

Comparative Study of different Recommender System Techniques

A dissertation submitted in partial fulfilment of the requirements for the MSc in

Data Science

by

SAWSAN SHAKIR

Department of Computer Science and Information Systems
Birkbeck, University of London

September 2019

Supervisor: David Weston

Word count excluding appendixes 11,563

Academic declaration

“This report is substantially the result of my own work except where explicitly indicated in the text. I give my permission for it to be submitted to the TURNITIN Plagiarism Detection Service.

I have read and understood the sections on plagiarism in the Programme Handbook and the College website.

The report may be freely copied and distributed provided the source is explicitly acknowledged.”

Abstract

Recommender systems have advanced rapidly in many online businesses and there are many techniques used to build recommender systems but it is not easy to choose the method that performs best.

The aim of project is to compare recommender system algorithms by their performance which is carried out using three main evaluation measures, which are: prediction accuracy represented by mean absolute error and mean square root error, classification accuracy represented by precision - recall curve and true positive rate - false positive rate and finally, ranking accuracy represented by mean average precision, mean average recall, normalized discounted cumulative gain and coverage.

Four common algorithms are reviewed and implemented which are user-based collaborative filtering, item-based collaborative filtering, singular value decomposition model-based and we tried to improve the user-based collaborative filtering by implementing a hybrid recommender system using demographic features of the users as an enhanced factor with rating similarity features. Two datasets have been used MovieLens ML-1m and Book-crossing.

After implementing the project, it became clear that there is no particular algorithm that we can prove to be best among other techniques when applied to the two datasets, as we found out that the best technique is not only algorithm dependant but also dataset and domain dependant.

Table of Contents

Academic declaration.....	2
Abstract.....	3
List of Figures	6
List of Tables	7
Acknowledgements.....	8
1- Introduction	9
1-1 Overview of Recommender systems	9
1-2 The project Aims	10
1-3 The project Objectives	10
1-4 Roadmap	11
2 - Background	12
2-1 Recommendation system workflow	12
2-2 Goals of recommender system.....	12
2-2-1 Rating prediction problem.....	12
2-2-2 Ranking top-k problem	12
2-3 Categories of recommender systems	12
3 Methodology.....	15
3-1 user-based collaborative filtering (UBCF)	15
3-2 Enhanced user-based collaborative filtering with demographic features (EUBCF).....	16
3-3 Item-based collaborative filtering (IBCF)	17
3-4 Singular value decomposition based Collaborative filtering (SVD).....	18
3-5 Evaluation Protocols	19
3-5-1 Accuracy of predictions:	19
3-5-2 Accuracy of classifications:	20
3-5-3 Rank accuracy:	22
4 Comparison and Evaluation of the Experiments.....	25
4-1 The Datasets	25
4-1-1 Data Cleansing	26
4-1-2 Statistical Analysis.....	26
4-2 creating training and testing set	29
4-3 Experiments implementation	30
4-3-1 user-based collaborative filtering (UBCF)	31

4-3-2 Enhanced User-based with demographic features (EUBCF)	31
4-3-3 Item-based collaborative filtering (IBCF)	32
4-3-4 Single value decomposition (SVD)	33
4-4 Testing the prediction implementation code	33
4-5 Results and techniques Comparison.....	34
5- Conclusion and future work.....	44
5-1 Conclusion.....	44
5-2 Future work and improvement.....	45
6- References	46
Appendixes.....	49
Appendix A	49
Appendix B	50
Appendix C	51
Appendix D	54
Appendix E	55
Appendix F	57
Appendix G.....	59
Appendix H.....	61
Appendix I	63

List of Figures

Figure 1 Recommendation system workflow	12
Figure 2 the form of a singular-value decomposition	18
Figure 3 Receiver operating characteristic (ROC) curve	21
Figure 4 Precision-Recall curve	21
Figure 5 1-5 Range of ratings distribution for ML-1m.....	26
Figure 6 1-10 Range of explicit ratings distribution for Book-Crossing.....	27
Figure 7 distribution for ML-1m.....	28
Figure 8 distribution for Book-Crossing	28
Figure 9 Age of user's distribution for ML-1m	28
Figure 10 Age of user's distribution for Book-Crossing.....	29
Figure 11 Training and testing set in recommender system.....	29
Figure 12 ML-1m Precision-Recall curve.....	36
Figure 13 ML-1m ROC curve	36
Figure 14 Book-Crossing Precision-Recall curve	36
Figure 15 Book-Crossing ROC curve.....	36
Figure 16 Enlarged ROC curve of the Book-Crossing for IBCF algorithm	37
Figure 17 Mean average precision at K comparison for ML-1m.....	38
Figure 18 Mean average precision at K comparison for Book-Crossing	38
Figure 19 Mean average Recall at K comparison for ML-1m	39
Figure 20 Mean average Recall at K comparison for Book-Crossing.....	40
Figure 21 Catalog coverage % for ML-1m	42
Figure 22 Catalog coverage % for Book-Crossing.....	42
Figure 23 Types of recommender system methods.....	49

List of Tables

Table 1 Confusion table of recommender system	20
Table 2 Structure of user demographic vector for ML-1m	31
Table 3 Structure of user demographic vector for Book-Crossing	32
Table 4 The prediction accuracy results for ML-1m.....	34
Table 5 The prediction accuracy results for Book-Crossing	34
Table 6 Confusion tables result of the experiments in ML-1m.....	35
Table 7 Confusion tables result of the experiments in Book-Crossing	37
Table 8 Mean Normalized Discounted Cumulative Gain (NDCG) for ML-1m	41
Table 9 Mean Normalized Discounted Cumulative Gain (NDCG) for Book-Crossing.....	41
Table 10 advantages and disadvantages of the recommender systems methods	50

Acknowledgements

I would like to thank my project supervisor Dr David Weston of the Department of Computer Science at Birkbeck to whom I am grateful for helping me throughout this project with guidance and advice.

In addition, I would like to thank professor George Magoulas for his useful and valuable advice in helping to choose the suitable algorithms.

I would also like to sincerely thank my husband for his full supports over the study years.

Thank you

Sawsan Shakir

1- Introduction

1-1 Overview of Recommender systems

In the past, people used to buy products they need from different shops and spend many hours until they find what they are looking for. Later on with the advent of the World Wide Web, finding information about products became easier due to the easy access to a huge variety of information readily available online, which came about with the rapid growth of e-business services (buying products, product comparison, auction, etc.) (Dietmar et al., 2011).

However, the amount of the information can overwhelm the user and make it harder to make good choices, an example is when searching for “Data Science” on Amazon UK site under the books section and over 90 thousand matches are found. So, the increased choice changed from being an improvement to being a detriment. Also, the abundance of choice not only makes it harder for the user to choose but can potentially stop the user from choosing altogether and so reducing sales and hence profitability.

To alleviate the above web sites started offering products based a new breed of computer systems called Recommender Systems. Where as well as of providing the numerous matches to a user’s query, Recommender Systems try to provide a small number of products that are expected to be what the user is actually seeking.

These new systems use a variety of techniques to find the suitable products, and in essence they attempt to provide the following features (Charu, 2016):

- Relevance: to find the product the user is seeking quickly and without browsing through a large number of products
- Novelty: something new to the user that he may not know is available
- Serendipity: recommend a product that is surprising and not expected with the anticipation that it will be desirable.
- Diversity: to recommend products of a different types to increase the chance that the user would like some of these.

In general, recommender engines which are a section of information retrieval and artificial intelligence, are giant tools used to analyse big data. Particularly user activity and products information to make suggestions based on the analysis results and make better decisions from the many alternatives available over the Web (Gorakala, 2016)

In technical terms, a recommendation engine task is an objective function which can predict how much an item will be useful to the user.

If $U = \{\text{users}\}$, $I = \{\text{items}\}$ then $F = \text{Objective function that measures the usefulness of items (I) to users (U)}$, given by:

$$F: U \times I \rightarrow R$$

Where $R = \{\text{recommended items}\}$.

For each user (u), we want to choose the item (i) that maximizes the objective function (Gorakala, 2016):

$$u \in U, I_u = \operatorname{argmax}_u u(u, i)$$

1-2 The project Aims

The aim of the project is to compare the performance of different techniques (also referred to as algorithms in this document) used in recommender systems when applied to the same dataset and under the same conditions. Four algorithms are implemented starting from the simple memory-based method which depends on similarities in historical ratings to the more advance technique like SVD which is used to find out the latent factors. The comparison is based on three performance measures which are Prediction Accuracy, Classification Accuracy and Ranking Accuracy. This comparative study expects to produce a better understanding of which algorithms works best and under what conditions.

A secondary aim is for myself to explore the field of recommender systems and learn about the different techniques and how to apply them in practice. This involved performing the different stages of constructing a data science system from data gathering to researching to find the most suitable techniques for the recommender algorithms that were applied.

Furthermore, another aim was learning about the different evaluating methods to find the most performing technique to gain the most out of such systems.

This project will also be an opportunity for myself to improve my software development skills applied in python as I am a new convert to the fields. This involved researching and finding the most suitable data structure to manage manipulating the large amount of data efficiently. This also required finding applicable methods from different libraries to perform the required tasks.

1-3 The project Objectives

In order to achieve the aims discussed in the previous section:

- Selecting the data set:
The Movielens and Book-Crossing datasets are found to be suitable for the project requirements as have users' numeric ratings of the movies and books they have reviewed and these datasets have demographic features for the users which are required to implement one of the algorithms. Some datasets have textual reviews which require text analysis which is a different field of data science outside the scope of this project's aims.

- Choosing the algorithms:

There is a wide range of techniques for recommender systems and the most common ones are memory-based and model-based. Both of which we have been applied in this project. For the memory-based we used user-based and item-based collaborative filtering and for the model based we used singular value decomposition. Furthermore, we also applied the enhanced user based collaborative filtering technique as a hybrid technique expecting a better performance.

- Implementation of the algorithms

As much as possible within the time constraints we aimed to implement the techniques from first principle in python. When not possible due to the mentioned time constraints then suitable libraries and their functions we researched and applied to aid in achieving the implementation of finding the desired ratings predictions.

- Evaluation measures

With recommender systems there are three main evaluation methods which are Prediction Accuracy, Classification Accuracy and Ranking Accuracy.

These were applied in this project and for the Ranking Accuracy out of the numerous methods we have applied Mean Average Precision, Normalized Discounted Cumulative Gain and Coverage.

1-4 Roadmap

Section 2 describes the goals and common types of recommender systems and outlines the advantage and disadvantage of these methods.

A Project methodology is in Section 3, outlining the key steps of implementing the four algorithms we used and describes the evaluation methods we used to compare the four techniques.

Section 4 discusses the results and compares evaluation measurements scored by the four algorithms.

Section 5 gives a summary, conclusion and recommendations for future work.

2 - Background

The following sections explain the types of the recommendation systems, the techniques and algorithms used in order to build a recommender system.

2-1 Recommendation system workflow

In order to build a recommender system, figure (1) shows the main steps. Firstly, the training dataset is provided to the recommendation algorithm. Then, the recommendation model is created which is used to predict a new dataset. In order to evaluate the system, the test dataset is fed to the model and the predictions are created for all ratings. The known ratings and their predictions are used to evaluate the system (Malaeb, 2017).

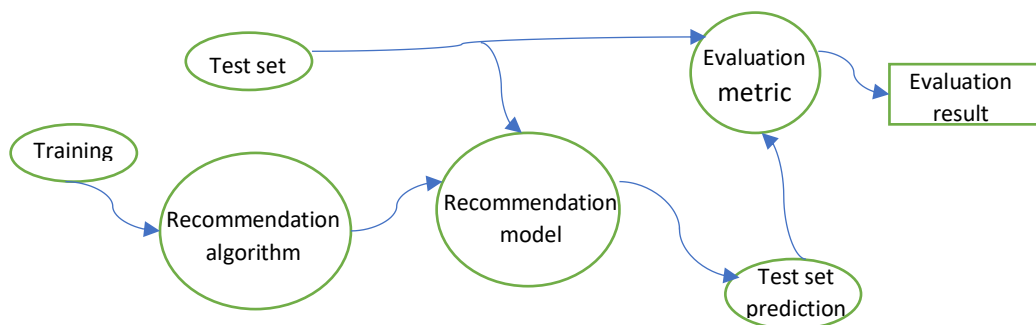


Figure 1 Recommendation system workflow

2-2 Goals of recommender system

Before explaining the methods and techniques in details, we will introduce two main recommendation tasks (Charu, 2016):

2-2-1 Rating prediction problem

This task is to predict the rating value for user-item matrix. The user preferences for item is computed from the known ratings of that user and it is used as model to predict the missing values. This problem is called matrix completion problem because in predicting the missing values we complete the incompleated matrix of ratings.

2-2-2 Ranking top-k problem

Determine the top-k items to recommend to a particular user or find the top-k users to recommend to them a particular item. This problem is known as the top-k recommendation problem.

The solution for the rating prediction can be made to solve the ranking of the top-k unrated items for the user.

2-3 Categories of recommender systems

There are a number of categories based on the techniques used to build the systems (Appendix A figure (23) provides a chart showing the numerous types of recommender algorithms).

The main three types are:

2-3-1 Collaborative filtering recommender systems

The main idea of collaborative recommendation approaches is to exploit information about the past behaviour or the opinions of an existing user community for predicting items to the current user that he most probably like or be interested in (Dietmar et al., 2011). Collaborative filtering takes its root from the idea of asking opinions of others which people have been doing for a long time.

Collaborative filtering is classified into two types: memory-based and model-based methods. Memory based generates the predictions by using all the ratings of user-item data to find the similarity between all users who share the same interests and send the target user a list of recommended items which have high predictions. The main advantages of memory based are the sparsity and scalability in addition to ease of implementing (Postmus S, 2018).

Model-based is similar to the supervised or unsupervised machine learning when the model is built from available data and using techniques like, clustering, matrix factorization, rule-based association and Bayesian. The unseen ratings are predicted based on the model which is created (Charu, 2016).

The advantage of this method is that it does not need memory to implement the model, and it is better with sparsity and scalability than memory-based which require a lot of memory and CPU time (Bansari et al., 2017).

2-3-2 Demographic Recommender Systems

Demographic recommenders serve recommendations based on demographic information of the users, which is used to classify users in a way to recommend specific products or predict ratings to these groups of users (Dietmar et al., 2011). This kind of system combined with other recommender systems like content-based or knowledge-based creates hybrid recommendation that solves the cold start problem. One other advantage of this system is not to rely on past ratings or focus on situational information but to depend on the characteristics of the user which are more stable over time (Theobald, 2018).

2-3-3 Content recommender systems

The user rates a movie, a book or any item because he/she likes one or more of the features of that item for example like the actors, the writer or the story, and hence is more likely that he/she will prefer items with the same features (Gorakala, 2016). Content-based system tries to match the items with features similar to items that are preferred in the past by the user and in this way, we only need the ratings of the active user and no need for the ratings of other users (Charu, 2016).

One of the major plus points of this technique is that it only requires item features and users' profile and it does not need for ratings or feedback from the users. While, the disadvantage is when there is not enough information about the items (Butola, 2016).

