence it reduces the sensitivity of the system to the number of users who have rated an item.

v. **Cascade:** this method involves staged process. In this technique, first a recommendation technique is applied to produce a recommendation list and in the second stage another recommendation technique refine the recommendation from the previous recommended list.

vi. **Feature augmenting recommender:** in this method output of one algorithm is used as input feature for another. For example, one technique produces classification of an item and that information is then incorporate into the processing of next recommendation technique.

vii. **Meta level recommender:** in this approach, one algorithm train a model and then use that model as an input to another algorithm.

CHAPTER 3

DATA MINING TECHNIQUES

This chapter gives an overview of some popularly used data mining techniques used in the context of recommender system. We first describe common preprocessing methods such as dimensionality reduction and sampling. After this, we review some classification and clustering techniques.

Data mining process mainly consists of 3 steps: data preprocessing, data analysis and data interpretation. The Figure 3.1 shows basic flow of data for data mining.

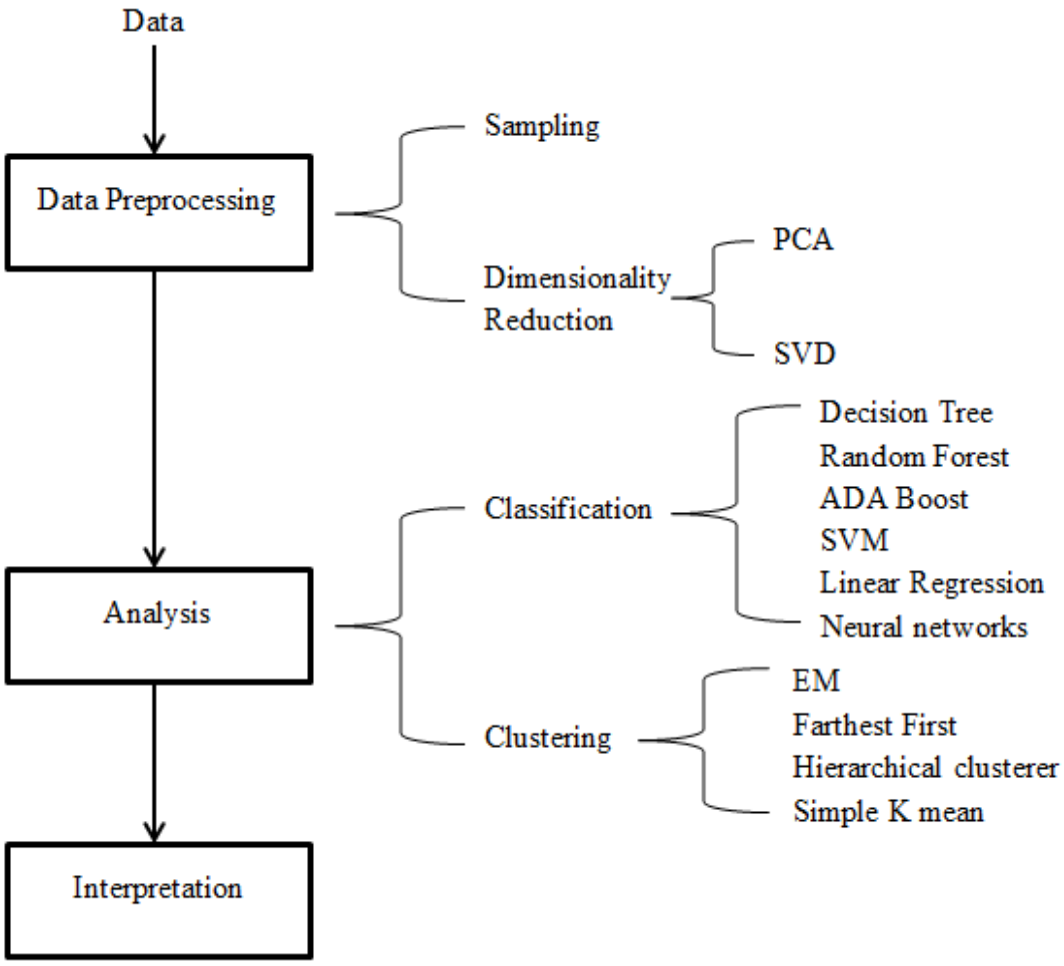

Figure 3.1: Steps and methods in data mining.

Although the number of algorithms available for the above steps are more than listed above. In this chapter we give brief introduction of these algorithms.

3.1 Data Preprocessing

Before analyzing data we need to process (clean, filter and transform) it. Some of the preprocessing techniques are described below.

3.1.1 Sampling

Often we have very large datasets and processing such a huge data is very expensive. A subset of relevant data is selected from a very big dataset is done with the help of sampling. Sampling can be done with several techniques; the most common sampling technique is Random sampling, in which every item has equal probability of getting selected. Another popular sampling technique is stratified sampling, in which data is split into partitions based on some features. These partitions can be used to create training and test datasets. Training dataset is used to learn the models and test dataset is used to evaluate the model to make sure it performs well.

Sampling has a problem that it can lead to over specialization to the specific division of the training and testing datasets. To solve this problem, training process can be repeated multiple times. From the original dataset, training and test sets are created after this model is trained used training data and tested on the test set. This process is repeated *K* times by selecting different training and test set every time. Finally average performance of *K* learned models is noted down. This process is called cross-validation. In *n*-Fold cross validation, the data set is divided into *n* folds. One of the folds is used for testing the model and remaining *n-1* folds are used for training [24].

3.1.2 Dimensionality Reduction

In recommender systems, we use huge data sets that define a huge dimensional space but very sparse information in that space. Suppose a data set have a list of 100 movies and 1000 people, for each people we know whether he like or dislike each of 100 movies. So for each person we have a binary vector of length 100. Performing on such a big vector is time consuming and situation can become worse if we have bigger data set. Dimensionality reduction technique solves this problem by transforming original high

dimensional space into a lower dimensionality. So rather than performing on these vectors of length 100 directly, we divide movies in 5 genres. Now figure out whether a person like or dislike the entire genre. In this way we reduce the data from a vector of size 100 to a vector of size 5.

Two main dimensionality reduction techniques are Principal Component Analysis (PCA) and Singular Value Decomposition (SVD).

3.1.2.1 Principal Component Analysis

It is a classical statistical method for finding the pattern in high dimensional data set [25]. In PCA, a data set might have dozens of independent factors and all of these needs to be plotted on independent axis. All the information is plotted in two or three axis. First principal component capture more variance than the second principal component and so on. Dimensionality can be reduced by neglecting those components with small contribution to the variance. First principal component have maximum amount of the information than the second principal component and so on. If first principal component is storing 80.5% of information than removing second component imply losing only 19.5% of the information. Although PCA is a powerful technique for dimensionality reduction but it has some limitations also. PCA relies on linear data although for nonlinear data it has been proposed. Another assumption of PCA is that the original data set has been drawn from Gaussian distribution. An approach to use PCA in the context of online joke recommender system has been proposed by Goldbreg et al. [26].

3.1.2.2 Singular value decomposition (SVD)

SVD [27] is one of the powerful techniques for dimensionality reduction. It is related to PCA because of realization of the Matrix Factorization approach. The main task in SVD decomposition is to find a lower dimensional feature space where the new features represent “concepts” and the strength of each concept is computable [51]. It can be used as the basis of latent-semantic analysis; it is very powerful technique for text classification in information retrieval. Sarwar et al. [28] describe two different ways to use SVD to improve collaborative filtering. In the first way, SVD can be used to uncover

relation between customers and products. To achieve this target, they first fill the zeros in the user-item matrix with the item average rating and then normalize by subtracting the user average. This matrix is then factored using SVD and the resulting decomposition can be used directly to compute the predictions. Another approach is to use low dimensional space resulting from the SVD to improve the neighborhood formation for later use in a kNN approach. Serwar et al. [30] describe the biggest advantage of SVD is that to compute an approximated decomposition there are incremental algorithms. This allows accepting new users or ratings without having to recompute the model that is already have been built with the existing data. Brand [29] extended the same idea for the online SVD model.

3.2 Analysis

After preprocessing, we need to analyze the data to get the useful information. In this section we give brief overview of classification and clustering approach.

3.2.1 Classification

It does the work of mapping between a feature space and a label space; features represent characteristics of the elements to classify and labels represent their classes. For example, a restaurant recommender system can be implemented by a classifier that classifies restaurants into one of two categories either good or bad based on the number of features that describe it. Classifiers are generally of two types supervised and unsupervised (clustering). In supervised classification, a set of categories is known in advance and we have a set of labeled data which consists a training set. In this section we will discuss supervised learning algorithms.

3.2.1.1 Decision Tree

The observation which we want to classify is composed of attributes and target values. Decision trees [31] are the classifiers on a target class in the form of a tree structure. A decision tree contains two types of nodes; first one is decision node, at this node single attribute value is tested to determine to which branch particular sub tree applies. Another type of node is leaf nodes which indicate the value of the target attribute. CART, C4.5,

ID3, Hunts Algorithm and SLIQ are the most common decision tree algorithms. Hunt algorithm is one of the most earliest and easiest to understand. When all the observations belong to some class then decision tree induction stop. It ensures that the impurity of the leaf nodes is zero. Pruning is used by most of the decision trees by which a node is no further split if number of observations in a node are below a certain threshold. Advantage of tree classifier is that it is inexpensive to construct and it is very fast in classifying unknown instances. They can be used to produce a set of rules that are easy to interpret while maintaining accuracy comparable to other classifying techniques. It is very difficult to build a decision tree that tries to explain all the variables involved in the decision making process. A RS for online purchases that combines Association Rules and Decision Trees presented by Cho et al. [32]. Decision tree is used as a filter to select which set of users should be targeted with recommendations. Decision trees are also used as tool for item ranking in RS.

3.2.1.2 Artificial Neural Networks (ANN)

It [33] is an assembly of inter connected nodes and weighted links that is inspired in the architecture of biological brain. Nodes are called neurons. These are composed into networks, after training with sufficient data; they have ability to learn a classification problem.

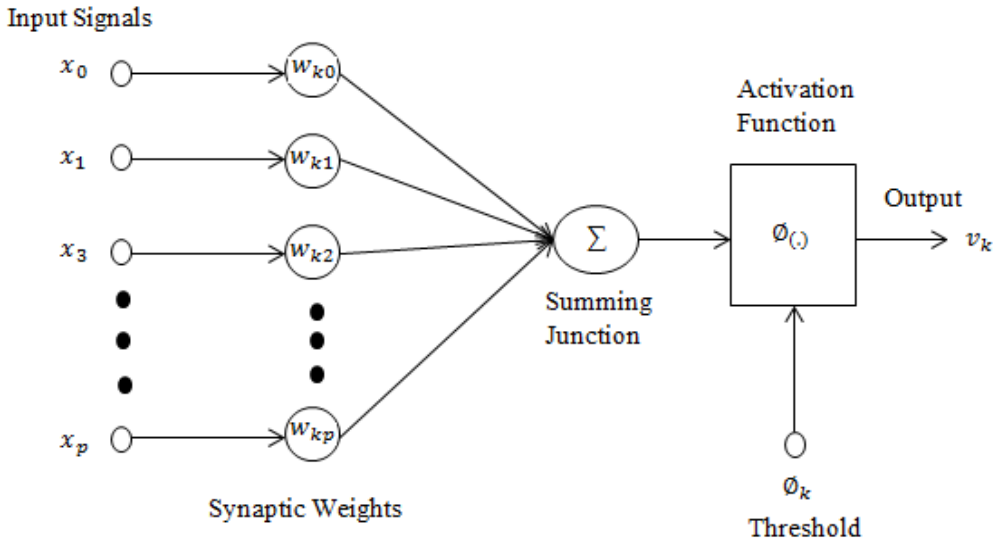

Figure 3.2: Perceptron model.

One of the most simplest case of ANN is Perceptron model shown in Figure 3.2. Output of ANN can be obtained by adding each of its input values with the weights of its links and by comparing its output against some threshold value. The perceptron model is a simple and efficient linear learning classifier. One of the main advantage of ANN is that depending on the activation function it can also perform non linear tasks. But the disadvantage of ANN is that it is hard to come up with the ideal network topology for a given problem. Inputs from several data sources can be combined in ANN. Hsu et al. [34] uses data from different sources: user profiles, viewing communities, program metadata and viewing context to build a TV recommender system. A hybrid content based collaborative recommender system was built by Christakou and Stafylopatis [35]. The content based recommender system is implemented using three neural networks per user, each of them corresponding to one of the following features: "kinds", "stars" and "synopsis". They trained the ANN using the Resilient Backpropagation method.

3.2.1.3 Support Vector Machine (SVM)

In support vector machine classifier [36], a linear hyperplane or decision boundary is to be find that separates the data in such a way that the margin is maximized. For a two class separation problem in a two dimesion space their can be many possible boundary lines to separate the two classes as shown in Figure 3.3. The main motive behind the SVM is that we want to maximize the margin so that we are less likely to missclassify the items in the future.

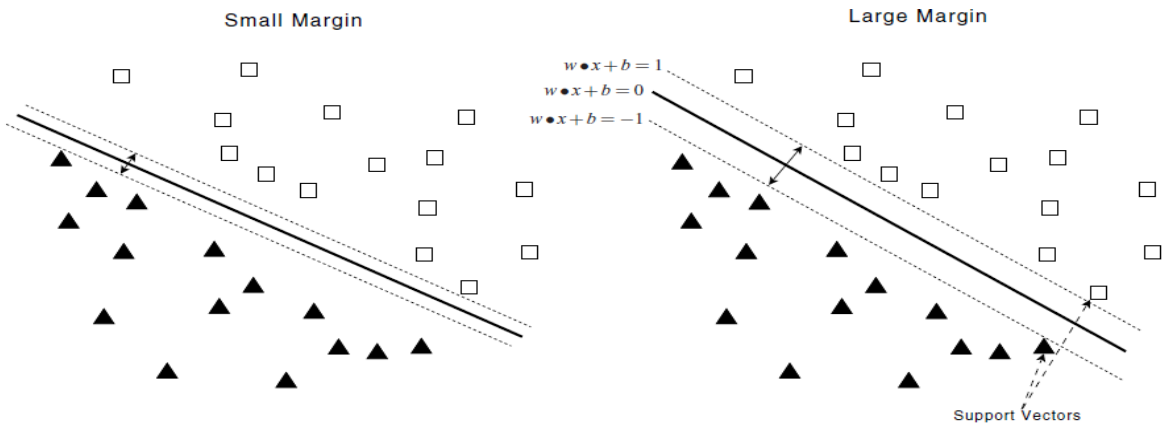

Figure 3.3: Boundary decisions to separate two classes.

Soft margin classifier is used when the items are not linearly separable. On the other hand, we need to transform the data into higher dimension space if the decision boundary is not linear. Because of its performance and efficiency recently it has gained popularity. It has shown promising results in RS. A study has been conducted by Kang and Yoo [38] that aims at selecting the best preprocessing technique for predicting missing values for an SVM based RS. A TV program RS was build using SVM by Xu and Araki [39]. For features they use information from Electronic Program Guide (EPG). In order to reduce the features they removed words based on lowest frequency. Oku et al. [37] build context-aware RS and they use context-aware vector machines (C-SVM). They compare the use of C-SVM, standard SVM and an extension that uses CF as well as C-SVM. For Restaurant recommendations, they show the effectiveness of the context aware methods.

3.2.1.4 Random Forest

It is an ensemble algorithm for regression, classification. This algorithm combines the idea of bagging and feature selection. It constructs multiple trees at training time and output the class that is the mode of classification or regression of individual tree. It correct the over fitting problem of decision tree. Random forest is best option to use when user doesn't know which model to use or when user need to produce a decent model under time pressure [43]. Each tree gives a classification and Random forest chooses the classification having most votes. Increasing the correlation between two trees of forest increase the forest error rate. A tree with low error rate is a strong classifier. Strength of individual tree can be increased by decreasing the forest error rate. This algorithm can be applied efficiently on large data sets. It automatically detects what variables are important in the classification. It automatically estimates the missing values and maintains a very good accuracy even when large values are missing. Generated forests on one data can be saved and can be used in future on other data sets. Capacity of Random forest can be extended unlabeled data, leading to unsupervised clustering.

3.2.1.5 ADA Boost

It is a type of adaptive boosting a machine learning algorithm. It was formulated by Yoav Freund and Robert Schapire in 2003. To improve the performance of different algorithms

it can be combined with them. The output of weak classifier is combined with weighted sum that represents the final output of the classifier. ADA boost is very sensitive to noisy data and outliers. But disadvantage of this boosting algorithm is that in some problems it faces the problem of over fitting than other classifier algorithms. But the advantage of this classifier is that it is very easy to implement. ADA boost is also known as best out of box classifier. When this algorithm is used with decision tree learning, information gathered at each stage of the ADA boost algorithm about the relative hardness of each training sample is fed into the tree growing algorithm such that later tree tend to focus on harder to classify examples [44]. It focuses on difficult data points that are misclassified in previous weak classifier.

3.2.1.6 Linear Regression

It is an approach for modeling the relationship between dependent variable y and independent variable$x$. If number of independent variable is one then it is called simple linear regression and if numbers of independent variables are more than one, in this case it is called multiple linear regressions. This model is widely used in practical application. Reason for this is that the model which is linearly dependent on unknown parameters are easier to fit than models which are non-linearly related to their parameters. In linear model we are interested to find out that one variable is varying as a straight line function of another variable. The amount of variability of a variable is variance, which is defined as its average squared deviation from its mean. This variability is measured in terms of its standard deviation and it is defined as the square root of the variance. So in linear models for prediction, our interest is to find mean, variance of each variable and the correlation coefficient between each pair of variables.

3.2.2 Clustering

It is an activity of grouping a set of items in such a way that same group will contain items that are most similar to each other than to those in other groups. It is an important task of data mining and of other statistical data analysis that are used in many areas including machine learning, image analysis, pattern recognition and information retrieval. Generally accuracy of unsupervised machine learning models is less as compared to the

supervised machine learning models. Many techniques are used for clustering, in this section we give a brief overview of four techniques which are used by us in the implementation.

3.2.2.1 Expectation-Maximization

It is as iterative method and it is basically used for finding the maximum likelihood of parameters in statistical models. It consist of two steps and these steps are performed alternatively in iteration first step is calculating the expectation (E) of log- likelihood and second step is maximization (M) which computes parameters maximizing the expected log- likelihood found in the expectation [45]. The key feature of this clustering algorithm that make it efficient is its ability to optimize a large number of variables, its ability to find good estimates for even missing information in dataset. It is very good algorithm for finding the likelihood of problems that cannot be solved directly. Finding a maximum likelihood solution require taking a directives of the likelihood function with respect to all unknown values i.e. parameters and the latent variables. In statistical models, result is usually a set of interrelated equations in which solution to the parameter requires the value of the variable and vice-versa by substituting one set of equation into other produces an unsolved equation. In EM algorithm, one can simply pick one arbitrary values for one of the two sets of unknown and use them to estimate the other and then use these new values to find better estimate of the first and keep this thing alternating between the two until the two values reaches to a fixed point. In general there may be multiple maxima and there is no guarantee by EM algorithm that local maxima can be found [46]. EM is widely used for data clustering in machine learning and computer vision. With its ability to deal with the missing values, EM has become a useful tool to price and manage risk of portfolio.

3.2.2.2 Simple K Means

It is one of the simplest unsupervised learning algorithms. It works on a simple principal that it divides the *n* observations in to *k* clusters such that each observation belongs to the cluster with the nearest mean. Main idea behind this clustering method is to define *k* centers one for each cluster. These centers are placed as far as possible so that better

results can be obtained. *K*- Means was originated from signal processing and it even today also used in this domain. Although *k* mean is widely used clustering method, but pure *k*- means algorithm is limited in term of flexibility as a result of limited use. Second issue in simple *k* means is to choose right value of *k* and it cannot be used with arbitrary distance functions or on non-numerical data [47]. Simple *k* means after combining with simple linear regression for supervised learning in NLP and in computer vision. Advantages of simple *k* means are that it is fast, robust and easier to understand. It also gives best results when data set are distinct or well separated from each other. The main disadvantages of this method are that we need to know how many number of cluster centers required also these clusters need to be exclusive i.e. clusters should not overlap each other. This clustering method is unable to handle noisy, categorical and non-linear data set.

3.2.2.3 Hierarchical Clustering

In data mining, hierarchical clustering is a method of cluster analysis in which we build a hierarchy of clusters. Hierarchical clustering approach has been divided into two types: first is Agglomerative, it is a bottom up approach; in it pairs of clusters are merged as one move up in the hierarchy. Second type is Divisive, it is top down approach; in it all observations start in one cluster and split down recursively as one move down in hierarchy [48].

3.2.2.4 Farthest First

It is variant of simple k means that places each cluster center in turn at the point farthest from the existing cluster centers. This center must lie within the data area. This algorithm speed up the clustering in most of the cases since less reassignment and adjustment is needed [49]. To each cluster and attribute it fits a discrete distribution.

CHAPTER 4

ATTACKS ON RECOMMENDER SYSTEM

In social recommendation behavior, people share their with each other and decide whether or not to act on the basis of what they hear from others [Herlocker et al. 2006]. Collaborative filtering recommender system is an electronic version of social recommendation. One basic difference between collaborative recommender system and social recommender system is that in collaborative recommender system interactions can be scaled to group of thousands and people do not know each other. But in every day social recommender system giver of recommendations has a known identity on which a receiver can rely and interaction is limited to small number of people.

Results of a collaborative recommender system highly dependent on the honest feedback of its users. Recommendations are calculated based on the genuine feedback will be of good quality. As in collaborative recommender system identities of other users is not known, so it is possible to introduce the fake user profiles identical to the genuine users present in the system. In this way attacker make the system produce recommendation behavior that he desires. For these type of vulnerabilities shilling attack term is used [40, 41]. Recent studies show that the behavior of most of the recommendation algorithms can be manipulated by even modest attacks [42].

Collaborative filtering systems are the most common type of web personalization system and are highly vulnerable to attacks. Attackers inject a large number of biased profiles into the system, resulting in system recommendation that favor or disfavor a given item. Profile injection attack term is often used for such type of attacks. Promoting and demoting a particular item is the only reason for using such attacks. In the profile injection attacks, attacker with aim to bias the system's results interacts with the collaborative recommender system to build a large number of fake profiles identical with the original users in the system [42].

Injecting fake profiles in the system can be done by many ways, attacker can insert profiles either manually or he can develop an automated tool to insert profiles. If the system is small in term of users and ratings they provide then manually insertion

technique is also efficient. Because manually inserting profiles in the system take lots of effort and time. In small system even small number of fake profiles can promote and demote a particular item. But if our system is large containing thousands of users and billions ratings, in this condition small number of fake users do not cause much effect in the system. And if attacker tries to insert large number of profiles manually then it will not be an optimal solution as it will take lots of effort and time. In this scenario, he must adopt the automation tools to insert the biased profiles in the system resulting in recommendations that favors or disfavors a given item. A collaborative recommendation system must be open to user inputs, so it is very difficult to design a collaborative recommender system that cannot be attacked. So researchers are mainly focused on the defending against such attacks.

4.1 Attack Dimensions

An attack in a collaborative filtering recommender system consists of a set of attack profile, each containing biased rating data associated with a fake user identity. It also includes a target item to be either promoted or demoted. Profile injection attacks can be categorized based on the following four parameters:

i. Knowledge required by the attacker to mount the attack.
ii. Intent of attack.
iii. Profile size.
iv. Size of the attack.

i. **Knowledge required by the attacker to mount the attack:** from the attacker's perspective, the best attack against a system is one that generates maximum impact with the least amount of effort. This effort can be measured in term of knowledge required that an attacker must have to know in order to mount a particular attack. On the basis of knowledge required, attacks can be divided in to two categories: one is high knowledge attack in which attacker must have detailed knowledge of rating distribution in a recommender system database. Some attacks may require mean rating and standard deviation for every item.

ii. **Intent of attack:** basic purpose of an attacker is to either promote or demote a particular item. “Push” and “Nuke” are the terms which are widely used for promoting and demoting a particular item respectively. An attacker may insert biased profiles to make a less likely item to be more likely “push” and a more likely item to be less likely “nuke”.

iii. **Profile size:** it is defined as number of ratings assigned in a given attack profile. From the attackers point of view addition of ratings is relatively lower in cost as compared to the creating of addition profile. Real users rarely rate more than a small fraction of ratable items in a large recommendation space. That is why an attack profile cannot give rating to a large number of items. A attack profile with many ratings can be easily distinguished from those of genuine users and is reasonably certain indicator of an attack.

iv. **Size of attack:** it can be measured by the number of profiles being added to the system by the attacker and the number of ratings supplied by each biased profile. Automating the profile injection process impose less cost to the attacker. And by this approach more number of biased profiles can be injected with least effort or cost. To control the biased profile the site owner can impose the cost of profile creation which requires human intervention. Although this approach cannot stop the injection of fake profiles completely. But definitely it helps to reduce the injection of fake profiles.