2-3-4 Hybrid recommender system

Improving the performance of the recommender systems can be achieved by combining two or more techniques. Each method of recommender system has advantages and disadvantages (Burke, 2002). And combining these systems offsets the disadvantages with the advantages (Gorakala, 2016).

There are many types of hybrid recommender system but we focused on feature combination hybrid which we used in one of our experiments due to availability of demographic feature in the dataset and this is explained in the next section.

Feature combination hybrid is based on the idea of combining the input data from different techniques. The most widely used is combining content and collaborative filtering. There are variety of ways to perform this with different types of information. For example, it is possible to construct the rating matrix in terms of genres rather than movie-id in movie recommender systems (Charu, 2016).

The feature combination hybrid makes the system less sensitive to the number of users who have rated items because it does not rely on the user ratings (Burke, 2002).

Appendix B table (10) illustrates the advantage and disadvantage of the main methods of recommender system.

3 Methodology

Throughout this section we will be describing the methods that are used in our experiments and the evaluation measurements that are used to compare the algorithms:

3-1 user-based collaborative filtering (UBCF)

The idea of user-based is simply finding the similarity between the active user and the neighbour or peer users who have the same preferences in the past. Then predict the unseen items to the active user according to the ratings of the same items by the similar users (Dietmar et al., 2011).

The matrix below is a made-up sample of rating matrix for 9 users and 5 movies used to illustrate the algorithm.

MovieID	1	2	3	4	5
UserID					
1	2	4	4	0	3
2	0	4	0	5	0
3	0	5	0	0	3
4	3	2	0	0	0
5	0	0	5	3	0
6	0	4	0	0	5
7	0	0	0	5	0
8	0	5	0	0	4
9	3	5	5	0	0

The ratings scale is 1 to 5 and 0 represents unrated items. To predict the rating of user 9 for movie 5, We can see there is a similarity between the user 9 {3,5,5} and the user 1 {2,4,4}.

To calculate the similarity between the users by using Pearson correlation coefficient which is calculated between two variables as the covariance of the two variables divided by the product of their standard deviations, (Gorakala, 2016). This is given by ρ (rho). The Pearson correlation coefficient takes values from +1 strong positive correlation and -1 strong negative correlation:

$$\rho_{xy} = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y}$$

$$\text{sim}(u, v) = \frac{(r1u - \mu_u) * (r1v - \mu_v) + (r2u - \mu_u) * (r2v - \mu_v) + (r3u - \mu_u) * (r3v - \mu_v)}{\sqrt{(r1u - \mu_u)^2 + (r2u - \mu_u)^2 + (r3u - \mu_u)^2} * \sqrt{(r1v - \mu_v)^2 + (r2v - \mu_v)^2 + (r3v - \mu_v)^2}}$$

$$\begin{aligned} \text{sim}(1, 9) &= \frac{(2-3.25)*(3-4.33) + (4-3.25)*(5-4.33) + (4-3.25)*(5-4.33)}{\sqrt{(-1.25)^2 + (0.75)^2 + (0.75)^2} * \sqrt{(-1.33)^2 + (0.67)^2 + (0.67)^2}} \\ &= 0.996 \end{aligned}$$

To find the rating of user 9 for the movie 5 we apply the following formula:

$$\begin{aligned} \hat{r}_{uj} &= \mu_u + \frac{\sum_{i=1}^l (r_{ij} - \mu_i) \cdot \text{correlation}_{ui}}{\sum_{i=1}^l |\text{correlation}_{ui}|} \\ &= 4.33 + \frac{(3-3.25)*0.996}{0.996} = 4.08 \end{aligned}$$

3-2 Enhanced user-based collaborative filtering with demographic features (EUBCF)

In this experiment, we considered the benefit of demographic features of the users and used it as enhanced factor. According to the Burke's taxonomy (Burke, 2002) this algorithm can be classified as a feature combination hybrid because of using features from different dataset and combining into a single recommender system (Vozalis et al., 2004).

In this hybrid algorithm, the base algorithm is user-based collaborative filtering and is enhanced with information extracted from the demographic data.

The demographic similarity represents the correlation between two users which is found by creating vectors for all users which contain information that is available in the user dataset like age, occupation and gender. Once the user demographic vectors are created, the demographic similarity is calculated by dot product between these vectors.

The enhanced correlation is obtained by multiplying the rating-based correlation and demographic correlation as additive factor to enhanced the rating-based correlation. To find the enhanced correlation between an active user (u) and a neighbour user (i) (Vozalis et al., 2004).

$$\text{Enh_cor}_{ui} = (\text{sim}_{ui} \times \text{dem_cor}_{ui}) + \text{sim}_{ui}$$

To predict the rating of unseen movie (j) for the user (u) with the enhanced correlation using the same formula that used in the user-based collaborative filtering. The only different is using enhanced correlation instead of rating-based correlation.

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{i=1}^l (r_{ij} - \mu_i) \cdot \text{Enhanced correlation}_{ui}}{\sum_{i=1}^l |\text{Enhanced correlation}_{ui}|}$$

3-3 Item-based collaborative filtering (IBCF)

The algorithm is a transpose of the user-based algorithm and is based on finding the similarity between the items and not between the users, based on the ratings. For example, if user A liked science-fiction movie in the past then user A will still like movies which have similar genre to the first movie (Gorakala, 2016). After finding the similarity between the items, the active user will receive recommendation of unseen items which are similar with the rated items in the past.

We will use the same matrix used in the previous algorithm which is a rating matrix for 9 users and 5 movies.

MovieID	1	2	3	4	5
UserID					
1	2	4	4	0	3
2	0	4	0	5	0
3	0	5	0	0	3
4	3	2	0	0	0
5	0	0	5	3	0
6	0	4	0	0	5
7	0	0	0	5	0
8	0	5	0	0	4
9	3	5	5	0	0

We need to predict the rating of user 2 for movie 5. When we compare the movie-id vectors we can see there is a similarity between the rating vectors of movie 5 {3,3, 5, 4} and movie 2 {4,5,4,5} (Dietmar et al., 2011).

To find the similarity, the cosine similarity method is used as a metric measure and it has been shown to produce accurate results (Gorakala, 2016). It represents the similarity by the cosine angle between two vectors. Cosine = 0 means the vectors are parallel and there is no relation between the vectors and when the value is near 1 this indicates high similarity and the vectors are perpendicular (Gorakala, 2016).

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

A and B are vectors, $\|A\|$ and $\|B\|$ are the square root of dot product for the vectors with itself.

To find the similarity between two movies; m1 with ratings (r1m1, r2m1, r3m1) and m2 with ratings (r1m2, r2m2, r3m2), (Charu, 2016).

$$\text{sim}(m5, m2) = \frac{r1m5 \cdot r1m2 + r2m5 \cdot r2m2 + r3m5 \cdot r3m2}{\sqrt{r1m5^2 + r2m5^2 + r3m5^2} \cdot \sqrt{r1m2^2 + r2m2^2 + r3m2^2}}$$

$$\text{sim}(m5, m2) = \frac{3 \cdot 4 + 3 \cdot 5 + 5 \cdot 4 + 4 \cdot 5}{\sqrt{3^2 + 3^2 + 5^2 + 4^2} \cdot \sqrt{4^2 + 5^2 + 4^2 + 5^2}} = 0.964$$

After finding the similarity we can find the prediction of rating movie 5 for the user 2 based on the formula below:

$$\hat{r}_{ut} = \frac{\sum_{j \in Q_{t(u)}} \text{Cosine}(j, t) \cdot r_{uj}}{\sum_{j \in Q_{t(u)}} \text{Cosine}(j, t)}$$

$$\text{Predict- rating (user 2, movie 5)} = \frac{4 \cdot 0.964}{0.964} = 4$$

In this example we compared the item 5 with only one movie but normally we compare the movie with all other movies and find the similarity between them.

3-4 Singular value decomposition based Collaborative filtering (SVD)

Using matrix factorization model-based in collaborative filtering is widely implemented in recommender systems as a way to solve scalability and sparsity problems better than memory methods.

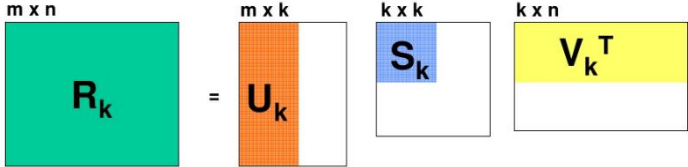
The idea of matrix factorization is restructuring the ratings matrix into two low rank matrices with less sparsity and from these two matrices we can discover the latent factors of users' preference and latent factor of movies' features (James, 2018).

With SVD each matrix can be decomposed into three parts $M = U \Sigma V^T$, where M is $(m \times n)$ an original matrix,

U is an $(m \times k)$ is column-orthonormal matrix and containing the eigenvectors of M .

V is an $(n \times k)$ is column-orthonormal matrix and it is always used in its transposed form, and V^T rows are orthonormal.

Σ is a $(k \times k)$ diagonal matrix. All elements not on the main diagonal are 0. The elements of Σ are called the singular values of M (Leskovec J. et al, 2014).



$$\begin{matrix} m \times n & & m \times k & & k \times k & & k \times n \\ \boxed{R_k} & = & \boxed{U_k} & \boxed{S_k} & \boxed{V_k^T} \\ & & & & & & \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix} & = & \begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix} & \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} & \begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix} \\ M & & U & \Sigma & V^T
 \end{matrix}$$

Figure 2 the form of a singular-value decomposition

(Leskovec J. et al, 2014)

To explain the technique and understand what are the three matrices represent, we need to concentrate on columns of these matrices which represent the features which are hidden in the main matrix M . The example, figure (2) above shows a ratings matrix M where the rows are the users and the columns are the movies. Looking at the main matrix M , we can notice that the first four people have rated specific movie genres. Whereas, the last three users rated other kind of movies.

The U matrix connect the users with the hidden features and in this case it is the genres of the movies for example, the first user gave low rating for some movies because of that we can see the first value in matrix U is 0.14 which is lower than the other values of the column. And second column is zero as he did not rate some other movies.

On the other hand, we can see that the matrix V connect the movies with the hidden feature. 0.58 values of the first row indicate that the first three columns of matrix M belong to specific genre. While, the 0.71 represent other genre for the last two columns in matrix M . The matrix Σ gives the strength of each hidden feature.

The strength of first genre of the movies is 12.4 larger than the strength of another genre 9.5 because the first genre has more information or it is rated more than the other one (Leskovec J. et al, 2014)

3-5 Evaluation Protocols

In this section, we present the evaluation measurements used to compare between the algorithms. The most common measurements of accuracy metrics based on historical data are: (Dietmar et al., 2011)

3-5-1 Accuracy of predictions:

The accuracy of predictions measures how well the system does to predict the real rating value for a specific item. The metrics of predicting accuracy are important for evaluating the prediction ratings that are recommend to the user and the system could fail if it does not succeed to predict the ratings correctly even when the rank of user's item list is correct (Herlocker et al., 2004).

The most popular method evaluates the system ability to predict the rating of users is mean absolute error MAE, which find the average deviation of the predicted ratings and the actual ratings for the users and items in the testing set (Dietmar et al., 2011).

$$MAE = \frac{\sum_{i=1}^n |\hat{r}_i - r_i|}{n}$$

Another accuracy method is mean square error MSE and root mean square error RMSE

$$MSE = \frac{\sum_{i=1}^n (\hat{r}_i - r_i)^2}{n} \quad RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{r}_i - r_i)^2}{n}}$$

Where

(r_i) is the real rating of item (i) and (\hat{r}_i) is the predicted rating of this item and (n) is the number of users in the test set.

When there are large error values, the RMSE is more affected than MAE and therefore, it is used for application when the robustness of prediction of various ratings is very important. And because of this we used it in the comparison of the algorithms. Whereas, the MAE is used when the importance of the outliers is limited in the evaluation (Charu, 2016).

3-5-2 Accuracy of classifications:

In recommender system the aim of classification task is to measure how many times the system makes correct or incorrect decision about recommend the items or not and identify the most relevant items for a user. Classification accuracy has a problem with sparsity when evaluating the top recommended items which have not been rated. To avoid this problem one approach can be taken is ignoring the recommendation for items for which there are no actual ratings and the recommendation task has been modified to predict the top recommended items that have been rated. (Herlocker et al., 2004).

The most used measurements for classification matrices and the quality of information retrieval tasks are Precision and recall. (Dietmar et al., 2011). They have been used by Billsus and Pazzani to evaluate recommender systems (Herlocker et al., 2004).

The precision matrix (P) is the number of hits divided by the total number of recommended items (proportion of good items returned). While, the Recall (R) relates the number of hits to the number of test dataset (proportion of everything that is recovered) (Andrews, 2014).

When evaluating recommender systems normally the focus is on recommending the top-n items to the users so that it is make sense to calculate precision and recall at k which represented the number of items recommended instead of all the items.

The precision and recall are binary metrics so we need to translate the rating scales (from 1 to 5) into binary scale (relevant and not relevant items) as shown in table (1).

	Relevant	Irrelevant	Total
Recommended	TP	FP	TP+FP
Not Recommended	FN	TN	FN+TN
Total	TP+FN	FP+TN	N

Table 1 Confusion table of recommender system

For example, we assume that in the test dataset the actual rating for an item over 3.5 is relevant item and actual rating under 3.5 is irrelevant when 3.5 is threshold value or is the cut-off between good and bad. The recommended item is the item which has a predicted rating ≥ 3.5 and the not recommended item is the item which has a predicted rating < 3.5 . (Malaeb, 2017).

Precision, recall and fallout can be calculated when the number of recommended items is (K) as following:

$$\text{Precision@k} = \frac{\text{number of recommended item that are relevant (TP)}}{\text{number of recommended item at(k)(TP+FP)}}$$

$$\text{Recall@k (true positive rate)} = \frac{\text{number of recommended item that are relevant (TP)}}{\text{total of relevant item(TP+FN)}}$$

$$\text{Fallout@k (false positive rate)} = \frac{\text{number of recommended item that are irrelevant (FP)}}{\text{total of irrelevant item (FP+TN)}}$$

The mean precision and mean recall are simply the average of all users' precision and recall at top-n recommendation list (Marginalia, 2017).

$$\text{Mean P@k} = \frac{\sum_{i=1}^U P_{i@k}}{U}$$

$$\text{Mean R@k} = \frac{\sum_{i=1}^U R_{i@k}}{U}$$

When U is the total number of users.

Another measurement of the performance of a recommender system is the ROC curve which provides a graphical representation and plots Recall (true positive rate) with the Fallout (false positive rate). The recommender system is perfect when the curve goes straight up to 1.0 true positive rate and 0.0 false positive rate until all relevant items retrieved and it goes straight right towards 1.0 false positive rate as shown in Figure (3). The dash line in the figure represent random guessing and the AUC =0.5 (Liu, Y., 2017) and the blue area represent the area under the curve (AUC).

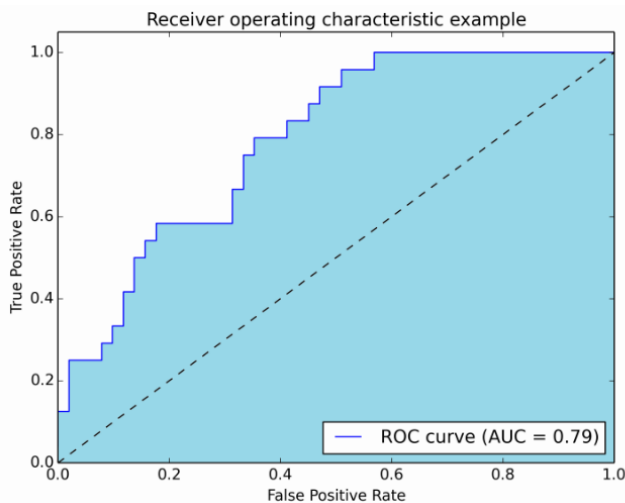


Figure 3 Receiver operating characteristic (ROC) curve
(Stack exchange, 2018)

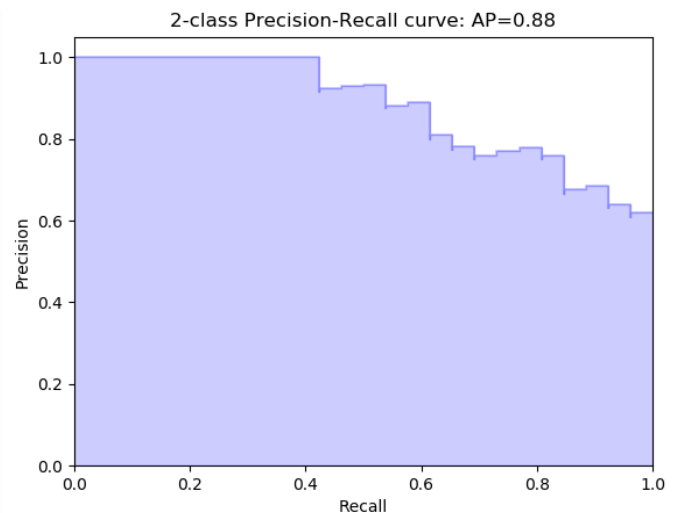


Figure 4 Precision-Recall curve
(Scikitlearn.org, 2019)

In Precision-Recall curve figure (4) the prefect system is when the curve goes from 1.0 precision and 0.0 recall and then it goes straight to the right towards 1.0 precision and 1.0 recall then drops downwards to 0.0 precision and 1.0 recall.

The goal of this performance measurement is to maximize this area and it is used as the overall measure of the quality of the recommender system (Schroder et al., 2011).

The decision on whether Precision-Recall curve or ROC curve is used can be based on two factors; the domain of the application and its goal. The Precision-Recall describes the ratio of the recommendation items were suitable to the users. When the goal is to recommend list of items to a user, then small or large ratio of the recommendation items were unsuitable (false positive) could be recommended is not as important as the ratio of the recommendation items that were relevant to the users. So that Precision-Recall curve is more suitable for this application.

The ROC is more suitable for application when the recommendations incur a cost, where the goal is to maximise the suitable recommendations while minimising the cost (Shani G et al., 2011).

It is recommended that both are applied and by inspecting the curves the more suitable one will become easier to work out (Shani G et al., 2011). Hence, we have followed this advice and both graphs were plotted.

3-5-3 Rank accuracy:

We considered the following three rank accuracy measurements:

Mean Average Precision (MAP):

When recommending N number of items to a user, it is better to submit all the relevant items which the user certainly likes. But when submitting the recommendation list with irrelevant items, it is better to recommend the more relevant items first followed by the items that are irrelevant so that the rank of the list is important to be evaluated (Zygmunt, 2012).

In this measure we calculated the precision at every position in the ranked recommendation list of items and took the average of precision for all precisions in the list as shown below (Sawtelle, 2016).

$$AP = \frac{\sum_{k=1}^n (P(k) * relvant(k))}{number\ of\ relevant}$$

Where P(k) is the precision in k position of the recommended list and relevant(k) is represented as 1 if the kth item is relevant and 0 if irrelevant.

For example, when user1 receives recommended items like prediction {2,4, 6} and the actual relevant items for that user are Labels= {1,3,4,5,6} and user2 has prediction {6,3, 2} and the actual relevant items for that user are Labels= {1,3,4,5,6} we calculate the mean average precision as explained below.

The relevant list is represented as binary list 1 when the recommended item is in the actual list and 0 if not. We can see that when the relevant items are ordered first in user2 the average precision is 1 which is the prefect case (Marginalia, 2017).

	P@1	P@2	p@3	AP@3
User1 {0,1,1}	0	1/2	2/3	$(0+1/2 + 2/3)/2 = 0.583$
User2{1,1,0}	1	2/2	2/3	$(1 + 2/2+ 0)/2 = 1$
				Mean average precision = $(0.583 + 1)/2 = 0.791$

Another rank evaluation method is Normalized Discounted Cumulative Gain (NDCG) which is used to measure the quality of rank list in information retrieval using graded relevance scale of the items in the list. When items are placed in the tail of the list, the important effect of these items has less influence (Postmus S, 2018).

To define the NDCG, we first should find the discounted cumulative gain (DCG). When user (i) has a list of relevant (u) with length of k then the DCG for that user is calculated as (Marginalia, 2017):

$$DCGi@k = \sum_{j=1}^k \frac{2^u - 1}{\ln(i+1)}$$

Following that we need to find the ideal discounted cumulative gain (IDCG) which uses this time a sorted relevant (u) list and represents the perfect case when the list is ranked as below:

$$IDCGi@k = \sum_{j=1}^k \frac{2^u - 1}{\ln(i+1)}$$

Then the NDCG is calculated as following

$$NDCG = \frac{DCGi@k}{IDCGi@k}$$

The last evaluation technique is Coverage which is the percentage of items in the system that can be used in predictions. If it is low in value then the system is not very useful to the users due to some items are not being recommended. This is particularly useful for tasks like “Find All Good Items” used when enhancing exact word match searches. When used on web commercial sites then it is called catalog coverage measure as evaluates the percentage of the catalog being recommended to the users (Herlocker et al., 2004).

This measure should be used with accuracy measurement else it can be misleading, example if items in the catalog are not interesting to all of the system users and the recommender system does not recommend it then the coverage is low while accuracy is high (Herlocker et al., 2004).

Let us denote I is the set of all items that available in the dataset (the catalog) and I^j is the set of items in the L list. We can find the catalog coverage by the following (Mouzhi et al., 2010):

$$Catalog\ coverage = \frac{|\bigcup_{j=1..N} I_L^j|}{|I|}$$

It is expected that the coverage values are increased when the recommendation list (N) is increased until coverage values become stable without affect in increasing N .

4 Comparison and Evaluation of the Experiments

4-1 The Datasets

For this project we used two datasets:

The first is MovieLens, a movie ratings dataset which contains 1,000,209 ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000.

There are three tables in the dataset (Grouplens.org ,2019).

1- Movie file contains the movie information in the following fields:

- movieid.
- title, the title of movie including the year of publishing.
- genre, there are 18 genres and some movies have two or three genres.

2- User file has demographic information about each user:

- Gender, is denoted by a "M" for male and "F" for female.
- Age, is range from under 18 to 55.
- Occupation, there are 21 types of occupation and represented as an integer for each occupation

3- Rating file, the user ratings of the movies

- UserIDs range between 1 and 6040
- MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only, hence 1 to 5)
- Timestamp is represented in seconds – not used in the project.
- Each user has at least 20 ratings

More details about the dataset can be found in Appendix C.

The second dataset is Book-Crossing, a book ratings dataset, which contains 278,858 users providing 1,149,780 ratings of about 271,379 books. The dataset was collected in September 2004 from Book-Crossing community.

This dataset has three main tables of data (Informatik.uni-freiburg.de, 2004).

1- BX-Books file contains the books information in the following fields:

- ISBN
- Four field about book information which are Book title, Book-Author, Year-Of-Publication, Publisher.
- URLs linking to cover images

2- BX-Users file has demographic information about each user:

- Age, it contains null if the age not available.
- Location, it contains null if the age not available.

3- BX-Book-Ratings, the user ratings of the movies

- UserIDs
- ISBN, there are rated books not found in BX-Books tables.
- Ratings which are either explicit, expressed on a scale from 1-10 or implicit, expressed by 0.

More details about the dataset can be found in Appendix D.

4-1-1 Data Cleansing

MovieLens dataset did not require any cleansing. However some cleansing was performed to the Book-Crossing dataset.

The book ratings where the books are missing from the book table were removed. For implementing the algorithms, we used the explicit ratings only and for statistical significance we excluded the books that have less than 10 explicit ratings and the users who rated less than 20 books. The final number of ratings is 48,718 ratings.

4-1-2 Statistical Analysis

The MovieLens matrix of the rating dataset has 4.47 % density of ratings which makes the matrix having high sparsity of 95.53%.

The Book-Crossing matrix of explicit rating has density of 0.004 % which makes the sparsity about 99.99%.

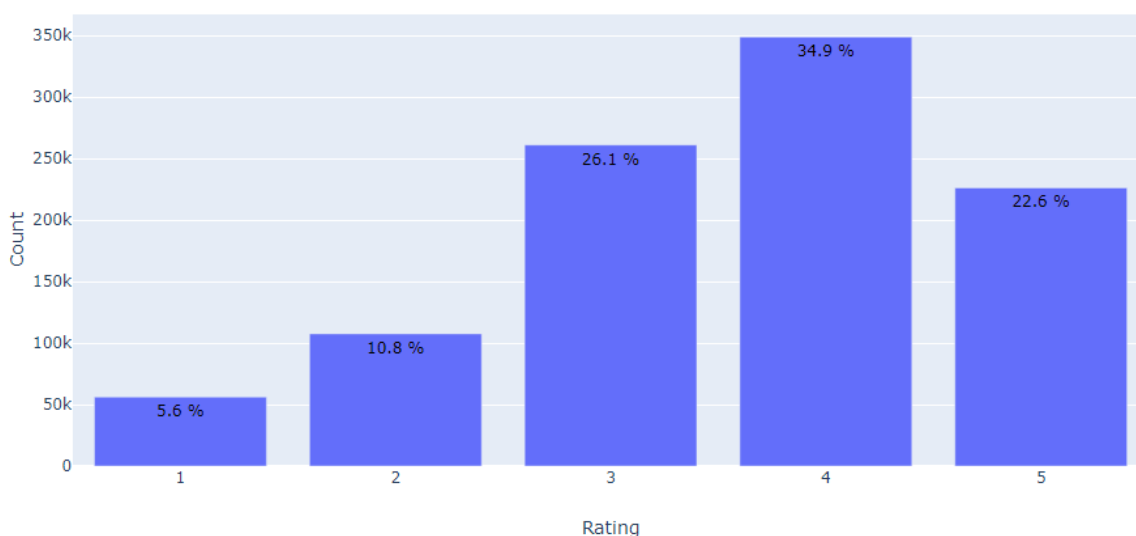


Figure 5 1-5 Range of ratings distribution for ML-1m

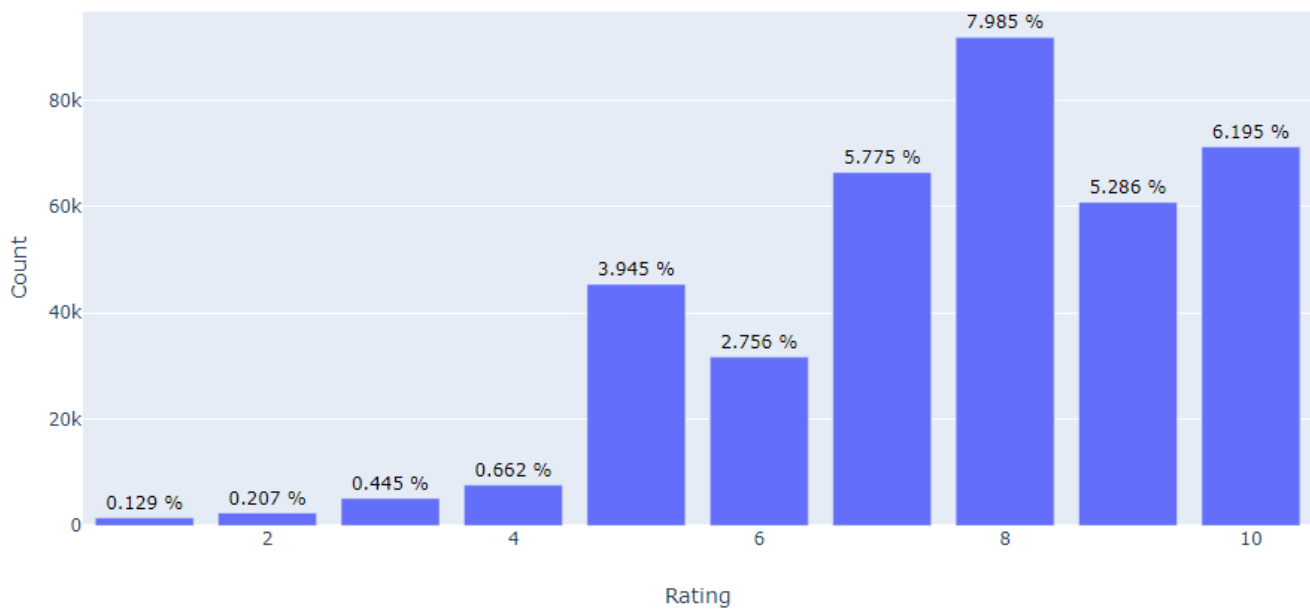
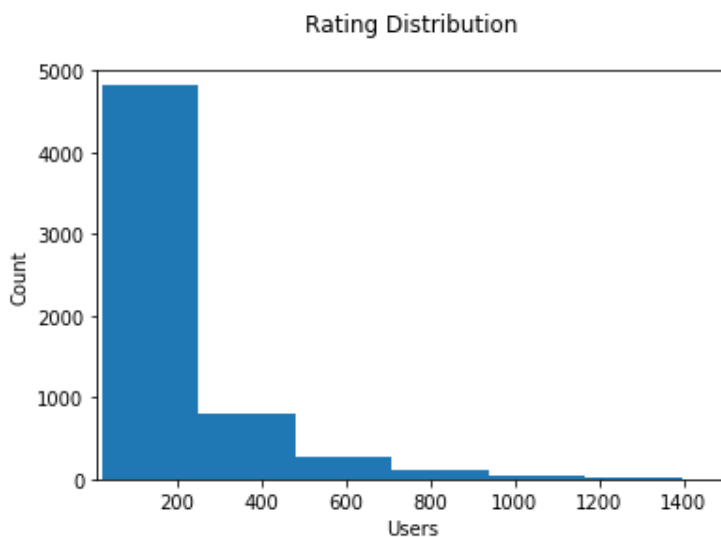


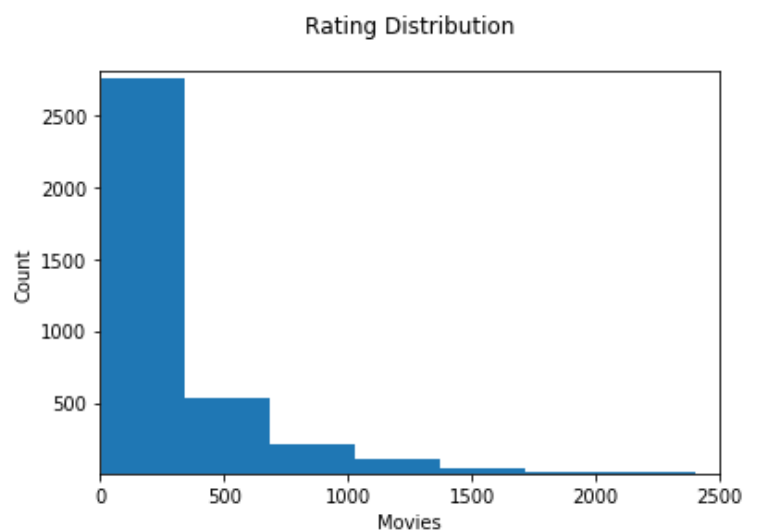
Figure 6 1-10 Range of explicit ratings distribution for Book-Crossing

From the figure (5), we can notice that there is a considerable skew in the ratings distribution for higher ratings 3 and 4. Over half of the movies have ratings of 4 or 5 and just over a quarter of the movies with a rating of 3. The users gave ratings of 2 to 10% of the total, which was twice as many as rating of 1 to 5% of the total.