Recent work has focused on the techniques that can be used to protect the predictive integrity of collaborative recommender system from the malicious biasing. There are basically two approaches to do the same. One approach is to increase the robustness of the recommender system and another approach is to detect and removing biased profiles. To increase the robustness of the recommender captachas can be used but this is also not a full proof solution. It will just increase the cost of injecting biased profiles. So researchers are mainly focused on detecting and removing ratings of biased profiles during prediction calculation. Here also we will describe various types of attack model and metrics used for detecting biased profiles and then perform predictions.

An illustrative example:

Table 4.1: An example of push attack favoring the target item 6.

	Item1	Item2	Item3	Item4	Item5	Item6	Correlation with Ashish
Ashish	5	2	3	3		?	
User1	2		4		4	1	-1.00
User2	3	1	3		1	2	0.76
User3	4	2	3	1		1	0.72
User4	3	3	2	1	3	1	0.21
User5		3		1	2		-1.00
User6	4	3		3	3	2	0.94
User7		5		1	5	1	-1.00
Attack 1	5		3		2	5	1.00
Attack 2	5	1	4		2	5	0.89
Attack 3	5	2	2	2		5	0.93
Correlation with item 6	0.85	-0.55	0.00	0.48	-0.59		

Consider a recommender system that identifies movies that user might like to watch using user based collaborative approach. A user profile in the system consists of users' ratings in the scale of 1 to 5 with 1 as the lowest and 5 as the highest rating on the various movies. In the previous visits Ashish has built his profile and he return to the system for new recommendations. Consider our system in Table 4.1. It consists eight genuine users including Ashish. It also contains 3 attack profiles (Attack 1-3) inserted by the attacker. All the attack profiles have given ratings to the item 6. Attack profiles may closely match the profiles of one or more of the existing users.

Assume that our system is using user based collaborative filtering approach. Prediction rating for Ashish on item 6 can be obtained by finding the closest neighbor to Ashish. If our system is not attacked then the most similar user to Ashish, using correlation based similarity would be user 6. In this case prediction associated with item 6 for Ashish would be 2. Or simply we can say that item 6 is will be disliked by Ashish. Now consider the scenario that our system is attacked, now attack 1 profile will be the most similar one to the Ashish and it would yield a predicted rating of 5 for item 6. And it is completely

opposite to what he has predicted without the attack. So we can say that in this example, attack is successful and Ashish will get item 6 as a recommendation, it does not matter whether this is really the best suggestion for him.

Now suppose if our system is using item based collaborative filtering approach. The prediction rating for item 6 would be calculated by comparing the rating vector of item 6 with those of other items. This method does not lend itself to an attack as previous one because attacker does not have control over the other users for their ratings to a particular given item. If attacker obtains some knowledge of rating distributions then this can make a successful attack. In our example of Figure 4.1, attacker knows that item 1 is a popular item among a significant group of users to which Ashish also belongs. Designing an attack profile such that high rating is associated with both items 1 and 6. Attacker can try to increase the similarity of these two items resulting in a higher possibility that Ashish will receive item 6 as a recommendation.

4.2 Attack Models

A profile injection attack against a recommender system consists of a set of profiles injected to the system by the attacker. A general structure of these profiles is shown in Figure 4.1.

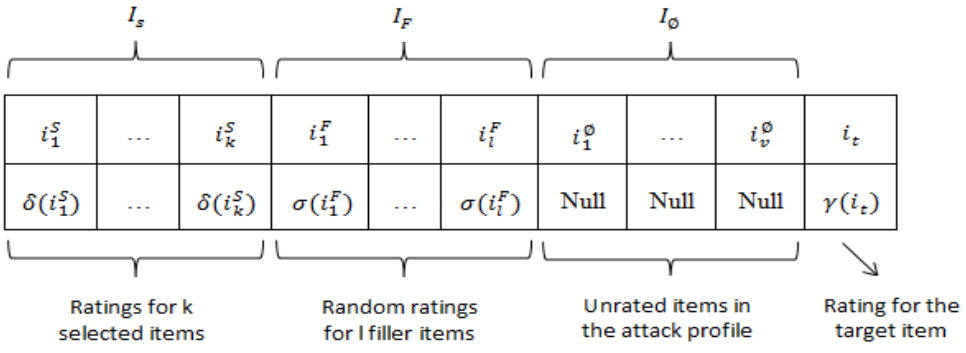

Figure 4.1: General structure of a profile in a profile injection attack.

In the above structure of a profile, items are divided in to four categories and each profile consist each category.

i. **Target item (i_t):** there will be exactly one target item in the items set.

ii. **Selected items (i_s):** it is a set of selected items that represents a small group of items that have been selected because of their association with the target item i_t. For some of attacks this set may be empty. These are generally chosen randomly.

iii. **Filler item (i_F):** the set of filler items represents a group of randomly selected items in the database which are assigned ratings according to the proportion of the attack. Size of the selected item set is small, so size of each profile (total number of ratings) is determined by the size of filler item set.

iv. **Unrated item set ($i_\emptyset$):** this is the set of items that are not rated by attack profile.

$$i_\emptyset = I - (i_s \cup i_F \cup i_t) \quad (4.1)$$

A profile injection attack against a collaborative filtering system consists a large number of attack profiles of same type based on same attack model is added to the database of real user profiles. The motive behind such an attack is to either increase (in case of push attack) or decrease (in case of nuke attack) the system's predicted rating on a target item for a given user.

4.2.1 Push Attack Models

Lam and Riedl [2004] [41] introduced two basic types of attack models i.e. random and average attack models.

4.2.1.1 Random Attack

In the random attack model, average rating of system is assigned to the filler items and pre specified ratings are assigned to the target item e.g. rating 5 is given to the target item in the case of push attack and rating 1 is given to the target item in the case of nuke attack. In the random attack model, set of selected items is empty. Knowledge required for this attack is minimal because the overall mean of the system can be easily measured by outsider. However some cost of execution is involved, but it is not that much because this attack involves assigning mean rating to every filler item. Burke et al. 2005b confirms that the attack is not particularly effective.

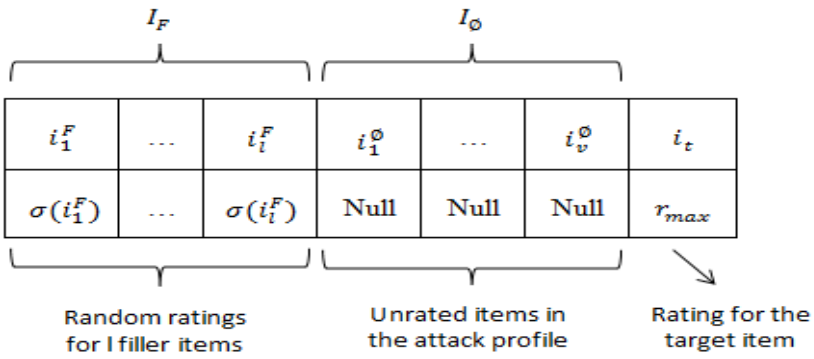

Figure 4.2: An attack profile based on random or average attack model.

4.2.1.2 Average Attack

Lam and Riedl [2004] describes a powerful attack which uses the individual mean for each item rather than the global mean except for the target item. Each filler item is assigned rating i.e. mean rating for that item across the users in the database who have rated it. Similar to random attack, set of filler items is chosen randomly. And set of selected items i.e. I_s can be combined in the set of unrated items i.e. $I_\emptyset$ as I_s is not rated in this attack. This attack can also be used as nuke attack. The basic difference between random and average attack is the way in which ratings are assigned to the filler items in the profile. Figure 4.2 shows the general form of average attack.

Some effort is also involved in this attack for producing the ratings. Average attack has knowledge cost of order $|I_F|$ i.e. number of filler items in the attack profile. Although previous experiments [Burke et al. 2005a] have shown that his attack is successful in user based collaborative technique even when size of filler item set is small. Thus the knowledge cost involved in this attack can be substantially reduced. However this attack model is not effective against item based collaborative algorithms.

4.2.1.3 Bandwagon Attack

This attack can be seen as the extension of random attack, where set of selected items are also included in attack profiles. These selected items are the small number of frequently rated items i.e. items that are popular in the domain. It can be viewed in term of both ratings assigned to them by genuine users and because these items are generally liked. For example these popular items can be a Box-office hit movie or a best-selling book.

These types of items have huge probability of being rated by large number of genuine profiles. In the attack profiles, these selected items are given high ratings to ensure high degree of similarity between attack and genuine profiles.

This attack model is considered to have low knowledge cost. It does not require any system specific data because it is not very difficult to determine the set of most popular items in any item space. Figure 4.3 shows the attack profile for the bandwagon attack. Items set $I_s$is selected because this set of items has been rated by a large number of users in the database and items in the set I_s are given maximum rating values together with the target item i_t. Ratings for the set of items in I_F is determined in the similar manner as in random attack. it has been demonstrated that the bandwagon attack outperforms both random and average attack.

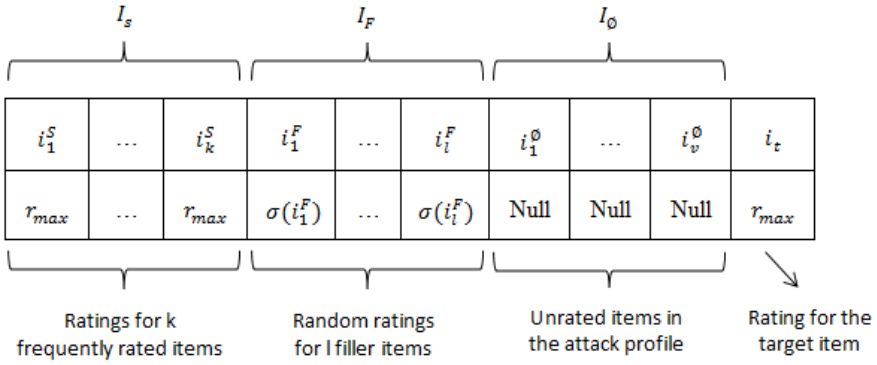

Figure 4.3: A bandwagon attack profile.

4.2.1.4 Segment Attack Model

In the previous work [Lam and Riedl 2004] concluded that user based algorithms were less robust than item based algorithms and previous attacks have been found effective. But from the cost point of view, such attacks are suboptimal and they push items to users who may not be likely purchasers. To address these issues, segment attack model a reduce knowledge push attack specifically for item based algorithms have been introduced [Mobasher et al. 2005].

To increase the impact of promotional activity, it is required to target one's effort on a specific group of users. For example, suppose that Eve had written a romantic book for

youngsters. She would prefer that her book be recommended to buyers who had expressed an interest in this genre, e.g. buyers of Hobbit books rather than buyers of java programming or cooking.

The set of selected items I_s represent the items that are favored by the targeted segment of users and items of this set assigned maximum rating value together with target item. To provide maximum impact on the item based collaborative filtering algorithms minimum ratings are given to the filler items. Previous experimental results show that this attack is quite effective against both item based and user based collaborative filtering algorithms.

4.2.2 Nuke Attack Models

Push attack models that are described above can also be used for nuking a target item. For example, in random and average attack models this can be achieved by associating rating r_{min} with the target item i_t instead of r_{max}. But previous research shows that the attack models that are effective for pushing items are not necessarily effective for nuke attacks. There are two attack models specifically for nuking items and both of these attack models can be considered low knowledge attacks as they do not need any system specific knowledge.