In figure (6), the explicit ratings distribution for Book-Crossing shows a skew for higher ratings which are more common amongst users and rating 8 has been rated the greatest number of times. The Implicit ratings are not represented in figure 6 as not used, and these add up to about 60%.



a) Distribution of ratings per user



b) Distribution of ratings per movie

Figure 7 distribution for ML-1m

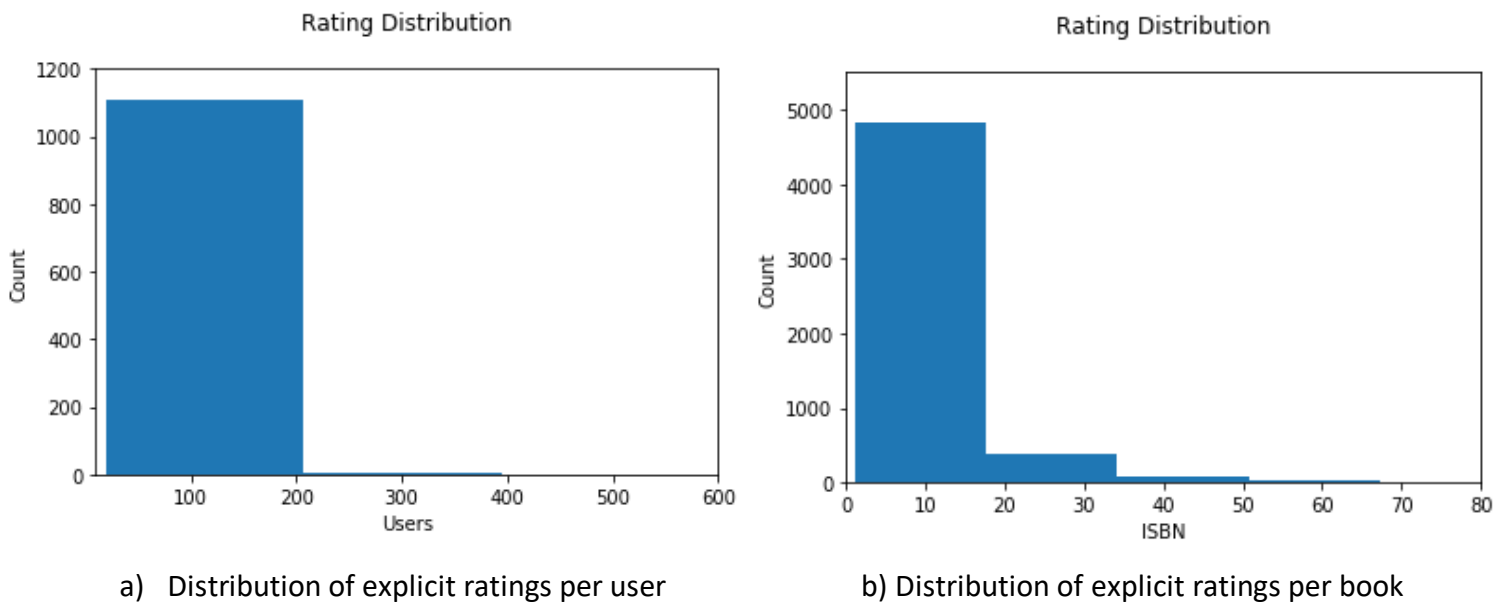


Figure 8 distribution for Book-Crossing

The figures 7a and 7b illustrate the distribution of ratings per user and per movie. It can be seen that both figures have the same distribution and most of the users have less than 500 ratings and few of users who have ratings above 1000. Similarly, with the movies' dataset, majority of movies have less 200 ratings and few of them have above 500 ratings.

For Book-crossing figure 8a shows that number of users have less than 20 explicit ratings and in 8b illustrates that most books have around 300 explicit ratings and less.

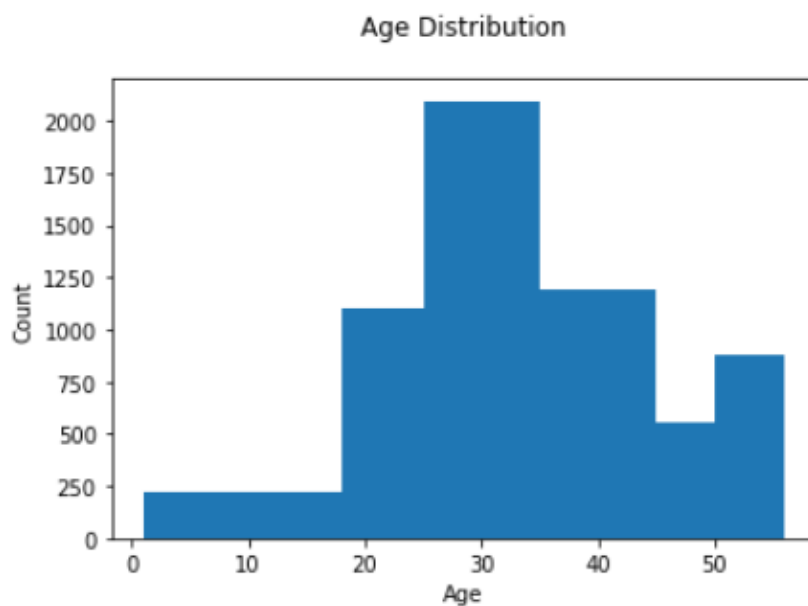


Figure 9 Age of user's distribution for ML-1m

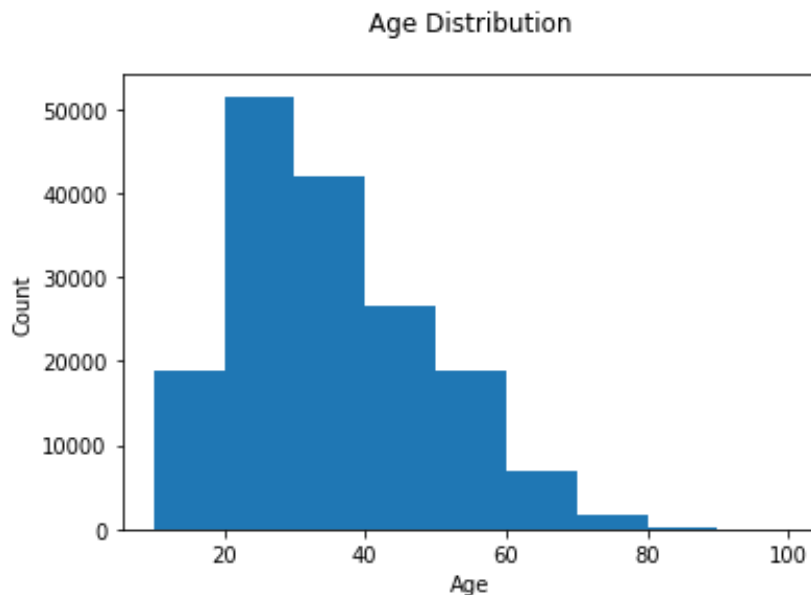


Figure 10 Age of user's distribution for Book-Crossing

We can note that the majority of the active users in ML-1m are in their 25 – 35s and in their 20 - 40s for book-Crossing datasets as demonstrated in figures 9 and 10.

4-2 creating training and testing set

In typical machine learning applications, the testing set is completely separated from training set and the testing set is used when we need to test how well this model is able to predict new data unseen in training set. In recommender systems the procedure is different because we need all of the user/items ratings in both training and testing sets to build the model. Hence, we hid a specific percentage of the ratings from the training set selected randomly and act as if the users never rated them. In the testing set, we unhide the hidden ratings and find out how well the system is able to predict them and if the recommended items are actually relevant for the users (Steinweg-Woods, 2016). As shown in figure (11)

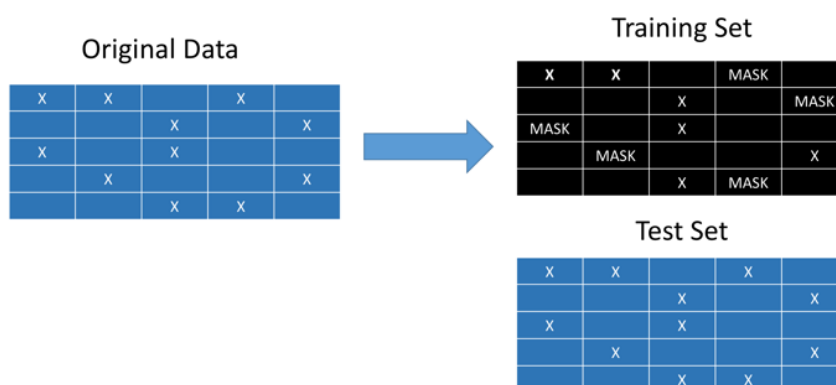


Figure 11 Training and testing set in recommender system
(Steinweg-Woods, 2016)

For ML-1m dataset we split the ratings data set into 70% for training which has 700,146 ratings for 6,040 users on 3,660 movies and each user has at least 14 ratings. The remainder 30% ratings used for testing has 300,063 ratings for 6,040 users and 3,548 movies and each user has at least 6 ratings. We used the training set to fit the algorithms after replacing the ratings of the testing set with nan values.

For Book-Crossing dataset, we implemented the algorithms with explicit ratings only which is split into 70% training which has 34,102 ratings for 1,117 users on 5,247 books and each user has at least 14 ratings. 30% ratings for testing set has 14,616 ratings for 1,117 users on 4,515 books and each user has at least 6 ratings. We hid the ratings of the testing set from the training set and then implement the algorithms.

4-3 Experiments implementation

We have implemented four experiments using different techniques and evaluated these algorithms to compare the results. In all four experiments we used the same following stages to build the system but different techniques according to the algorithm that were used. The main stages are:

After implementing the algorithms and finding the predicted ratings for all users as it will be explained in following sections, we created the top-n recommendation list for a specific user by extracting the predicted ratings which were unseen by the user and ranked them in a descending order.

In the evaluation stage, we created data frame for all pairs of users and movies in the test set with the real and predicted values of ratings. For all four experiments we calculated the MAE and RMSE using the methods (`metrics.mean_absolute_error`) and (`np.sqrt(metrics.mean_squared_error)`) respectively.

Finding the mean precision ($P@k$) and mean recall ($R@k$) by using different top-n recommendation list where n was of the following values: 3, 5, 10, 15, 20, 25. The threshold to classify the items as recommended or not and relevant or not was set at 3.5 in ML-1m dataset and 6 in Book-Crossing dataset. For each top-n of recommended lists, we found the precision and recall for each user using the formulas which are explained in section (3-5-2), then we calculated the mean precisions and recalls of the system by taking the average of the precision and recall of all users.

We repeated the above to find the true positive value (tp) and false positive value (fp) for each user and computed the mean of the (true positive) and (false positive) for each experiment.

Regarding the Mean Average Precision ($MAP@K$) and Mean Average Recall ($MAR@K$), we used (`ml_metrics.mapk`) and (`recmetrics.mark`) methods from `ml_metrics` and `recmetrics` libraries which take three parameters.

The first parameter is a list of lists of relevant items. The second parameter is a list of lists of recommended items for all users. The third parameter is the number of recommendation list.

For the Normalized Discounted Cumulative Gain (NDCG). We created a list of lists which has binary values 1 if an item is in the relevant list and in the recommended list and the value 0 if it is not. Using a function to find the NDCG for different top-n recommendation list where n in [3, 5, 10, 15, 20, 25] for the all techniques.

Finally, the catalog coverage for the algorithms is calculated by using (`recmetrics.coverage`) method which has two parameters. The first one is the list of lists of recommended items for all users and the

second parameter is list of movies that are available in the ratings dataset. The function is implemented with the same range of top-n recommended movies where n in [3, 5, 10, 15, 20, 25] for the four algorithms.

For the complete source code of the four experiments implementation and evaluation can be found on the attached USB flash drive with details in appendix H and G.

4-3-1 user-based collaborative filtering (UBCF)

The first method we implemented is user-based collaborative filtering. To start, we needed to create user-movie training and testing matrixes and found the mean ratings in the training set for each user. Then found the similarity matrix between each user with all other users in train matrix using (pairwise.cosine_distances) function.

After that, used the predict ratings formula which is explained in section (3-1) using (dot product) function between the similarity matrix and the training matrix of ratings to get a matrix of predicted ratings. The algorithm implementation in appendix H and G.

4-3-2 Enhanced User-based with demographic features (EUBCF)

To execute this technique, we had the similarity matrix between the users based on the rating activity from previous experiment. For ML-1m dataset we used the users' dataset to extract the demographic features and find the similarity between the users based on gender, age and occupation by creating a binary matrix with 29 columns as explained in the table (2).

1-2	Gender	Male 1 Female 0	
3-9	Age	< = 18 18 < Age <= 24 25 < Age <= 34 35 < Age <= 44 45 < Age <= 49 50 < Age <= 55 > 56	When the user's age within one group then it is represented by 1 and other age slot are zeros.
10-29	Occupations	We have 19 different occupations.	1 represent the user's occupation and the rest slot of other occupations are zeros.

Table 2 Structure of user demographic vector for ML-1m

With Book-Crossing dataset some users have missing age value and these are denoted with nan. Also, users who are older than 90 years and younger than 10 are considered outliers and have their age replaced with nan. Afterwards the nan values were replaced by the mean of the users age.

The age is binned into 8 groups as represented in table (3) below. Once the age features represented in binary matrix, we found the similarity between the users based on demographic features.

		bins	labels	
8 columns	Age	10 < Age <= 20	15	When the user's label age within one group then it is represented by 1 and other age slot are zeros.
		20 < Age <= 30	25	
		30 < Age <= 40	35	
		40 < Age <= 50	45	
		50 < Age <= 60	55	
		60 < Age <= 70	65	
		70 < Age <= 80	75	
		80 < Age <= 90	85	

Table 3 Structure of user demographic vector for Book-Crossing

After creating the demographic vector for the users, we used the (cosine_similarity) function to find the similarity matrix which was used as enhanced factor to the similarity rating matrix by multiplying them together and then added to the similarity rating to create enhanced similarity matrix which was applied to find the predicted ratings as we did in UBCF experiment. The algorithm implementation in appendix H and G.

4-3-3 Item-based collaborative filtering (IBCF)

We implemented this technique by creating user-movie train and test matrixes. Then found the similarity matrix between each movie and each book with all other movies and books in the training matrix using (correlation pairwise_distances) function. After that, used the predict ratings formula which was explained in section (3-2), using (dot product) function between the similarity matrix and the training matrix of ratings to get a matrix of predicted ratings. The algorithm implementation in appendix H and G.

4-3-4 Single value decomposition (SVD)

In this experiment, after generating the train and test matrix, we normalized the ratings in training set for each user by taking the mean of users' ratings and then for each user subtracted the mean from each rating.

We used (SciPy function (svds)) because this function allows the user to select how many latent factors are used. In order to tune the latent factors, we implemented grid search hyper parameter which is available in (Surprise library) using (GridSearch) function and specified the grid parameter with number of factors in range of [50, 100, 150, 200, 250, 500].

We got the best results for the ML-1m dataset when the number of factors is 50 with least error values of 0.892 RMSE and 0.703 MAE. For Book-Crossing dataset, the best result was when the number of factors is 200 with minimum errors in 3.394 RMSE and 3.025 MAE.

When we applied the function svds and the normalized ratings matrix. This function returned three matrices which were used to predict the ratings by multiplying these three matrices by using dot function then the result matrix is added to the mean ratings for each user.

The error results for training matrix were different from the result we got from the surprise library because function SVD () in this library has other parameters which affect the errors result however in our experiment we only focused on the factor number. The algorithm implementation in appendix H and G.

4-4 Testing the prediction implementation code

Due to the importance of the prediction implementation code as pivotal for the rest of the code we have provided the manual testing that was done in appendixes E, F and I.

The testing was performed for the prediction ratings in user-based and item-based collaborative filtering algorithms and SVD technique was implemented using (svds) function and EUBCF is the same prediction formula as UBCF.

The manual testing was performed using small made up sample of data containing 5 users and 5 movies and has 15 ratings where each user has 3 ratings. The testing entailed manually calculating the predicted ratings using the formula in sections (3-1 and 3-3) and comparing the results with the implementation results.

4-5 Results and techniques Comparison

In this section we compare the performance of the four recommender system techniques and discuss their evaluation results

Algorithms	Train		Test	
	MAE	RMSE	MAE	RMSE
UBCF	0.699	0.889	0.728	0.925
EUBCF	0.690	0.878	0.727	0.924
IBCF	0.720	0.904	0.783	0.983
SVD	0.676	0.866	0.774	0.975

Table 4 The prediction accuracy results for ML-1m

Algorithms	Train		Test	
	MAE	RMSE	MAE	RMSE
UBCF	0.347	0.515	1.745	2.709
EUBCF	0.274	0.422	1.750	2.714
IBCF	0.868	1.155	5.106	6.586
SVD	0.569	0.857	1.145	1.510

Table 5 The prediction accuracy results for Book-Crossing

Table (4) shows the MAE and RMSE results for training and testing sets for the four algorithms in dataset ML-1m. It can be seen that the SVD algorithm made the smallest errors in training set with 0.676 and 0.866 for MAE and RMSE respectively and EUBCF achieved lowest results in testing set in both MAE and RMSE errors with 0.727 and 0.924. In terms of the error achieved by IBCF, it was highest in training and testing and in both types of error which makes item-based the worst performing among the other techniques. Another interesting point is that EUBCF shows better performance in the testing set compared with UBCF.

For Book-Crossing dataset, table (5) shows different results of the lowest and highest errors compared with previous dataset where SVD has the best prediction accuracy in the testing set with 1.145 and 1.510 for MAE and RMSE respectively and EUBCF has the lowest errors in the training set. IBCF technique has been the worst algorithm in prediction accuracy and EUBCF has not shown advances above UBCF in the testing set.

Top-n list	Mean precision P@k	mean recall R@k (True positive rate)	False positive rate
3	0.838	0.189	0.063
5	0.807	0.290	0.115
10	0.718	0.451	0.243
15	0.641	0.541	0.324
20	0.579	0.599	0.377
25	0.528	0.639	0.416
ML-1m UBCF			

Top-n list	Mean precision P@k	mean recall R@k (True positive rate)	False positive rate
3	0.839	0.189	0.062
5	0.808	0.291	0.115
10	0.719	0.452	0.244
15	0.642	0.542	0.324
20	0.580	0.600	0.377
25	0.528	0.640	0.417
ML-1m EUBCF			

Top-n list	Mean precision P@k	mean recall R@k (True positive rate)	False positive rate
3	0.839	0.163	0.065
5	0.814	0.257	0.119
10	0.736	0.416	0.254
15	0.663	0.506	0.341
20	0.601	0.564	0.404
25	0.550	0.605	0.451
ML-1m IBCF			

Top-n list	Mean precision P@k	mean recall R@k (True positive rate)	False positive rate
3	0.837	0.159	0.062
5	0.800	0.248	0.124
10	0.723	0.410	0.270
15	0.650	0.501	0.364
20	0.589	0.561	0.427
25	0.538	0.602	0.474
ML-1m SVD			

Table 6 Confusion tables result of the experiments in ML-1m

Table 6 compares confusion tables' result for ML-1m dataset of the four experiments. We considered for all the techniques different pre-determined number of recommendations which are 3, 5, 10, 15, 20 and 25.

Overall, what stands out from the tables is that for all algorithms the mean precision decreased with increasing in the length of top-n recommendation list. The opposite is observed for the mean recall which has increased (improved) with increasing the number of recommendation list. From the tables result we are able to plot the ROC curve and a Precision-Recall curve as shown in the figures 12 and 13.

Looking at the details, as Precision-Recall curve is close to the upper right corner indicating better accuracy. For ML-1m dataset (figure 12) below depicts EUBCF has the highest accuracy with maximum precision and recall are 0.839 and 0.640 respectively. The difference between UBCF and EUBCF algorithms is negligible. SVD has achieved the lowest accuracy with maximum precision and recall are 0.837 and 0.602 respectively.

The closer the ROC curve to the upper left corner the better the accuracy. In figure (13) it could be seen that UBCF and EUBCF have better performance than IBCF and SVD techniques.

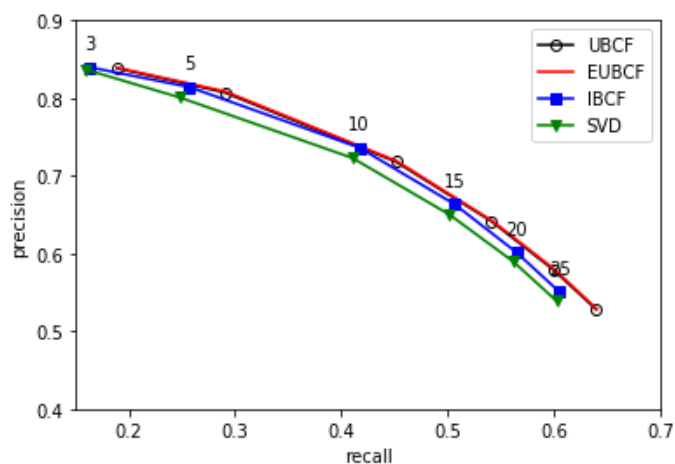


Figure 12 ML-1m Precision-Recall curve

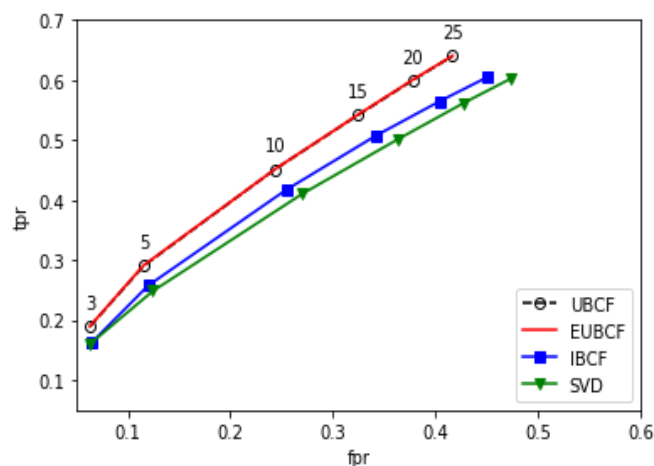


Figure 13 ML-1m ROC curve

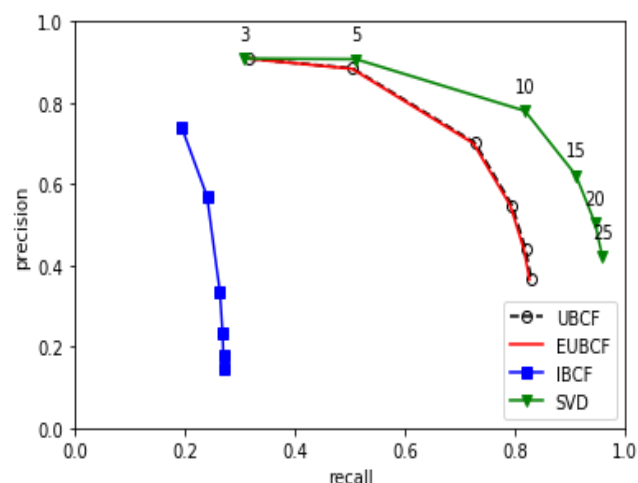


Figure 14 Book-Crossing Precision-Recall curve

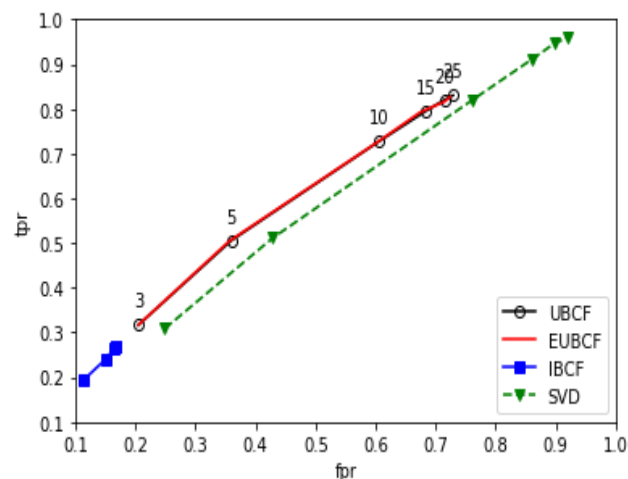
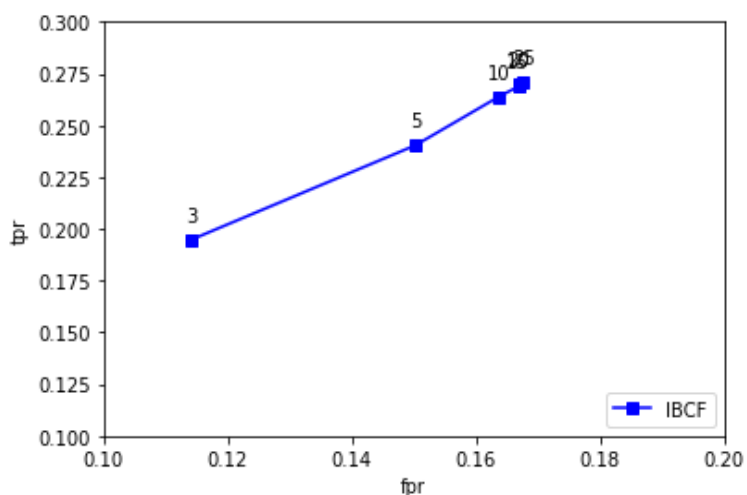


Figure 15 Book-Crossing ROC curve

Figure 16 Enlarged ROC curve of the Book-Crossing for IBCF algorithm



Top-n list	Mean precision P@k	mean recall R@k (True positive rate)	False positive rate
3	0.908	0.316	0.204
5	0.884	0.505	0.359
10	0.701	0.727	0.606
15	0.548	0.795	0.683
20	0.441	0.819	0.714
25	0.365	0.830	0.728
Book-Crossing UBCF			

Top-n list	Mean precision P@k	mean recall R@k (True positive rate)	False positive rate
3	0.908	0.317	0.204
5	0.883	0.504	0.356
10	0.700	0.724	0.601
15	0.546	0.791	0.673
20	0.439	0.816	0.715
25	0.364	0.826	0.724
Book-Crossing EUBCF			

Top-n list	Mean precision P@k	mean recall R@k (True positive rate)	False positive rate
3	0.737	0.194	0.114
5	0.570	0.240	0.150
10	0.335	0.263	0.163
15	0.234	0.268	0.166
20	0.178	0.270	0.166
25	0.144	0.270	0.167
Book-Crossing IBCF			

Top-n list	Mean precision P@k	mean recall R@k (True positive rate)	False positive rate
3	0.908	0.308	0.248
5	0.907	0.511	0.428
10	0.780	0.818	0.759
15	0.621	0.911	0.860
20	0.505	0.945	0.897
25	0.420	0.959	0.920
Book-Crossing SVD			

Table 7 Confusion tables result of the experiments in Book-Crossing

With regards to results of the Book-Crossing dataset, it can be seen from table (7) that the mean precision scores have decreased while the mean recall have increased when the number of recommended books has gone up. Figure (14) illustrates that a superior performance is obtained by SVD algorithm which was able to recommend books which are more interesting to the users as the curve is the nearest to the top right corner with maximum precision and recall 0.908 and 0.959 respectively. IBCF has the lowest precision among the other techniques. EUBCF has not shown better result than UBCF algorithm.

ROC curve in figure (15) shows that IBCF technique has the same ratio of the unsuitable books in the recommended list when the top-recommended books is above 10 and this is possibly due to the shortage in the number of rated books for each user in the testing dataset. Figure (16) shows an enlarged copy of the IBCF curve.