4.2.2.1 Love-Hate Attack

It is a very simple attack, with no knowledge requirement. The attack profile in this attack give minimum rating value r_{min} to the target item i_t, the filler item set is given maximum rating r_{max}. Although a variation of this attack can also be used as push attack by switching the role of r_{min} and r_{max}. Figure 4.4 shows an attack profile structure based on the love-hate attack model. Since knowledge required to mount such an attack is very less, previous researches shoes that this attack is not effective at producing bias recommendations when used as push attack, but it is one of the most effective nuke attack in the user based collaborative filtering algorithms.

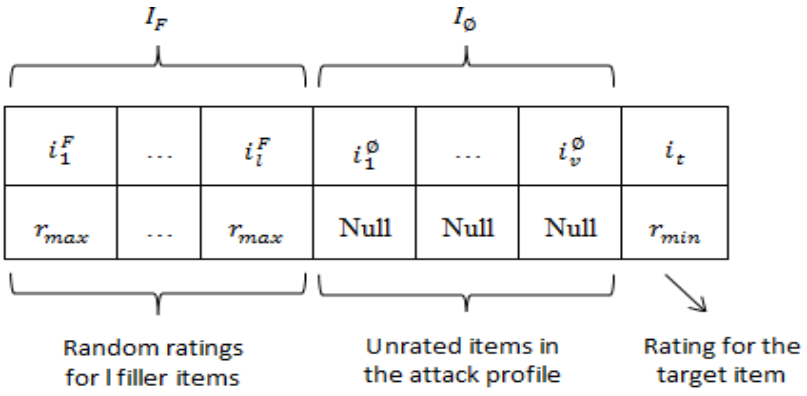

Figure 4.4: An attack profile based on love-hate attack.

4.2.2.2 Reverse Bandwagon Attack

This attack is a variation of the bandwagon attack. In it selected items are are poorly rated by many users. Along with the target item, these items are given minimum rating. Hence target item is associated with widely disliked items, increasing the probability that system will generate low predicted ratings for that item [14]. In this attack, set of selected items is chosen such that items of this set have below average ratings an rating of target item will be r_{min}. This attack is not that much effective as more knowledge intensive attack for nuking user based system. But this attack is very effective nuke attack against item based recommender system. General structure is shown in Figure 4.5.

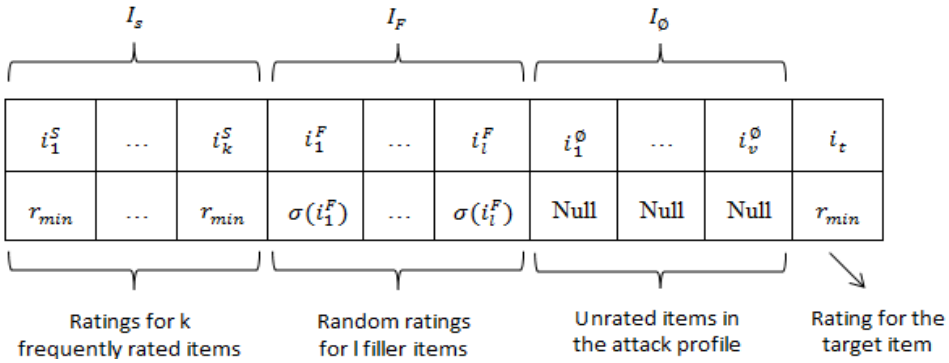

Figure 4.5: An attack profile based on reverse bandwagon attack.

CHAPTER 5

PROBLEM STATEMENT

With the rapid growth in the ecommerce web sites, recommendation systems plays very important role to provide personalized service to the user. A good recommender system determines the quality of service provided by ecommerce. Ecommerce websites like amazon.com and ebay.com are widely popular because of their recommender systems. But there is a problem with the publicly accessible collaborative systems, problem is with their security. Attacker who cannot be separated with distinguished from genuine user may inject biased profiles in the system to affect the service of system. It may leads to degradation of recommender system's objective and accuracy of the system. As previously stated even a small amount of attack can degrade the accuracy of the system.

Collaborative recommender system is highly vulnerable to the attack because it is based on the feedback from user, so an attacker can directly influence the ratings of the system. In particular we are interested in attacks that can be injected with minimum knowledge of the rating distribution. More specifically, we are interested in the following problems:

i. Degree of robustness that recommendation system offers against attacks.
ii. Whether it is possible for an attacker to mount different types of attacks that exploit the weakness of algorithm.
iii. Whether attacks can be detected.

CHAPTER 6

IMPLEMENTATION AND RESULTS

Performance of collaborative recommender systems is dependent on the honest feedback of its users. With the rapid growth of online shopping, news and many more services it become very important that recommender system recommend right item to its users. In last few years lot of research has been carried out in the area of recommender systems. Recommender systems are highly vulnerable to the attacks, in such a case we are interested not only in the performance of recommender system but a0lso change in the performance of system after the attack. Two evaluation measures i.e. robustness and stability were introduced by O'Mahony et al. [2004]. Robustness measure the performance of the system before and after the attack to determine the effect of attack on the completed system whereas stability measure the system's ratings change for the attacked item by the attack profiles.

6.1 Experimental Setup

For our experiments, we use publicly available Movie-Lens 100K dataset. It is widely used dataset for experiments on recommender system. This dataset contains 100,000 ratings by 943 users on 1682 movies. Ratings are given in integer form ranging between 1 to 5 where rating 1 is least (dislike) and 5 are the maximum (most liked) ratings. This dataset contains all the users who have rated to at least 20 movies. Mean rating is 3.6 for all users in this dataset.

To conduct our experiments in section 6.3, we divide our dataset into two parts i.e. training and test set. We kept 70% data for training and 30% data for testing purposes. For all attacks, we generated number of attack profiles and inserted in the system. There are approximately 1000 users in the system, so 10% attack size means that 100 attack profiles added to the system. As there are 1682 unique items in the system, so 1% filler size means one user gives ratings to the 168 items.

6.2 Measuring the Effectiveness of Attacks

Our target is to measure the effectiveness of the attack i.e. we want to measure the win for the attacker. The main motive behind the push attack is that attacker wants to push the item so that item is more likely to be recommended after the attack than before. After the attack there is high possibility that some rating difference will occur in the ratings of target item before and after attack, prediction shift term is widely used for this rating difference. Let U is the set of user and I is the set of items in the system. For each user-item pair (u,i), $\Delta_{u,i}$ denotes the prediction shift.

$$\Delta_{u,i} = p'_{u,i} - p_{u,i} \tag{6.1}$$

Where p' denotes the prediction after the attack and p before the attack. If $\Delta_{u,i}$ is positive that indicates that attack is successful in making the pushed item more positively rated. Average prediction shift for an item i over all users can be computed by using Equation 6.2.

$$\Delta_i = \sum_{u \in U} \Delta_{u,i} / |U| \tag{6.2}$$

In the similar manner average prediction shift $\bar{\Delta}$ for all items in the system can be computed by using Equation 6.3.

$$\bar{\Delta} = \sum_{u \in I} \Delta_i / |I| \tag{6.3}$$

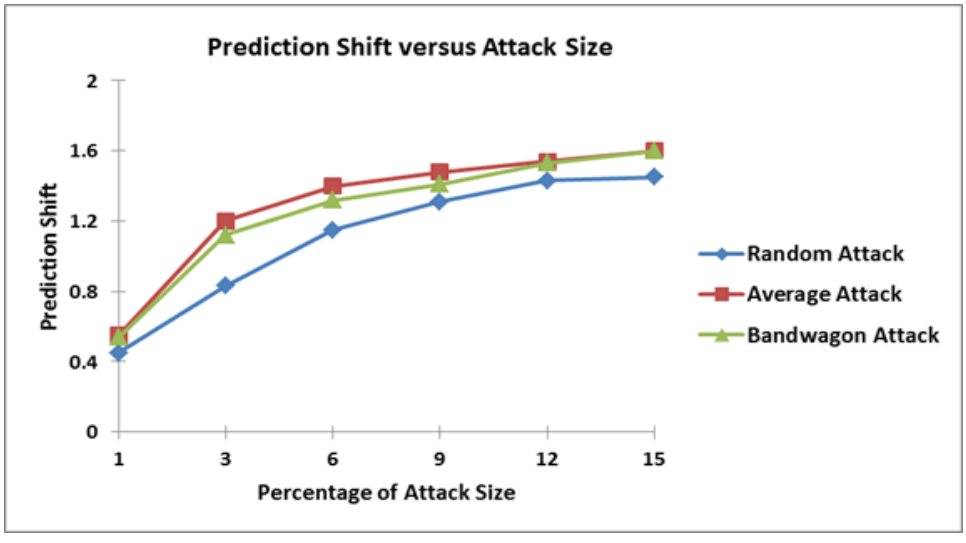

Figure 6.1: Prediction shift versus attack size for random, average and bandwagon attacks.

Average attack is highly successful as shown in Figure 6.1, however substantial knowledge is required for it. To magnify the impact of attacks as shown in above Figure, we kept filler size 5% fixed and changes the attack size. We notice that at low attack size average attack perform very well but as the attack size increases random attack also give better performance. After 12% attack size, bandwagon attack give almost equal performance to average attack.

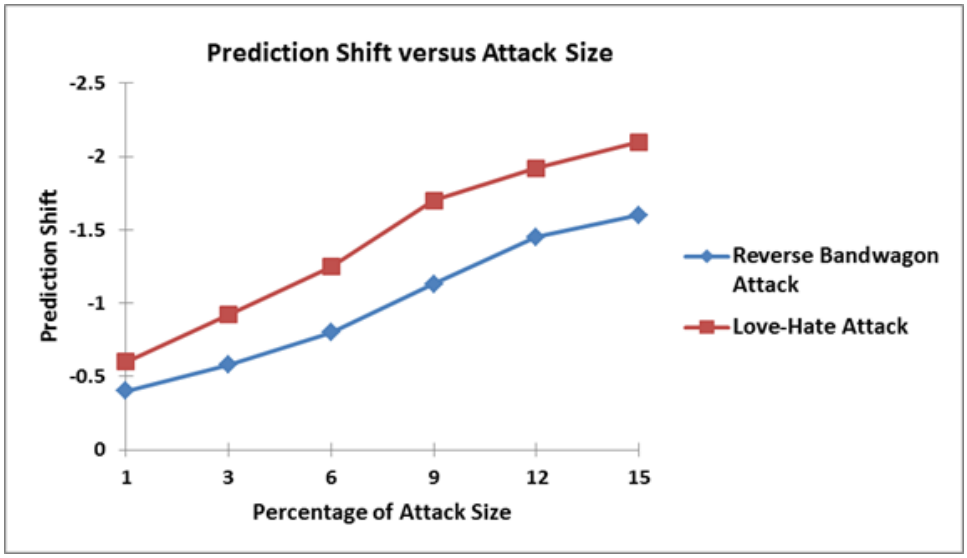

Figure 6.2: Prediction shift versus attack size for reverse bandwagon and Love-Hate attacks.

For nuke attacks, the direction of the impact on the predicted ratings is reverse to the direction in push attacks. Figure 6.2 shown the relationship between prediction shift and attack size of nuke attacks i.e. reverse bandwagon and love-hate attack. We kept filler size fixed at 10% and we vary attack size. We notice that love-hate attack give better performance than to reverse bandwagon attack and reason for the same is that low knowledge is require for love-hate attack.

Figure 6.3 shows the relationship between prediction shift and filler size. We kept attack size fixed at 10% to get better impact and we vary filler size. At all filler size values average attack performs better than all other attacks. But bandwagon attack which is performing better in Figure 6.1 is performing poor in this case. At low filler size random attack also perform poor but as the filler size increases it start performing well and approaches to the performance of average attack at 50% filler size.

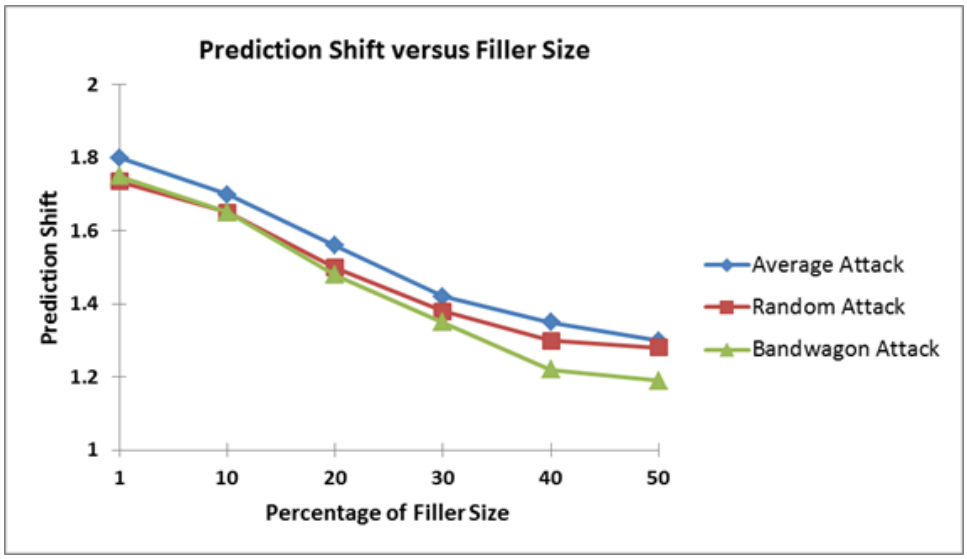

Figure 6.3: Prediction shift versus filler size for average, random and bandwagon attacks.

Figure 6.4 shows the relationship between prediction shift and filler size by keeping attack size fixed at 10% for nuke attacks i.e. reverse bandwagon and love-hate attack. Love-hate attack gives similar performance as it gives in Figure 6.2. At low filler size reverse bandwagon attack also give good performance but as the filler size increases the gap between performances also increase.

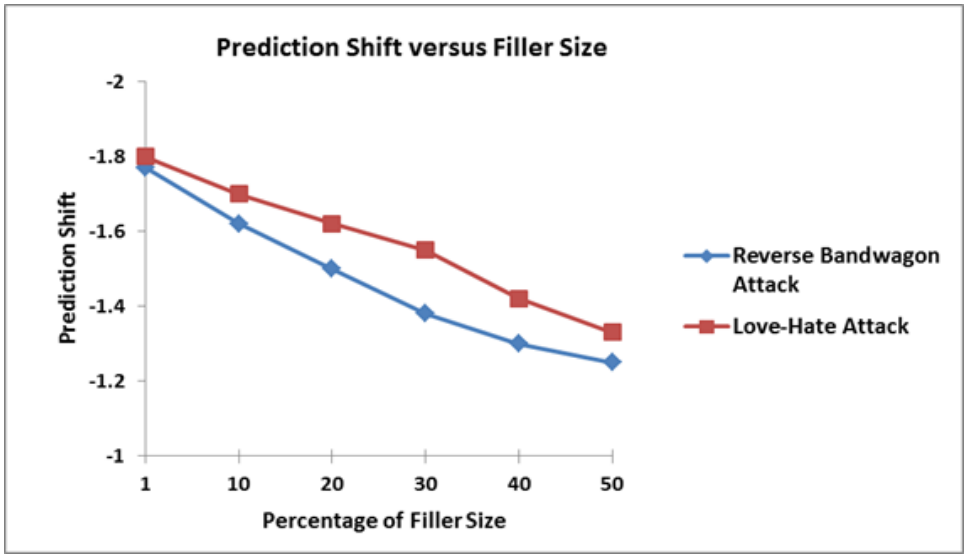

Figure 6.4: Prediction shift versus filler size for reverse bandwagon and Love-Hate attacks.

6.3 Attack Detection Techniques

For detecting and preventing the effect of profile injection attacks, some research has been carried out in last few years. For analyzing the rating patterns of malicious profiles several metrics and algorithms has been designed specifically for the detection of such attack profiles [Chirita et al. 2005]. These metrics were the basis for identifying the basic attack models such as average and random attack. Several new techniques were developed in O'Mahony et al. [2004] to defend against the attacks.

In this section, we focus on the profile classification i.e. genuine or fake user based on the metrics. After identifying suspicious profiles and discounting their contribution towards predictions. This type of approach will be successful based on how clear definition of suspicious profile is. Definition of attack models are used to characterize the model specific and generic attributes corresponding to these attacks. These attributes are used to build classification models for the detection of category of user i.e. genuine or malicious.

6.3.1 Attack Detection Attributes

In our project we use MovieLens data set; this data set contains user ids (UID), item ids (IID) and ratings (R). Original data set is injected with fake profiles according to the attack models. Structure of data set will be similar to shown in Table 6.1.

Table 6.1: Structure of MovieLens data set.

UID	IID	R

To categorize the user profiles we need some properties of users based on which we can tell the type of user i.e. genuine or fake. After finding attribute our data set will contains only user ids and attributes and it will look like as shown in Table 6.2.

Table 6.2: Structure of MovieLens data set after preprocessing.

UID	Attribute *1*	Attribute *2*	Attribute *3*	…	Attribute *n*

The attributes are of two types: generic and model-specific attributes.

6.3.1.1 Generic Attributes for Attack Detection

These attributes are based on simple phenomena that overall signature of attack profiles will be different from the original profiles. This difference comes from the rating behavior of user i.e. rating given to the target item and ratings distribution of ratings among the filler items. Many researchers [Lam and Riedl 2004; Chirita et al. 2005] believe that it is not possible for the attacker to gain the complete knowledge of the ratings in a real system. Result of this is that new profiles are likely to deviate from rating patterns seen for authentic user.

First time Chirita et al. [2005] proposed some generic attributes for the detection of profiles. Some of these attributes are as follows:

i. **Rating Deviation from Mean Agreement (RDMA)** [Chirita et al. 2005] is used to identify the attacker's profiles from a pool of large number of user profiles by examining the profile's average deviation per item, weighted by the number of ratings for that item. It can be calculated by using Equation 6.4.

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \frac{\left|r_{u,i} - \overline{r_i}\right|}{\mathrm{t_i}}}{N_u} \tag{6.4}$$

Where N_u is the number of items rated by user u, $r_{u,i}$ is rating given by user u to item i. $\mathrm{t_i}$ is the number of ratings provided for item i. $\overline{r_i}$ is average of these ratings.

ii. **Weighted Deviation from Mean Agreement (WDMA)** is used to identify the anomalies that place a high weight on rating deviations for sparse items. WDMA [50] differs from RDMA only in that in denominator it contains square of number of ratings provided for an item thus reducing the weight associated with items rated by many users. It can be calculated by using Equation 6.5.

$$WDMA_u = \frac{\sum_{i=0}^{N_u} \frac{\left|r_{u,i} - \overline{r_i}\right|}{\left|\mathrm{t_i}\right|^2}}{N_u} \tag{6.5}$$

iii. **Degree of Similarity with Top Neighbors (DegSim)** [Chirita et al. 2005] finds the average similarity of a profile's top k nearest neighbors. Attack profiles are supposed to have high similarity with their top 25 nearest neighbors than authenticated users [Chiritia et al. 2005; Resnick et al. 1994]. It can be calculated by Equation 6.6.

$$DegSim_u = \frac{\sum_{v=1}^{k} sim_{u,v}}{k} \quad (6.6)$$

$sim_{u,v}$ is the similarity between user u and user v.

iv. **Length Variance (LengthVar)** it is used to find the variance in the length of a given profile from the average length in the database [50]. If our database contains very large number of items then it is unlikely that a genuine user contain very large length, because a genuine user have to enter them manually where an attack profile may contain large profile because tools are used to insert these profiles into the system. It is a very important and useful attribute for the detection of attack profile from the system. It can be calculated by using Equation 6.7.

$$LengthVar_u = \frac{|l_u - \bar{l}|}{\sum_{k \in U}(l_k - \bar{l})^2} \quad (6.7)$$

$\bar{l}$ is the average length of profiles in the system and l_u is the length of user u in the system.

6.3.1.2 Model Specific Attributes

[Burke et al. 2006a, 2006b; Mobasher et al. 2006b] have found that generic attributes are not very good choice for distinguishing the attack profiles from the authentic profiles. As previously shown that attack profiles are injected based on some characteristics. These model specific attributes focuses on recognize the distinctive signature of a particular attack model. Popular model specific attributes are as follows:

i. **Mean Variance (MeanVar)** it is used for average attacks [50]. It can be calculated by using Equation 6.8.

$$MeanVar_{(p_t,u)} = \frac{\sum_{i \in (P_u - P_t)}(r_{i,u} - \overline{r_i})^2}{|P_u|} \quad (6.8)$$

$|P_u|$ is the number of ratings in profile P_u. $\overline{r_i}$ is the mean rating of item i across all users in the system.

ii. **Filler Mean Target Difference (FMTD)** this attribute is used for bandwagon, reverse bandwagon attack and segment attacks [50]. It can be calculated by using Equation 6.9.

$$FMTD_u = \left(\frac{\sum_{i \in P_{u,T}} r_{u,i}}{|P_{u,T}|}\right) - \left(\frac{\sum_{k \in P_{u,F}} r_{u,k}}{|P_{u,F}|}\right) \quad (6.9)$$

$P_{u,T}$ is the set of items in P_u that are given maximum rating in bandwagon attack and minimum rating in case of reverse bandwagon attack in user u's profile and all other items in P_u become the set $P_{u,F}$.

6.4 Evaluation metrics

To measure the performance of classification, two standard metrics are used i.e. precision and recall [42]. A confusion matrix contains four set of entries for classification, two of them are positive and other two are negative. True positive and true negative are the set of profiles which are classified correctly either as attack or authentic respectively and false positive and false negative are the set of profiles that are classified incorrectly that is attack or authentic respectively. Recall counts the total number of actual attacks in the system. Precision counts the total number of profiles that are labeled as attack. These metrics are denoted as:

$$recall = \frac{\#true\ poitive}{\#true\ poitive + \#false\ negative} \tag{6.10}$$

$$precision = \frac{\#true\ poitive}{\#true\ poitive + \#false\ positive} \tag{6.11}$$

6.5 Ensemble Approach for Profile Classification

For distinguishing the attack profiles and genuine profiles, first we compare six supervised machine learning algorithms and measure their precision and recall. We compare popular machine learning models i.e. decision tree, random forest, ada boost, support vector machine (SVM), linear regression and neural network. A brief description of all these models already has been given in the section 3.2. To measure the robustness of these predictive models, we use *k*-fold cross validation, *k* is an integer. We use 10 for the value of *k.* We randomly partition, our samples into 10 subsamples. Out of these 10 subsamples, 4 samples are used for testing and remaining 6 subsamples are used for training. The purpose of doing this is that, every subsample will be used for both the training and testing.

After measuring the accuracy of all the models, we find the best three performers in the below 8 tables. By chance in almost every case, our top three performers are support

vector machine, random forest and neural network. We use their results to make our predictive ensemble model. We use voting approach to make our ensemble model. Voting approach follow a simple methodology that, if two model saying a user profile fake then we will mark it as fake and if two models saying a user profile genuine then we will mark that user profile as genuine. Advantage of making ensemble model is that, it will never give worst performance although it has a problem also that it will never give best performance also. But as we always focused on worst cases, in that scenario it will gives good performance.

Table 6.3: Performance analysis for 10% average attacks.

Filler Size	**1%**		**10%**		**20%**		**30%**		**40%**		**50%**	
Models	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Decision Tree	.90	.892	.921	.93	.928	.919	.939	.912	.94	.94	.961	.968
Random Forest	.929	.930	.939	.92	.948	.948	.952	.956	.962	.961	.973	.979
Ada Boost	.9	.908	.914	.918	.93	.924	.935	.938	.948	.934	.950	.943
SVM	.938	.927	.949	.94	.959	.959	.971	.975	.979	.981	.988	.989
Linear Regression	.89	.862	.89	.907	.918	.91	.925	.918	.929	.902	.93	.931
Neural Network	.949	.938	.95	.943	.951	.954	.959	.958	.968	.967	.979	.98
Ensemble	.939	.932	.946	.934	.953	.953	.961	.963	.968	.969	.98	.982

Table 6.4: Performance analysis for 10% random attacks.