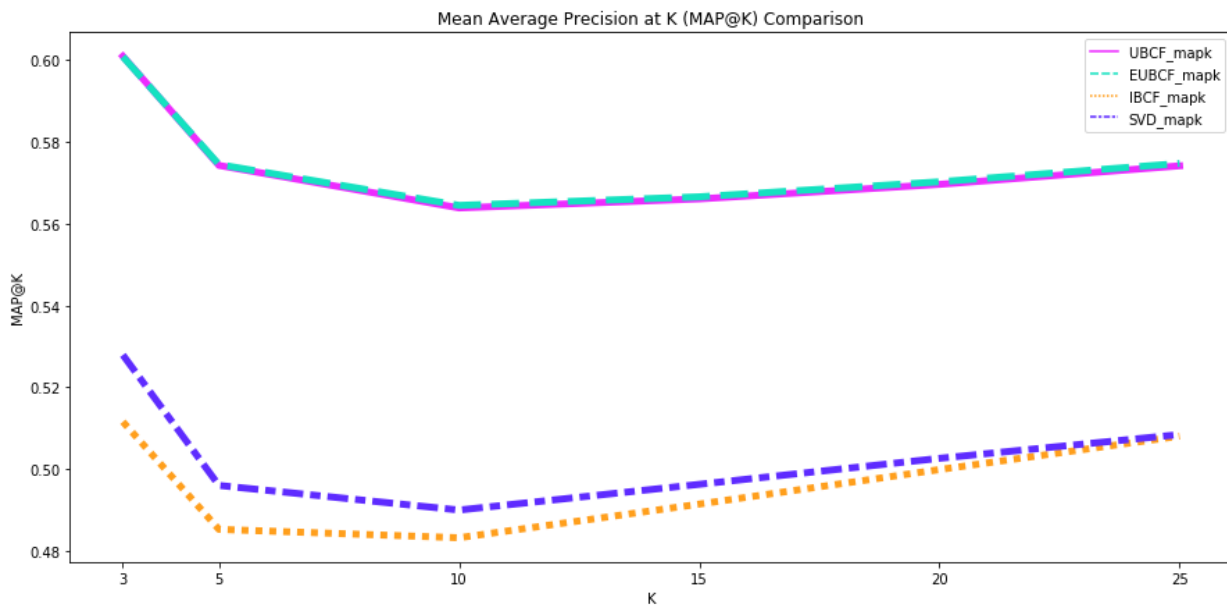


Figure 17 Mean average precision at K comparison for ML-1m

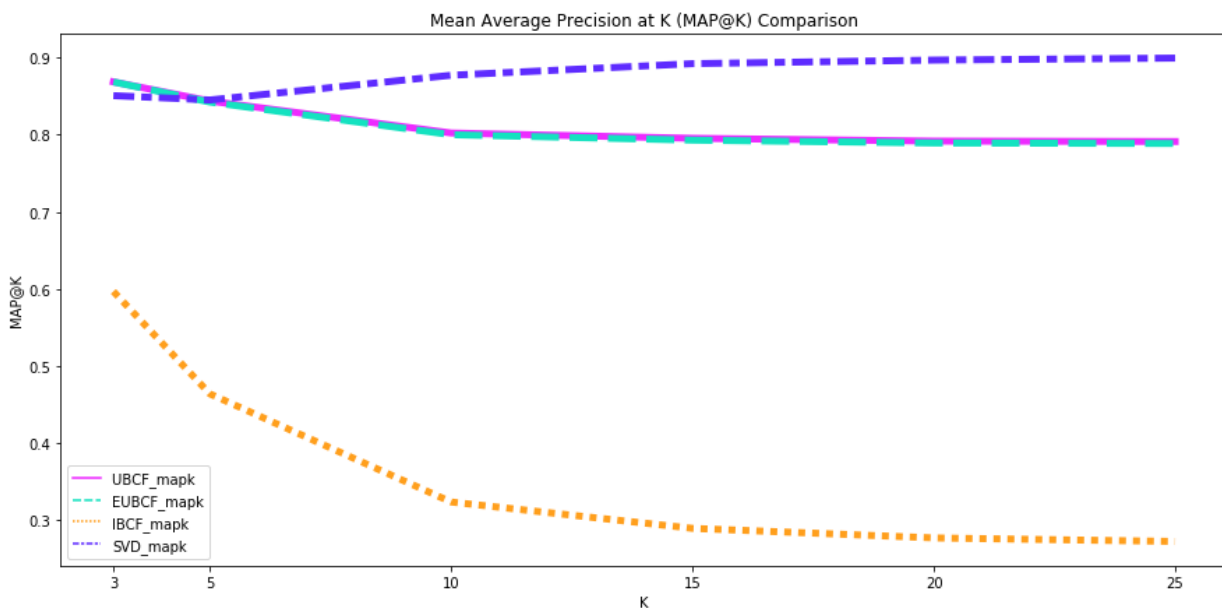


Figure 18 Mean average precision at K comparison for Book-Crossing

Figures (17, 18) above illustrate the mean average precision (MAP@K) for the four algorithms with number of recommendation list (k) (3, 5, 10, 15, 20, 25).

With respect of ML-1m dataset in figure (17), the mean average precision for all algorithms has decreased slightly with increasing in the length of top-n recommendation list and then increased gradually after top-n is 10.

EUBCF and UBCF algorithms have been able to achieve the highest MAP and recommend 60% of the relevant movies in the top of recommendation list when send top-3 movie. And when k=25 mean average precision has reached 0.57.

IBCF has the worst technique to recommend the relevant movies in the first rank of the list with 0.51 and 0.50 MAP when top-n list is 5 and 25 respectively. For SVD techniques, around 52% of the recommended movies which are relevant are ranked first to the users who received top-3 movies and when top-25 movies are recommended, SVD achieved 0.50 MAP.

For Book-Crossing dataset in figure (18), SVD algorithm has improved slightly when increased the recommended movies and the best result among the other techniques with MAP of around 0.85 and 0.89 when k = 3 and 25 respectively.

UBCF, EUBCF and IBCF algorithms have decreased gradually though all the top-n recommended movies list. For IBCF technique, the curve has been the lowest which makes the technique worst among the others. EUBCF has achieved results a little lower than UBCF algorithm.

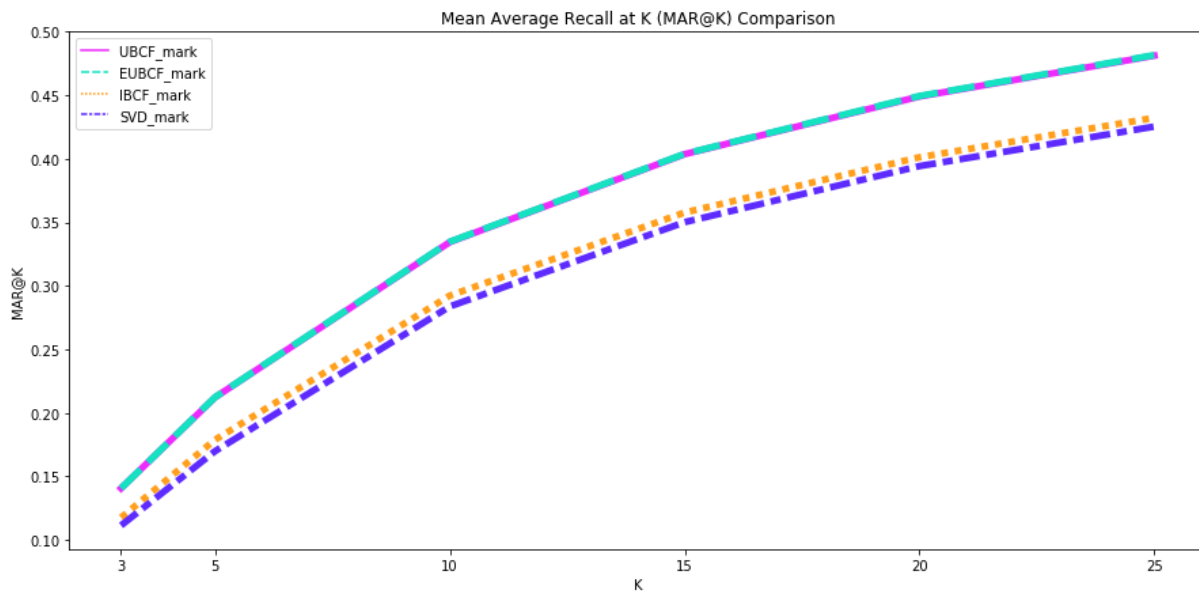


Figure 19 Mean average Recall at K comparison for ML-1m

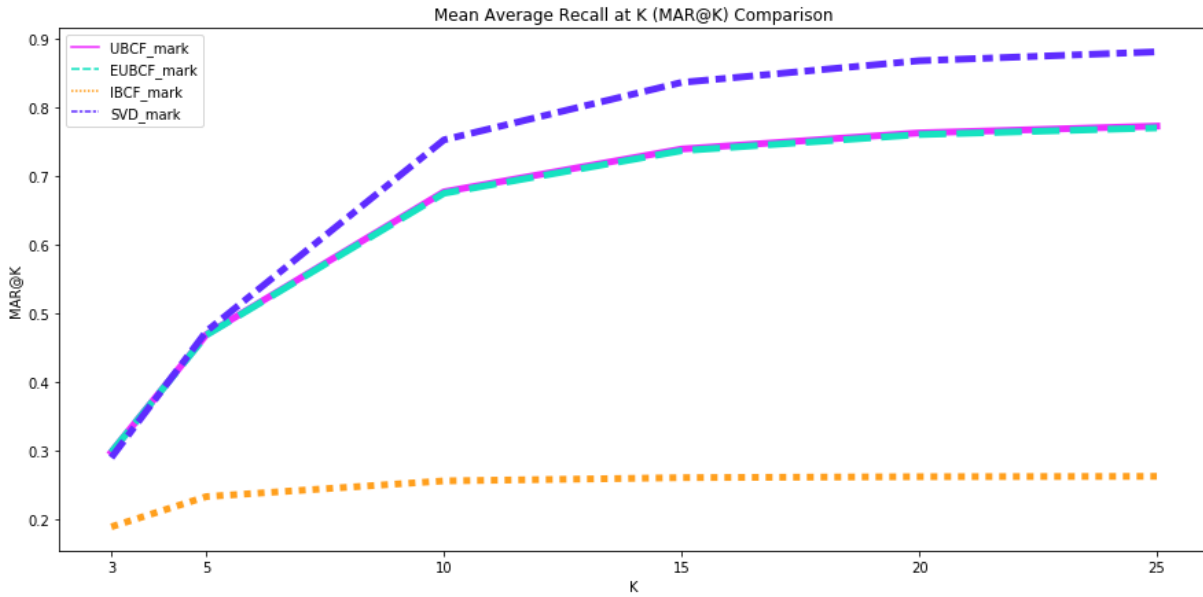


Figure 20 Mean average Recall at K comparison for Book-Crossing

Graphs (19, 20) compare the four algorithms with respect to the mean average recall (MAR@K) with different number of recommendation list (k) with values of (3, 5, 10, 15, 20 and 25).

Overall, we can notice for both dataset that the mean average recall for all techniques improved when the recommendation list movie number has increased.

For figure (19) for ML-1m dataset, UBCF and EUBCF have the highest MAR over all values of recommendation list with 0.140 and 0.481 when $k=3$ and 25 respectively. SVD algorithm has performed the lowest recall among the other algorithms.

In figure (20) for Book-Crossing dataset, SVD has MAR with around 0.289 in top-3 recommended movies and almost as much as for UBCF and EUBC algorithms but when top-n is 25, the MAR for SVD was 0.881 and for EUBCF was 0.77.

IBCF algorithm has been the lowest MAR in all top-n values and EUBCF has performed the same as UBCF.

Top-n list	NDCG-UBCF	NDCG-EUBCF	NDCG-IBCF	NDCG-SVD
3	0.833	0.834	0.735	0.781
5	0.826	0.827	0.725	0.762
10	0.815	0.816	0.712	0.746
15	0.812	0.813	0.707	0.740
20	0.811	0.812	0.703	0.738
25	0.812	0.813	0.702	0.737

Table 8 Mean Normalized Discounted Cumulative Gain (NDCG) for ML-1m

Top-n list	NDCG-UBCF	NDCG-EUBCF	NDCG-IBCF	NDCG-SVD
3	0.892	0.893	0.786	0.717
5	0.882	0.883	0.790	0.646
10	0.876	0.878	0.790	0.638
15	0.879	0.881	0.789	0.652
20	0.880	0.881	0.789	0.660
25	0.881	0.883	0.789	0.665

Table 9 Mean Normalized Discounted Cumulative Gain (NDCG) for Book-Crossing

The tables highlight information about the mean Normalized Discounted Cumulative Gain (NDCG) for all the techniques with range of recommendation lists of 3, 5, 10, 15, 20 and 25.

It can be seen that when the number of recommended movies has increased the NDCG values have gone down with all techniques except IBCF algorithm in Book-Crossing dataset, where NDCG values have improved slightly.

In table (8) for ML-1m dataset, UBCF and EUBCF have remarkably higher scores than IBCF and SVD and their quality ranking list are almost the same. However, in Book-Crossing dataset as shown table (9), the EUBCF has achieved the highest scores in all recommendation list cases and IBCF algorithm has the second-best results. SVD has the lowest values in this dataset.

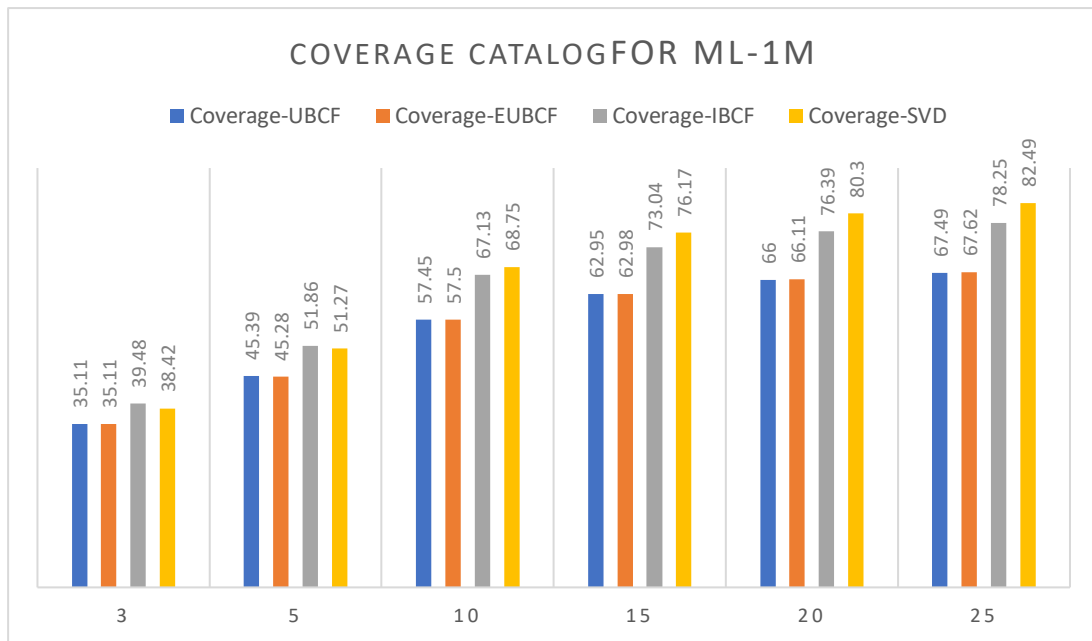


Figure 21 Catalog coverage % for ML-1m

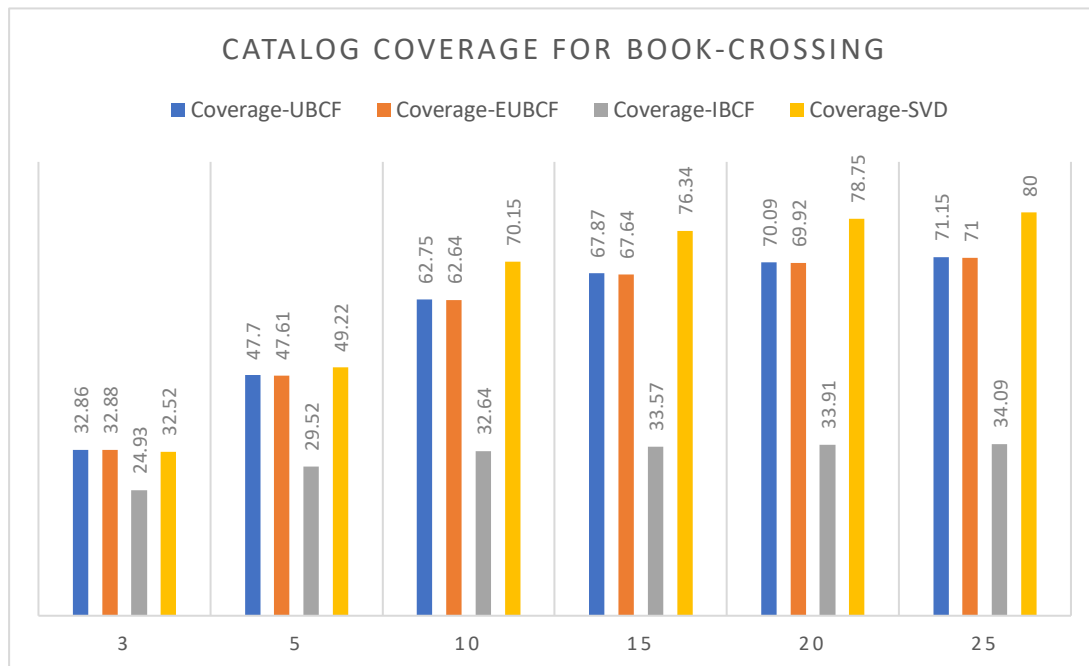


Figure 22 Catalog coverage % for Book-Crossing

The bar charts in figures (21 and 22) compare the catalog coverage values for the four algorithms and in different number of recommended movies (3, 5, 10, 15, 20, 25).

Overall, what stand out from the graph is that the coverage values have gone up when the recommendation list number of movies has increased in all techniques.

In details for ML-1m dataset, SVD technique has the highest values in all top-n cases except when sending 3 recommended movies with 38% which is slightly lower than IBCF which made coverage of 39%. And in the case of providing 25 recommended movies, SVD has almost 82% percentage coverage. While IBCF has achieved around 78%. UBCF and EUBCF algorithms have attained almost similar percentage of coverage in all different recommendation lists. Finally, it can be seen that EUBCF has made a little improvement above the UBCF.

For Book-Crossing dataset SVD has the highest percentage of coverage in all top-n movies recommended values with 32.52% and 80% when top-n recommended movies are 3 and 25 respectively.

EUBCF technique has the second highest coverage values and it has achieved similar results to UBCF algorithm. IBCF has the lowest coverage values with 24.39% and 34.09% when top-n recommended movies are 3 and 25 respectively.

5- Conclusion and future work

5-1 Conclusion

In this project we introduced comparative study between four recommendation algorithms which were implemented on two datasets (ML-1m and Book-Crossing) and their performance were compared using different evaluation measures. The results were analysed and from that we can conclude the following points:

- Enhanced user-based collaborative filtering achieves best results in terms of prediction accuracy with the lowest errors in both measures MAE and RMSE in ML-1m dataset. While, model-based singular value decomposition has the lowest MAE and RMSE in Book-Crossing dataset.
- With Regards to classification accuracy, SVD technique performed best in precision and recall with Book-Crossing dataset which has higher sparsity than MovieLens dataset and EUBCF has the highest percentage of the relevant movies in recommendation list with ML-1m dataset.
- With respect to mean average precision in ML-1m dataset, when applying the algorithms, the measure has been able to confirm that the UBCF and EUBCF have the highest ability to recommend movies were ordered in the relevant order.
Regarding the Book-Crossing dataset, SVD technique performed the best among the other algorithms based on the result of mean average precision and mean average recall.
- In both datasets, user-based and enhanced user-based collaborative filtering have the highest Normalized Discounted Cumulative Gain (NDCG). Meaning having the highest quality of the rank list.
- Singular value decomposition has achieved best result in catalog coverage among the other techniques which are implemented in both datasets to this project.
- The enhanced user-based algorithm has produced better performance in prediction accuracy of ML-1m dataset than user-based collaborative filtering but it has same performance in other evaluation measures.
- The enhanced user-based algorithm has produced better performance over user-based collaborative filtering in prediction accuracy of ML-1m dataset however almost same performance in the Book-Crossing dataset due to the limited number of demographic features.

From above we observed that the best performance is obtained by enhanced user-based collaborative filtering with MovieLen dataset but not in the Book-Crossing dataset because this dataset has more ratings than Book-Crossing dataset as collaborative filtering works better when the number of ratings is high. Singular value decomposition performed better with Book-Crossing because it works better with sparsity dataset.

We can conclude that there is no particular algorithm we can confidently consider best among other techniques. This is because the performance of the algorithm depends on the size of the dataset, the sparsity of the data and on the strength of the relationships between the features. Therefore, different techniques should be applied and evaluated and the best performing one is used in the specific domain.

5-2 Future work and improvement

This project could be improved on several aspects in the future:

- Ideally given more time, additional datasets which include more demographic features could have been used to improve the comparison accuracy and produce better results.
- More advance model-based techniques like rule-based reasoning, Naive Bayes could be implemented to find out their performance and compare the results.
- Using the movies and books content information to implement hybrid content-based with collaborative filtering which is expected to outperform the collaborative filtering which applied in the project.
- More evaluation methods could be applied to enhance the comparison study. An example is the hit rate (HR) by LOOCV method which measure the percentage of successes of the relevant items in recommended list. The higher the hit rate, the better the recommendation the system achieves.
- Record the execution time of the different techniques and use this as part of the comparison between these techniques.

6- References

- Andrews J, 2014, “Mean Average Precision isn’t so Nice”, available online at <https://juneandrews.com/2014/12/15/mean-average-precision-isnt-so-nice/>. Last accessed 20/08/2019.
- Bansari D. Patel, Palak V. Desai, Urvi N Panchal, “ Methods of recommender system: A review”, Published in International Conference on Innovations, 2017
- Becker, 2016. Matrix Factorization for Movie Recommendations in Python, available online at <https://beckernick.github.io/matrix-factorization-recommender/> [Accessed April 2019]
- Burke R, “Hybrid Recommender Systems: Survey and Experiments”, California State University, Fullerton, 2002
- Butola Manmohan singh ,“ Data Science – Hybrid Recommendation Systems “, Feb., 2016
- CS. Carleton, 2007, “General Collaborative Filtering Algorithm Ideas”, available online at http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/index.html. Last accessed 08/07/2019.
- Charu C. Aggarwal “Recommender Systems the Textbook”, Springer International Publishing Switzerland 2016.
- Cremonesi P, Turrin R, Lentini E, Matteucci M, “An evaluation Methodology for Collaborative Recommender Systems” Proc. of the 4th International Conference on Automated Solutions for Cross Media Content and Multi-channel Distribution (Axmedis), IEEE, 2008, Firenze, Italy
- Dietmar J, Markus Z, Alexander F, Gerhard F “Recommender Systems an Introduction”, Cambridge University Press (2011)
- Gorakala, Suresh Kumar “Building Recommendation” understand your data and user preferences to make intelligent, accurate, and profitable decisions”, Packt Publishing, 2016.
- Grouplens.org (2019), MovieLens 1M Dataset, 2019, available online at <https://grouplens.org/datasets/movielens/1m/> [Accessed February 2019].
- Herlocker J, konstan J, terveen L, Riedl J, “Evaluating Collaborative Filtering Recommender Systems “, ACM Transactions on Information Systems, Vol. 22, No. 1, January 2004.
- Informatik.uni-freiburg.de (2004), Book-Crossing Dataset, available online at <http://www2.informatik.uni-freiburg.de/~ciegler/BX/> [Accessed February 2019].

- James, 2018, The 4 Recommendation Engines That Can Predict Your Movie Tastes
https://medium.com/@james_aka_yale/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223 [Accessed April 2019]
- Kondurforov A, "Building a Recommendation system for e-commerce", AI Ukraine 2017.
- Leskovec J., Rajaraman A., Ullman J., "Mining of Massive Datasets", published by Cambridge University Press, 2014.
- Liu, Y., 2017. Python machine learning by example. Birmingham-Mumbai: Packt.
- Malaeb Maher, "Recall and Precision at k for Recommender systems, Detailed Explanation with examples", August 2017.
- Marginalia, 2017, "The confusion over information retrieval metrics in Recommender Systems, available online at <http://www.claudiobellei.com/2017/06/18/information-retrieval/>". Last accessed 08/08/2019.
- Mouzhi Ge, Carla Delgado-Battenfeld, Dietmar Jannach, "Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity", ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010.
- Postmus S, "Recommender system techniques applied to Netflix movie data", Research Paper Business Analytics, Vrije Universiteit Amsterdam, February 2018.
- Ricci F, Rokach L, Shapira B, Kantor P, "Recommender system handbook" Springer New York, 2011.
- Sawtelle S., 2016, "Mean Average Precision (MAP) For Recommender Systems", available online at <http://sdsawtelle.github.io/blog/output/mean-average-precision-MAP-for-recommender-systems.html>. Last accessed 15/08/2019.
- Schroder G, Thiele M, "Setting Goals and Choosing Metrics for Recommender System Evaluations", Dresden University of Technology Faculty of Computer Science, Database Technology Group, 2011.
- Scikitlearn.org (2019), Tuning the hyper-parameters of an estimator.,
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html [Accessed March 2019]
- Scikitlearn.org (2019), "precision_recall", available online at https://scikitlearn.org/stable/auto_examples/model_selection/plot_precision_recall.html. [Accessed March 2019].
- Shani G, Gunawardana A, 2011. "Evaluating Recommendation Systems". In: Ricci F, Rokach L, Shapira B, Kantor P, ed., "Recommender system handbook", Springer New York, 2011, PP.257-297.

- Stack exchange, 2018, "What does AUC stand for and what is it?", available online at <https://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it>. Last accessed 20/07/2019.
- Steinweg-Woods J., 2016, "A Gentle Introduction to Recommender Systems with Implicit Feedback", available online at . <https://jessesw.com/Rec-System/>. Last accessed 024/08/2019.
- Theobald O, "Machine learning make your own recommender system", Scatterplot press 2018.
- Vozalis M, Margaritis K, "Collaborative Filtering Enhanced by Demographic Correlation" Parallel Distributed Processing Laboratory, Department of Applied Informatics, University of Macedonia.
- Zanker M, Jannach D, "Introduction to Recommender Systems", Tutorial at ACM Symposium on Applied Computing, Sierre, Switzerland, 22 March 2010.
- Zygmunt Z, 2012, "What you wanted to know about Mean Average Precision", available online at <http://fastml.com/what-you-wanted-to-know-about-mean-average-precision>. Last accessed 08/08/2019.

Appendix A

This chart represents the types of recommender system methods (Dietmar et al., 2011)

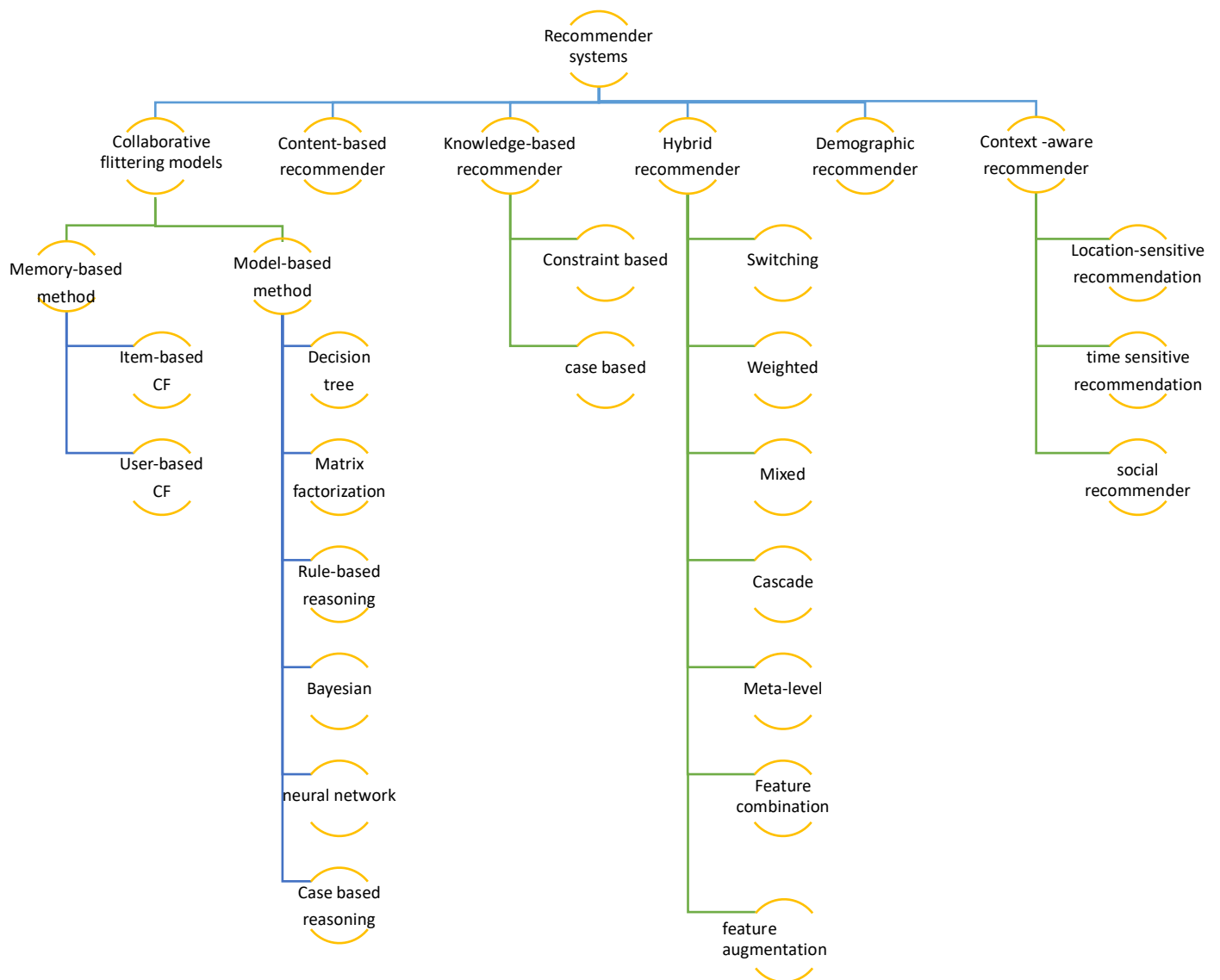


Figure 23 Types of recommender system methods

Appendix B

Table 1 compares the main advantages and disadvantages of the recommender systems methods (Burke, 2002) (Theobald, 2018) (CS. Carleton, 2007).