Filler Size	**1%**		**10%**		**20%**		**30%**		**40%**		**50%**	
Models	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Decision Tree	.91	.890	.912	.908	.937	.941	.951	.95	.958	.964	.971	.968
Random Forest	.939	.931	.949	.92	.951	.952	.958	.96	.962	.961	.978	.981
Ada Boost	.872	.867	.864	.881	.886	.904	.92	.915	.929	.90	.938	.941
SVM	.956	.94	.968	.96	.973	.965	.989	.981	.989	.991	1	.99
Linear Regression	.861	.875	.88	.851	.878	.91	.892	.905	.907	.917	.927	.91
Neural Network	.941	.949	.958	.953	.961	.967	.979	.978	.988	.981	.992	.994
Ensemble	.945	.94	.958	.951	.944	.961	.962	.975	.973	.979	.99	.988

Table 6.5: Performance analysis for 10% bandwagon attacks.

Filler Size	**1%**		**10%**		**20%**		**30%**		**40%**		**50%**	
Models	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Decision Tree	.901	.892	.912	.917	.928	.90	.94	.945	.951	.944	.961	.960
Random Forest	.94	.938	.949	.944	.959	.962	.960	.961	.972	.978	.98	.988
Ada Boost	.87	.865	.871	.86	.882	.894	.895	.90	.908	.905	.92	.924
SVM	.94	.942	.952	.958	.968	.952	.97	.981	1	.99	1	.994
Linear	.89	.882	.901	.891	.912	.918	.922	.91	.939	.931	.944	.941

Regression												
Neural Network	.951	.945	.962	.96	.969	.972	.979	.98	.981	.99	1	.994
Ensemble	.943	.942	.954	.953	.965	.962	.969	.974	.984	.986	.991	.992

Table 6.6: Performance analyses for 10% reverse bandwagon attacks.

Filler Size	1%		10%		20%		30%		40%		50%	
Models	P	R	P	R	P	R	P	R	P	R	P	R
Decision Tree	.892	.901	.884	.90	.9	.931	.918	.905	.921	.918	.931	.948
Random Forest	.940	.924	.944	.949	.959	.942	.961	.962	.982	.967	.99	.992
Ada Boost	.89	.872	.879	.881	.904	.908	.918	.90	.93	.918	.952	.945
SVM	.92	.902	.942	.938	.948	.918	.97	.971	.992	.99	1	.994
Linear Regression	.882	.892	.889	.902	.912	.918	.928	.949	.931	.931	.949	.942
Neural Network	.934	.941	.939	.916	.952	.932	.969	.958	.981	.989	1	.994
Ensemble	.933	.922	.942	.934	.953	.931	.967	.964	.985	.982	.992	.993

In table 6.2 to table 6.6, we examined the performance of above mentioned models of classification for three push attacks i.e. average, random, bandwagon attack and one nuke attack i.e. reverse bandwagon attack at fixed attack size of 10% and varying filler size by 1%, 10%, 30%, 40% and 50%. By analysis of above tables, we can say that as the filler size increases, all the models perform very well and it becomes easier to detect attack and authentic profiles. Some models give low accuracy by 86% at 1% filler size. When a user rate a large number of profiles in the system, the *LengthVar* attribute becomes very useful to distinguish profiles. As recall shows below 20% filler size, many attack profiles remains undetected but as the filler size increase, most of the profiles detected and very few remains undetected. Even at 50% filler size some models gives nearly 100% accuracy.

Table 6.7: Performance analysis for average attacks at 5% filler size.

Attack Size	1%		3%		6%		9%		12%		15%	
Models	P	R	P	R	P	R	P	R	P	R	P	R
Decision Tree	.822	.811	.844	.830	.859	.848	.868	.855	.889	.872	.901	.928
Random Forest	.880	.872	.894	.897	.905	.912	.909	.902	.929	.917	.957	.922
Ada Boost	.819	.802	.829	.810	.834	.828	.858	.840	.883	.868	.902	.915
SVM	.901	.902	.912	.928	.928	.908	.937	.911	.972	.959	.99	.984
Linear Regression	.862	.842	.872	.882	.882	.908	.908	.919	.918	.935	.939	.941
Neural Network	.892	.901	.919	.911	.922	.929	.959	.958	.961	.969	.97	.964
Ensemble	.891	.892	.908	.912	.918	.916	.935	.924	.954	.948	.972	.957

Table 6.8: Performance analysis for random attack at 5% filler size.

Attack Size	1%		3%		6%		9%		12%		15%	
Models	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Decision Tree	.838	.821	.848	.831	.858	.844	.866	.857	.889	.877	.911	.918
Random Forest	.891	.882	.894	.894	.909	.922	.929	.942	.949	.937	.969	.972
Ada Boost	.849	.832	.859	.849	.864	.867	.878	.849	.893	.888	.922	.919
SVM	.901	.912	.922	.918	.938	.948	.957	.941	.972	.989	.999	.989
Linear Regression	.852	.862	.878	.881	.892	.904	.918	.919	.928	.915	.949	.931
Neural Network	.902	.904	.929	.918	.932	.925	.959	.954	.971	.969	.989	.974
Ensemble	.898	.899	.915	.91	.926	.932	.948	.946	.964	.965	.986	.978

Table 6.9: Performance analysis for bandwagon attack at 5% filler size.

Attack Size	1%		3%		6%		9%		12%		15%	
Models	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Decision Tree	.841	.834	.854	.831	.878	.852	.896	.877	.909	.897	.921	.938
Random Forest	.901	.912	.914	.924	.939	.922	.951	.947	.969	.967	.982	.972
Ada Boost	.859	.842	.879	.854	.884	.877	.901	.889	.913	.891	.934	.929
SVM	.914	.918	.929	.917	.958	.947	.977	.963	.992	.989	1	.999
Linear Regression	.872	.842	.888	.871	.902	.894	.916	.917	.934	.924	.957	.951
Neural Network	.908	.892	.927	.928	.941	.931	.962	.959	.979	.969	.999	.984
Ensemble	.908	.907	.923	.924	.946	.933	.963	.953	.98	.975	.993	.985

Table 6.10: Performance analysis for reverse bandwagon attack at 5% filler size.

Attack Size	1%		3%		6%		9%		12%		15%	
Models	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Decision Tree	.832	.824	.844	.834	.862	.855	.886	.872	.909	.887	.921	.928
Random Forest	.911	.902	.925	.934	.948	.937	.948	.957	.968	.957	.989	.983
Ada Boost	.869	.852	.889	.874	.904	.897	.911	.901	.923	.908	.934	.939
SVM	.924	.908	.938	.927	.959	.948	.979	.968	.997	.99	1	.999
Linear Regression	.882	.872	.892	.881	.912	.904	.926	.919	.945	.934	.962	.953
Neural Network	.918	.902	.928	.918	.948	.934	.967	.968	.988	.979	1	.998
Ensemble	.918	.904	.930	.926	.952	.939	.965	.964	.984	.975	.996	.986

In table 6.7 to table 6.10, we examined the performance of above mentioned models of classification for three push attacks i.e. average, random, bandwagon attack and one nuke attack i.e. reverse bandwagon attack at fixed filler size of 5% and varying attack size by 1%, 3%, 6%, 9%, 12% and 15%. By analysis of above tables, we can say that as the attack size increases, all the models perform very well and it becomes easier to detect attack and authentic profiles. Some models give accuracy as low as 81% at 1% attack size. Reason for this low accuracy is that at 1% attack size only 10 attack profiles is

inserted in the system. As compared to 943 authentic users attack profiles remains very less. Because of improper ratios of these two types of profiles some models does not give good performance. But when we increase the attack size after 6%, a sharp rise in the accuracy is clearly visible. At 15% of attack size, very good accuracy is given by many models in almost every case. Reason for this accuracy is at 15% attack size we insert 150 attack profiles in the system and the ratio between authentic and attack profiles become very good as a result proper training and testing is possible.

6.6 Clustering Approach for the Attack Detection

In this section, we give a comparative study to analyze the accuracy of four clustering methods to distinguish authentic and genuine profiles. We compare expectation-maximization (EM), farthest first, hierarchical clustering and simple K mean using weka. A brief description of all these models has been given in section 3.2.2.

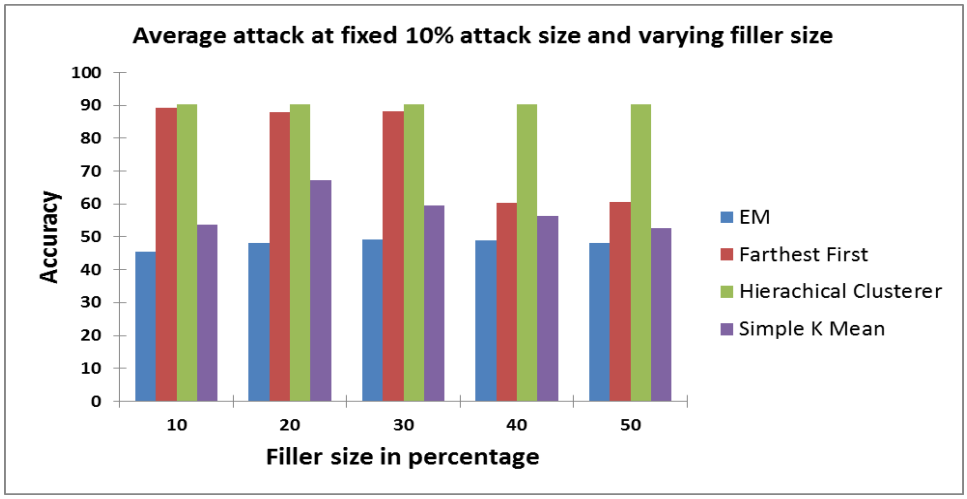

Figure 6.5: Performance analysis for 10% average attack.

In Figure 6.5, we compare the accuracy of four clustering models at 10% attack size and varying filler size for average attack. EM and hierarchical clustering model gives almost constant accuracy in all cases but EM gives least accuracy where hierarchical gives maximum accuracy. Farthest first gives good performance up to 30% filler size but it falls sharply 40% filler size onwards.

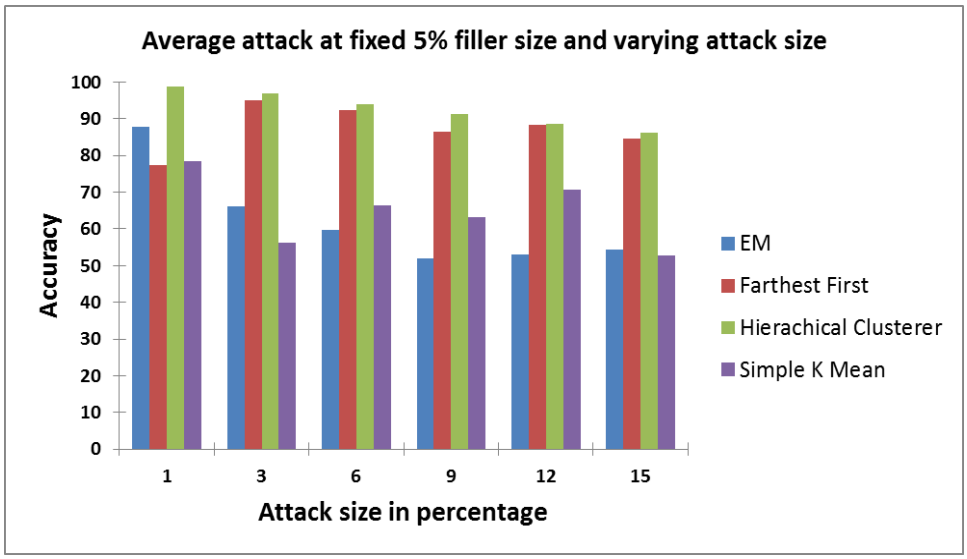

Figure 6.6: Performance analysis for average attacks at 5% filler size.

In Figure 6.6, we compare the accuracy of four clustering models at fixed 5% filler size and varying attack size for average attack. At 1% attack size, EM gives 87% accuracy but attack size 3% onwards it performance falls and gives accuracy in the range 50% to 65%. Hierarchical clustering model gives best accuracy in all cases although its accuracy decreases. Farthest First and Hierarchical clustering models gives comparable accuracy after attack size 3% onwards.

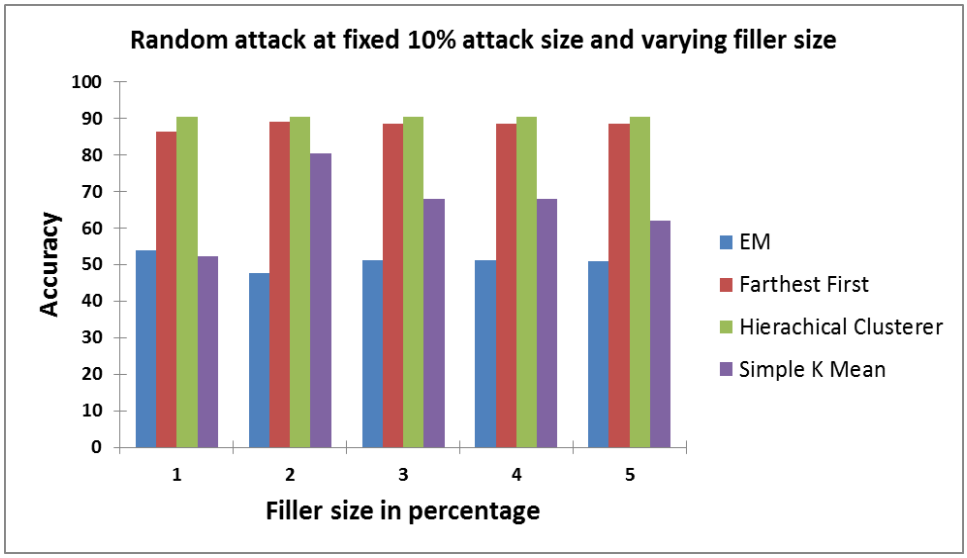

Figure 6.7: Performance analysis for 10% random attacks.

In Figure 6.7, we compare the accuracy of four clustering models at 10% attack size and varying filler size for random attack. EM in this case gives worst performance almost same as shown in Figure 6.5. Farthest First and Hierarchical clustering models gives comparable and almost constant performance in all case although performance of Hierarchical clustering method is slightly better than Farthest First.

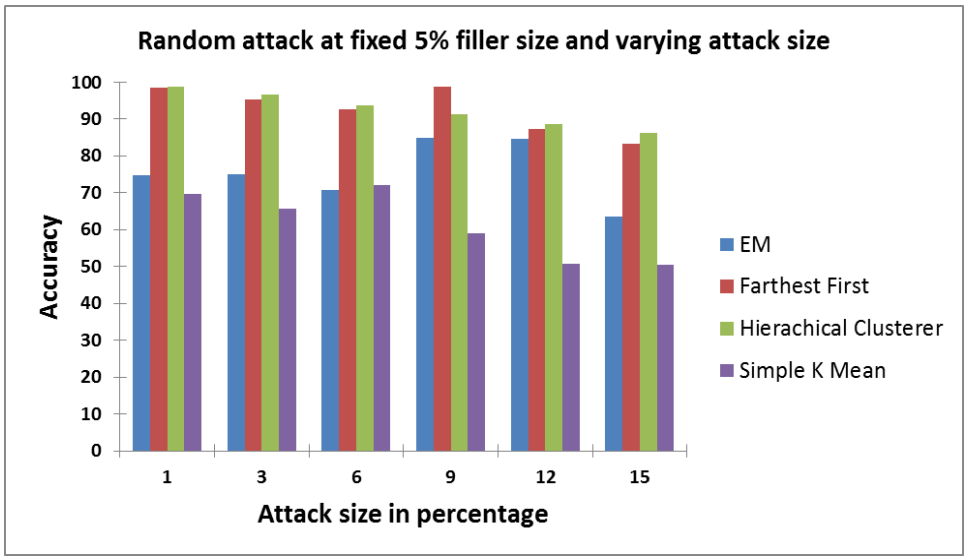

Figure 6.8: Performance analysis for random attack at 5% filler size.

In Figure 6.8, we compare the accuracy of four clustering models at fixed 5% filler size and varying attack size for random attack. At 1% attack size, Farthest First and Hierarchical clustering gives best accuracy about 98%. In all cases Hierarchical clustering model perform well except at 9% attack size where Farthest First gives best accuracy about 98%.

In Figure 6.9, we compare the accuracy of four clustering models at fixed 10% attack size and varying filler size for random attack for bandwagon attack. Farthest First and Hierarchical clustering methods gives best and constant accuracy about 90% in all cases. Although accuracy of other two models keep changing in all cases.

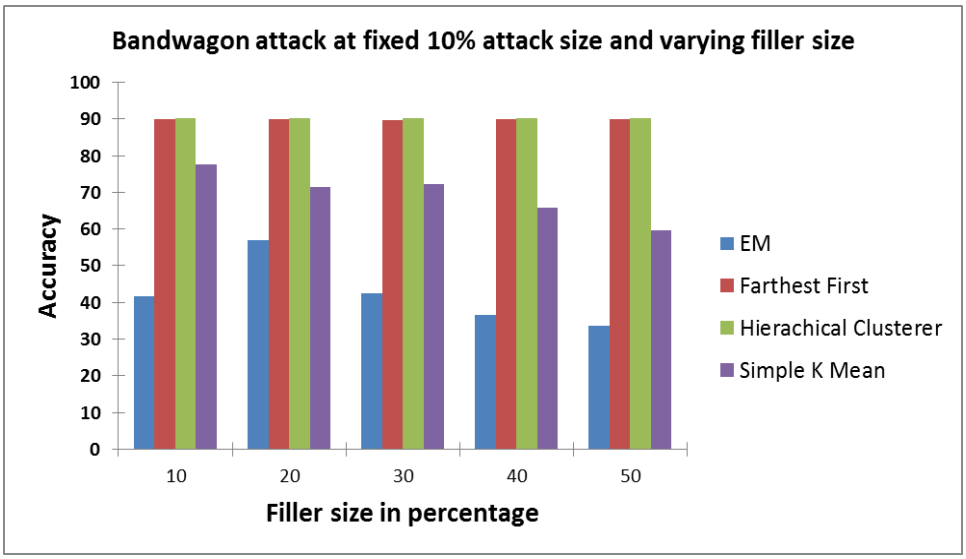

Figure 6.9: Performance analysis for 10% bandwagon attack.

In Figure 6.10, we compare the accuracy of four clustering models at fixed 5% filler size and varying attack size for bandwagon attack. Up to 6% attack size Hierarchical clustering gives best accuracy but after that Farthest First gives best accuracy even it also gives equal accuracy to Hierarchical clustering method at 1% and 6%.

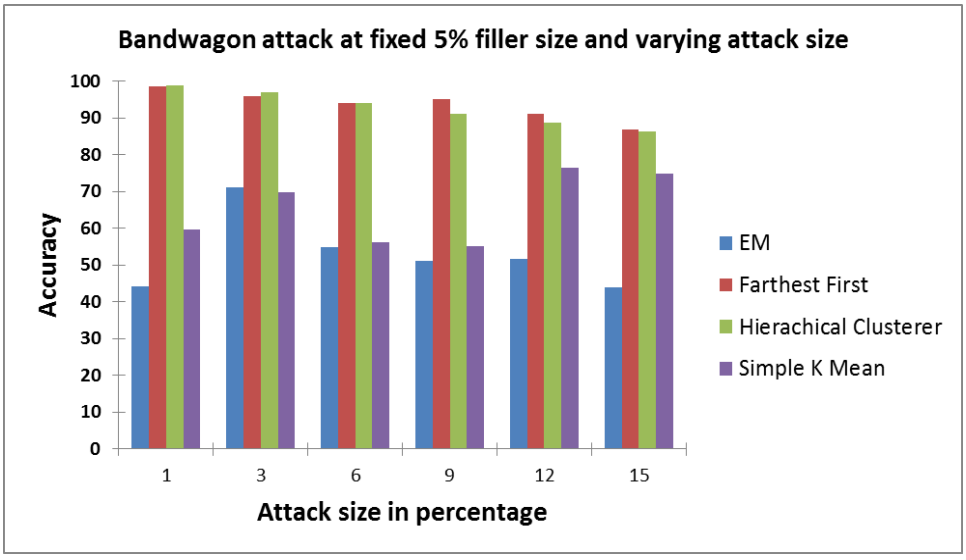

Figure 6.10: Performance analysis for bandwagon attack at 5% filler size.

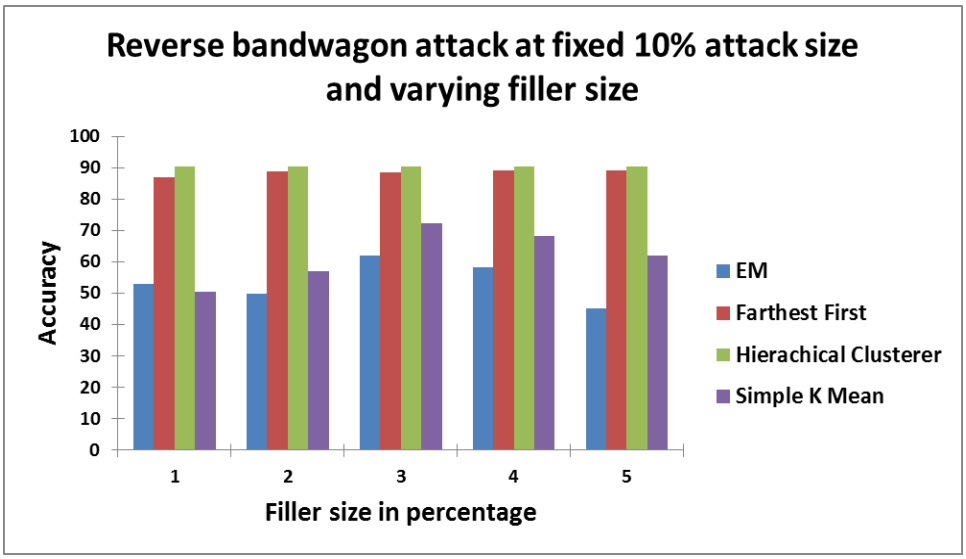

Figure 6.11: Performance analyses for 10% reverse bandwagon attacks.

In Figure 6.11, we compare the accuracy of four clustering models at fixed 10% attack size and varying filler size for random attack for reverse bandwagon attack. In all cases Hierarchical clustering method gives best accuracy in all cases, although Farthest First also gives comparable accuracy to Hierarchical method where EM perform least accurately in almost all cases.

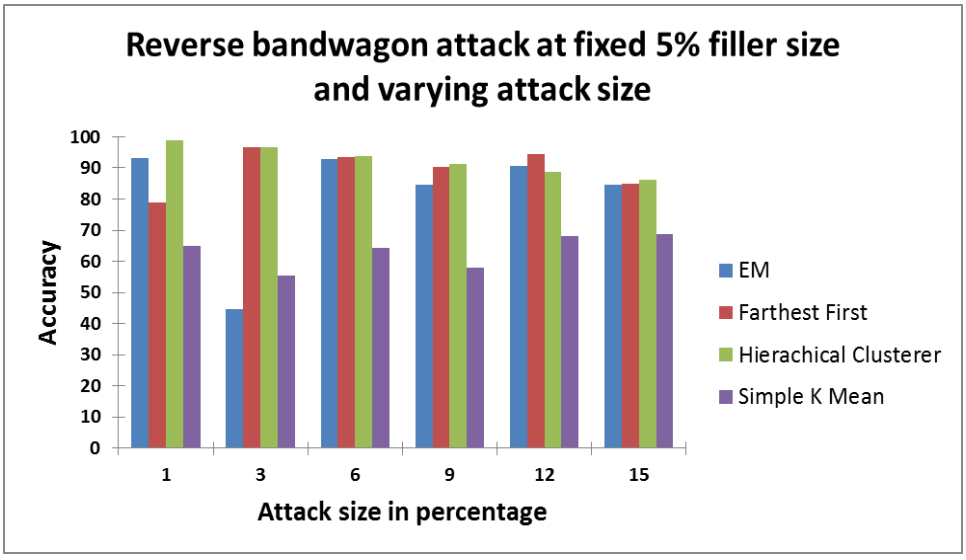

Figure 6.12: Performance analysis for reverse bandwagon attack at 5% filler size.

In Figure 6.12, we compare the accuracy of four clustering models at fixed 5% filler size and varying attack size for reverse bandwagon attack. At 3% attacks size both Farthest

First and Hierarchical clustering gives equal accuracy i.e. 95%. At 6% and 15% EM, Farthest First and Hierarchical clustering methods gives almost same accuracy.

From the analysis of clustering methods in Figure 6.4 to Figure 6.5, we can say that in almost all cases hierarchical clustering method perform with accuracy in the range 80% to 98% and it perform better than any other clustering method. Simple K Mean method does not perform best in any of the case. Farthest First and Hierarchical clustering method gives comparable performance in many cases. In few cases, EM gives good performance but as compared to Farthest First and Hierarchical clustering methods its performance is poor in most of the cases.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

We focused on finding the several keys in the area of attacks against recommender systems. We have shown that, even without the knowledge of systems or users; it is possible to mount successful attacks against collaborative recommender systems. Also we concluded that mounting the bandwagon attack for nuke attacks is effective but not for push attack. We also did investigation in the limited knowledge nuke attacks such as love-hate attack and reverse bandwagon attack. We have found that love-hate attack not only requires least amount of the system knowledge, but it is also more effective of this type of attacks.

We have presented a supervised classification approach for attack detection. Using the generic and model specific attributes, we were able to detect the attacks. We compare the performance of six statistical models and compare their performance. We find the best three performing models and ensemble them to develop our model by using voting technique of ensembling. We found that attacks like love-hate attack still present some problem as it can impact the system's recommendation even at low attack size. We also presented a comparative study of unsupervised approach for the attack detection.

7.2 Future Work

i. Proposed system can be applied on the emails to separate the spam and non-spam emails; attributes need to be selected according to the emails.

ii. This system can also be applied in the identification of the DOS attacks on the server. An attacker may generate a large number of requests to server to make down it. To find such requests on the server this proposed system can be used.

iii. More research can be done in the area of finding the more attributes for the detection of attacks as more the attributes more will be the accuracy of the system.

REFERENCES

[1]. Davoodi, Fatemeh Ghiyafeh, and Omid Fatemi. "Tag based recommender system for social bookmarking sites." In *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, pp. 934-940. IEEE, 2012.

[2]. Goldberg, David, David Nichols, Brian M. Oki, and Douglas Terry. "Using collaborative filtering to weave an information tapestry." *Communications of the ACM* 35, no. 12 (1992): 61-70.

[3]. Ricci, Francesco. "Travel recommender systems." *IEEE Intelligent Systems*17, no. 6 (2002): 55-57.

[4]. Mladenic, Dunja. "Text-learning and related intelligent agents: a survey." *IEEE Intelligent Systems* 4 (1999): 44-54.

[5]. Linden, Greg, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering." *Internet Computing, IEEE* 7, no. 1 (2003): 76-80.

[6]. Gross, Ralph, and Alessandro Acquisti. "Information revelation and privacy in online social networks." In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pp. 71-80. ACM, 2005.

[7]. Xiao, Bo, and Izak Benbasat. "E-commerce product recommendation agents: Use, characteristics, and impact." *Mis Quarterly* 31, no. 1 (2007): 137-209.

[8]. Dakhel, Gilda Moradi, and Mehregan Mahdavi. "A new collaborative filtering algorithm using K-means clustering and neighbors' voting." In *Hybrid Intelligent Systems (HIS), 2011 11th International Conference on*, pp. 179-184. IEEE, 2011.

[9]. Kim, Hyun-Tea, Eungyeong Kim, Jong-Hyun Lee, and Chang Wook Ahn. "A recommender system based on genetic algorithm for music data." In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 6, pp. V6-414. IEEE, 2010.

[10]. Shardanand, Upendra, and Pattie Maes. "Social information filtering: algorithms for automating “word of mouth”." In *Proceedings of the SIGCHI conference on*

Human factors in computing systems, pp. 210-217. ACM Press/Addison-Wesley Publishing Co., 1995.

[11]. Konstan, Joseph A., Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. "GroupLens: applying collaborative filtering to Usenet news." *Communications of the ACM* 40, no. 3 (1997): 77-87.

[12]. Resnick, Paul, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. "GroupLens: an open architecture for collaborative filtering of netnews." In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175-186. ACM, 1994.

[13]. Hill, Will, Larry Stead, Mark Rosenstein, and George Furnas. "Recommending and evaluating choices in a virtual community of use." In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 194-201. ACM Press/Addison-Wesley Publishing Co., 1995.

[14]. Su, Xiaoyuan, and Taghi M. Khoshgoftaar. "A survey of collaborative filtering techniques." *Advances in artificial intelligence* 2009 (2009): 4.

[15]. Bedi, Punam, and Sumit Kr Agarwal. "Managing Security in Aspect Oriented Recommender System." In *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, pp. 709-713. IEEE, 2011.

[16]. Smyth, Barry. "Case-based recommendation." In *The adaptive web*, pp. 342-376. Springer Berlin Heidelberg, 2007.

[17]. Linden, Greg, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering." *Internet Computing, IEEE* 7, no. 1 (2003): 76-80.

[18]. Kitts, Brendan, David Freed, and Martin Vrieze. "Cross-sell: a fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities." In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 437-446. ACM, 2000.

[19]. Breese, John S., David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." In *Proceedings of the*

Fourteenth conference on Uncertainty in artificial intelligence, pp. 43-52. Morgan Kaufmann Publishers Inc., 1998.

[20]. Karypis, George. "Evaluation of item-based top-n recommendation algorithms." In *Proceedings of the tenth international conference on Information and knowledge management*, pp. 247-254. ACM, 2001.

[21]. Burke, Robin. "Hybrid recommender systems: Survey and experiments." *User modeling and user-adapted interaction* 12, no. 4 (2002): 331-370.

[22]. Linden, Gregory D., Jennifer A. Jacobi, and Eric A. Benson. "Collaborative recommendations using item-to-item similarity mappings." U.S. Patent 6,266,649, issued July 24, 2001.

[23]. Claypool, Mark, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. "Combining content-based and collaborative filters in an online newspaper." In *Proceedings of ACM SIGIR workshop on recommender systems*, vol. 60. 1999.

[24]. Isaksson, Anders, Mikael Wallman, Hanna Göransson, and Mats G. Gustafsson. "Cross-validation and bootstrapping are unreliable in small sample classification." *Pattern Recognition Letters* 29, no. 14 (2008): 1960-1965.

[25]. Jolliffe, Ian. *Principal component analysis*. John Wiley & Sons, Ltd, 2002.

[26]. Goldberg, Ken, Theresa Roeder, Dhruv Gupta, and Chris Perkins. "Eigentaste: A constant time collaborative filtering algorithm." *Information Retrieval* 4, no. 2 (2001): 133-151.

[27]. Golub, Gene H., and Christian Reinsch. "Singular value decomposition and least squares solutions." *Numerische mathematik* 14, no. 5 (1970): 403-420.

[28]. Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl. *Application of dimensionality reduction in recommender system-a case study*. No. TR-00-043. Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[29]. Brand, Matthew. "Fast Online SVD Revisions for Lightweight Recommender Systems." In *SDM*, pp. 37-46. 2003.

[30]. Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl. "Incremental singular value decomposition algorithms for highly scalable recommender

systems." In *Fifth International Conference on Computer and Information Science*, pp. 27-28. 2002.

[31]. Quinlan, J. Ross. "Induction of decision trees." *Machine learning* 1, no. 1 (1986): 81-106.

[32]. Cho, Yoon Ho, Jae Kyeong Kim, and Soung Hie Kim. "A personalized recommender system based on web usage mining and decision tree induction."*Expert systems with Applications* 23, no. 3 (2002): 329-342.

[33]. Zhang, Xiang-Sun. "Introduction to artificial neural network." In *Neural Networks in Optimization*, pp. 83-93. Springer US, 2000.

[34]. Hsu, Shang H., Ming-Hui Wen, Hsin-Chieh Lin, Chun-Chia Lee, and Chia-Hoang Lee. "AIMED-A personalized TV recommendation system." In *Interactive TV: a Shared Experience*, pp. 166-174. Springer Berlin Heidelberg, 2007.

[35]. Christakou, Christina, Spyros Vrettos, and Andreas Stafylopatis. "A hybrid movie recommender system based on neural networks." *International Journal on Artificial Intelligence Tools* 16, no. 05 (2007): 771-792.

[36]. Cristianini, Nello, and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[37]. Oku, Kenta, Shinsuke Nakajima, Jun Miyazaki, and Shunsuke Uemura. "Context-aware SVM for context-dependent information recommendation." In*Proceedings of the 7th international Conference on Mobile Data Management*, p. 109. IEEE Computer Society, 2006.

[38]. Hanhoon, K. A. N. G., and Seong Joon Yoo. "Svm and collaborative filtering-based prediction of user preference for digital fashion recommendation systems." *IEICE transactions on information and systems* 90, no. 12 (2007): 2100-2103.

[39]. Xue, Gui-Rong, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. "Scalable collaborative filtering using cluster-based smoothing." In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 114-121. ACM, 2005.

[40]. Burke, Robin, Bamshad Mobasher, Roman Zabicki, and Runa Bhaumik. "Identifying attack models for secure recommendation." In *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*. 2005.

[41]. Lam, Shyong K., and John Riedl. "Shilling recommender systems for fun and profit." In *Proceedings of the 13th international conference on World Wide Web*, pp. 393-402. ACM, 2004.

[42]. O'Mahony, Michael, Neil Hurley, Nicholas Kushmerick, and Guénolé Silvestre. "Collaborative recommendation: A robustness analysis." *ACM Transactions on Internet Technology (TOIT)* 4, no. 4 (2004): 344-377.

[43]. Deng, Houtao, George Runger, and Eugene Tuv. "Bias of importance measures for multi-valued attributes and solutions." *Artificial Neural Networks and Machine Learning–ICANN 2011* (2011): 293-300.

[44]. Rätsch, Gunnar, Takashi Onoda, and K-R. Müller. "Soft margins for AdaBoost."*Machine learning* 42, no. 3 (2001): 287-320.

[45]. Zhang, Yongyue, Michael Brady, and Stephen Smith. "Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm." *Medical Imaging, IEEE Transactions on* 20, no. 1 (2001): 45-57.

[46]. Moon, Tood K. "The expectation-maximization algorithm." *Signal processing magazine, IEEE* 13, no. 6 (1996): 47-60.

[47]. Zhang, Ji-Hu, Thomas DY Chung, and Kevin R. Oldenburg. "A simple statistical parameter for use in evaluation and validation of high throughput screening assays." *Journal of biomolecular screening* 4, no. 2 (1999): 67-73.

[48]. Langfelder, Peter, Bin Zhang, and Steve Horvath. "Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R." *Bioinformatics*24, no. 5 (2008): 719-720.

[49]. Bilenko, Mikhail, Sugato Basu, and Raymond J. Mooney. "Integrating constraints and metric learning in semi-supervised clustering." In *Proceedings of the twenty-first international conference on Machine learning*, p. 11. ACM, 2004.

[50]. Mobasher, Bamshad, Robin Burke, Runa Bhaumik, and Chad Williams. "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness." *ACM Transactions on Internet Technology (TOIT)* 7, no. 4 (2007): 23.

[51]. Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer* 8 (2009): 30-37.

LIST OF PUBLICATIONS

[1]. Kumar Ashish, Garg Deepak, and Rana Prashant. “Ensemble approach to detect profile injection attack in recommender system,” *Fourth International Conference on Advances in Computing, Communications and Informatics (ICACCI-2015), IEEE,* August 10-13, 2015, Kochi. [Accepted]

[2]. Kumar Ashish, Garg Deepak, and Rana Prashant “Clustering approach to detect profile injection attack in recommender system,” *PCITC-2015, IEEE,* October 15-17, 2015, Bhubaneshwar. [Accepted]

[3]. Kumar, Ashish. "Software Architecture Styles: A Survey." *International Journal of Computer Applications* 87(9), 2014.

VIDEO LINK

1. https://www.youtube.com/watch?v=7FsrMneUZl4&feature=youtu.be