Methods		Advantages	Disadvantages
Collaborative filtering	Memory-based	<ol style="list-style-type: none"> 1- Easy to implement. 2- No need to the items content information for recommendation. 3- Easy to add and update the data because the prediction process using the entire database. 4- Relatively good the quality of prediction. 	<ol style="list-style-type: none"> 1- Depend on overlap in ratings between the users and has a problem when the density of ratings is low. 2- Need memory and very slow when predict the ratings as it uses all the data. 3- Limited in scalability when the dataset is large. 4- Do not generalize the data.
	Model-based	<ol style="list-style-type: none"> 1- Less effect with sparsity like singular value decomposition when the dimension is reduced. 2- Prediction speed faster than in the memory-based. 3- Easy to avoid overfitting as the model was built from real-word data. 	<ol style="list-style-type: none"> 1- Loose information for dimensionality reduction. 2- Inflexibility, it is difficult to add data in model-based system because of it does not need to query all the database.
Demographic recommender		<ol style="list-style-type: none"> 1- Does not have new user problem because it does not depend on user's ratings. 2- depend on the characteristics of the user which are more stable over time. 	<ol style="list-style-type: none"> 1- It has a problem when the is not adequate demographic information about the users. 2- Predict the user liking which is usually information that user prefers to keep it privately.
Content-based filtering		<ol style="list-style-type: none"> 1- Low profile items relevant rating can be recommended. 2- Less sensitive with users rating because the items do not change over the time and more permanent than people preferences. 3- For new or unrated items(cold-start). items can be recommended based on having similar attribute with items which the user have interacted with. 4- It is difficult to cheat the system because it is based on item to item relationship and not user potential fake reviews. 	<ol style="list-style-type: none"> 1- As this method depends on previous user usage of certain products, this method only recommends similar product and unable to provide discoverability of unfamiliar products. 2- Ineffective for new users. 3- Content based disregards the item ratings of another users. Therefore, they are limited to interpret the quality of an item.
Hybrid recommender systems		<ol style="list-style-type: none"> 1- Attempt to alleviate the weakness of different methods and leverage the strengths of these methods to generate system with best overall robustness. 2- Improve the sparsity. 	<ol style="list-style-type: none"> 1- There is increasing in complexity Implementation of the algorithms

Table 10 advantages and disadvantages of the recommender systems methods

Appendix C

MovieLens 1M Dataset

These files contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000. can be found at the following web site: <http://www.grouplens.org/>. (Grouplens.org, 2019)

RATINGS FILE DESCRIPTION

All ratings are contained in the file "ratings.dat" and are in the following format:

UserID, MovieID, Rating, Timestamp

- UserIDs range between 1 and 6040
- MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)
- Timestamp is represented in seconds.
- Each user has at least 20 ratings

USERS FILE DESCRIPTION

User information is in the file "users.dat" and is in the following format:

UserID, Gender, Age, Occupation, Zip-code

All demographic information is provided voluntarily by the users and is not checked for accuracy. Only users who have provided some demographic information is included in this data set.

- Gender is denoted by a "M" for male and "F" for female
- Age is chosen from the following ranges:

- * 1: "Under 18"
- * 18: "18-24"
- * 25: "25-34"
- * 35: "35-44"
- * 45: "45-49"
- * 50: "50-55"

* 56: "56+"

- Occupation is chosen from the following choices:

- * 0: "other" or not specified
- * 1: "academic/educator"
- * 2: "artist"
- * 3: "clerical/admin"
- * 4: "college/grad student"
- * 5: "customer service"
- * 6: "doctor/health care"
- * 7: "executive/managerial"
- * 8: "farmer"
- * 9: "homemaker"
- * 10: "K-12 student"
- * 11: "lawyer"
- * 12: "programmer"
- * 13: "retired"
- * 14: "sales/marketing"
- * 15: "scientist"
- * 16: "self-employed"
- * 17: "technician/engineer"
- * 18: "tradesman/craftsman"
- * 19: "unemployed"
- * 20: "writer"

MOVIES FILE DESCRIPTION

Movie information is in the file "movies.dat" and is in the following format:

MovieID, Title, Genres

- Titles are identical to titles provided by the IMDB (including year of release)

- Genres are pipe-separated and are selected from the following genres:

- * Action
- * Adventure

- * Animation
- * Children's
- * Comedy
- * Crime
- * Documentary
- * Drama
- * Fantasy
- * Film-Noir
- * Horror
- * Musical
- * Mystery
- * Romance
- * Sci-Fi
- * Thriller
- * War
- * Western

Appendix D

Book-Crossing Dataset

Collected by Cai-Nicolas Ziegler in a 4-week crawl (August / September 2004) from the [Book-Crossing](#) community with kind permission from Ron Hornbaker, CTO of [Humankind Systems](#). Contains 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books (Informatik.uni-freiburg.de, 2004).

The Book-Crossing dataset comprises 3 tables.

- BX-Users

Contains the users. Note that user IDs (*`User-ID`*) have been anonymized and map to integers. Demographic data is provided (*`Location`*, *`Age`*) if available. Otherwise, these fields contain *NULL*-values.

- BX-Books

Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given (*`Book-Title`*, *`Book-Author`*, *`Year-Of-Publication`*, *`Publisher`*), obtained from Amazon Web Services. Note that in case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavours (*`Image-URL-S`*, *`Image-URL-M`*, *`Image-URL-L`*), i.e., small, medium, large. These URLs point to the Amazon web site.

- BX-Book-Ratings

Contains the book rating information. Ratings (*`Book-Rating`*) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0.

Appendix E

Manual testing for the prediction ratings used in user-based collaborative filtering algorithm. the testing was performed using made up small sample of user-movie ratings. We assume a user is represented as (u) and a movie is represented as (m) then we create a ratings matrix with users as rows and movies as columns as below:

	m1	m2	m3	m4	m5	mean ratings for each user
U1	[0	0	1	0	3]	2
U2	[0	0	2	5	0]	3.5
U3	[4	0	0	0	3]	3.5
U4	[3	5	4	0	0]	4
U5	[0	0	0	2	1]	1.5

We implemented the code using (pairwise. cosine_distances) function to find the similarity between the users so that no need to test this function and we get the similarity matrix as below:

```
3 ### find the similarity between the users based on ratings activity
4
5 user_rating_sim = 1 - sklearn.metrics.pairwise.cosine_distances(rating_train_matrix)
6
7 #user_rating_sim = np.array(user_rating_sim)
8
9 print('user_rating_similarity')
10 print(user_rating_sim)
```

```
user_rating_similarity
[[1.         0.11744404 0.56920998 0.17888544 0.42426407]
 [0.11744404 1.         0.         0.21009029 0.8304548 ]
 [0.56920998 0.         1.         0.33941125 0.26832816]
 [0.17888544 0.21009029 0.33941125 1.         0.         ]
 [0.42426407 0.8304548  0.26832816 0.         1.         ]]
```

To find the ratings of the users for the movies we apply the following formula:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{i=1}^l (r_{ij} - \mu_i) \cdot \text{Cosine}(u, i)}{\sum_{i=1}^l |\text{Cosine}(u, i)|}$$

the following calculations are the predictions of the unknown ratings which represented as (0) for each user:

$$\text{Rate } u1, m1 = 2 + \frac{(4-3.5)*0.569+(3-4)*0.178}{0.569+0.178} = 2 + \frac{0.2845+(-0.178)}{0.747} = 2.142$$

$$\text{Rate } u1, m2 = 2 + \frac{(5-4)*0.1788}{0.1788} = \frac{0.1788}{0.1788} = 3$$

$$\text{Rate u1, m4} = 2 + \frac{(5-3.5)*0.117+(2-1.5)*0.424}{0.117+0.424} = 2 + \frac{0.1755+(0.212)}{0.541} = 2.716$$

$$\text{Rate u2, m1} = 3.5 + \frac{(4-3.5)*0+(3-4)*0.210}{0+0.210} = 3.5 + \frac{-1*0.210}{0.210} = 2.5$$

$$\text{Rate u2, m2} = 3.5 + \frac{(5-4)*0.210}{0.210} = 4.5$$

$$\text{Rate u2, m5} = 3.5 + \frac{(3-2)*0.117+(3-3.5)*0+(1-1.5)*0.830}{0.117+0+0.830} = 3.5 + \frac{0.117+(-0.415)}{0.947} = 3.185$$

$$\text{Rate u3, m2} = 3.5 + \frac{(5-4)*0.339}{0.339} = 4.5$$

$$\text{Rate u3, m3} = 3.5 + \frac{(1-2)*0.569+(2-3.5)*0+(4-4)*0.339}{0.569+0+0.339} = 3.5 + \frac{-0.569}{0.947} = 2.873$$

$$\text{Rate u4, m4} = 4 + \frac{(5-3.5)*0.210+(2-1.5)*0}{+0} = 4 + \frac{1.5*0.210}{0.210} = 5.5$$

$$\text{Rate u4, m5} = 4 + \frac{(3-2)*0.178+(3-3.5)*0.339+(1-1.5)*0}{0.178+0+0.339} = 4 + \frac{0.178+(-0.1695)}{0.517} = 4.0164$$

$$\text{Rate u5, m1} = 1.5 + \frac{(4-3.5)*0.268+(3-4)*0}{0.268+0} = 1.5 + \frac{0.5*0.268}{0.268} = 2$$

$$\text{Rate u5, m2} = 1.5 + \frac{(5-4)*0}{0} = \text{nan}=0$$

$$\text{Rate u3, m3} = 1.5 + \frac{(1-2)*0.424+(2-3.5)*0.830+(4-4)*0}{0.569+0.830+0} = 1.5 + \frac{-0.424-1.245}{1.399} = 0.169$$

We can see that these results are similar to the program results below:

```
user_prediction
user_prediction_shape
[[2.14131827 3.         1.09269512 2.71680318 2.25245524]
 [2.5         4.5        2.28161792 4.54631232 3.18584901]
 [3.61989502 4.5        2.87354537 4.         3.46465156]
 [3.38010498 5.         3.64432721 5.5        4.0177115 ]
 [2.         0.         0.16906738 2.45368768 1.37598903]]
```

For the complete source code of the four experiments implementation and evaluation can be found on the attached USB flash drive.

Appendix F

Manual testing for the prediction ratings used in item-based collaborative filtering algorithm. the testing was performed using the same sample of user-movie ratings above. We assume a user is represented as (u) and a movie is represented as (m) then we create a ratings matrix with users as rows and movies as columns as below:

	m1	m2	m3	m4	m5	mean ratings
U1	[0	0	1	0	3]	2
U2	[0	0	2	5	0]	3.5
U3	[4	0	0	0	3]	3.5
U4	[3	5	4	0	0]	4
U5	[0	0	0	2	1]	1.5

We implemented the code using (correlation pairwise _distances) function to find the similarity between the movies so that no need to test this function and we get the similarity matrix as below:

```
3 ### find the similarity between the movies based on ratings activity
4
5 item_correlation = 1 - pairwise_distances(rating_train_matrix.T, metric='correlation')
6 item_correlation[np.isnan(item_correlation)] = 0
7
8 print(item_correlation)
9 print(item_correlation.shape)
```



```
[[ 1.          0.45883147  0.16861332 -0.57365881  0.18604028]
 [ 0.45883147  1.          0.86859904 -0.35721725 -0.51604685]
 [ 0.16861332  0.86859904  1.          0.01363862 -0.66989385]
 [-0.57365881 -0.35721725  0.01363862  1.          -0.58688103]
 [ 0.18604028 -0.51604685 -0.66989385 -0.58688103  1.          ]]
```

To find the ratings of the users for the movies we apply the following formula:

$$\hat{r}_{ut} = \frac{\sum_{j \in Q_t(u)} \text{correlation}_{ui} \cdot r_{uj}}{\sum_{j \in Q_t(u)} \text{correlation}_{ui}}$$

$$\text{Rate } u1, m1 = \frac{(1)*0.168+(3)*0.186}{0.168+0.186} = \frac{0.726}{0.354} = 2.05$$

$$\text{Rate } u1, m2 = \frac{(1)*0.868+(3)*-0.516}{0.868+0.516} = \frac{-0.68}{1.384} = -0.491$$

$$\text{Rate } u1, m4 = \frac{(1)*0.013+(3)*-0.586}{0.013+0.586} = \frac{-1.745}{0.599} = -2.91$$

0.672

$$\text{Rate } u_2, m_1 = \frac{(2)*0.168+(5)*-0.573}{0.168+0.573} = \frac{-2.529}{0.741} = -3.41$$

$$\text{Rate } u_3, m_3 = \frac{(4)*0.168+(3)*-0.669}{0.168+0.669} = \frac{-1.335}{0.837} = -1.594$$

$$\text{Rate } u_4, m_5 = \frac{(3)*0.186+(5)*-0.516+(4)*-0.669}{0.186+0.516+0.669} = \frac{0.558-2.58-2.676}{1.371} = -3.426$$

$$\text{Rate } u_5, m_2 = \frac{(2)*-0.357+(1)*-0.516}{0.357+0.516} = \frac{-1.23}{0.873} = -1.408$$

We can see that these results are similar to the program results below:

```
user_prediction_shape
[[ 2.04913797 -0.49076916 -0.60463816 -2.90915456  1.39536184]
 [-3.40989149 -0.03988216  2.04036533  4.95963467 -3.40092162]
 [ 3.84314169  0.29458582 -1.59238742 -3.49430342  3.15685831]
 [ 3.66747349  4.23251759  4.34359978 -3.6553251  -3.42693441]
 [-1.26533961 -1.40905982 -0.94014058  0.89050089 -0.10949911]]
```

For the complete source code of the four experiments implementation and evaluation can be found on the attached USB flash drive.

Appendix G

Book-Crossing file

File name: Book-crossing recommender system

This file contains 3 files

File name	Type	Description
Book-Crossing dataset	csv	This file contains three files which are: <ul style="list-style-type: none">• BX-Book-Ratings: has ratings dataset• BX-Books: has books information dataset• BX-Users: has users information dataset
code	Python file (.py)	This file contains seven files as following: <ul style="list-style-type: none">• Enhanced user-based collaborative filtering (EUBCF): has python code of implementation and evaluation of EUBCF algorithm• Item based-collaborative filtering (IBCF): has python code of implementation and evaluation of IBCF algorithm.• User based-collaborative filtering (UBCF): has python code of implementation and evaluation of UBCF algorithm.• Singular value decomposition (SVD): has python code of implementation and evaluation of SVD algorithm.• tuning SVD in Book-Crossing: has python code of tuning the factor number for the function (svd) in SVD algorithm.• pre-processing and statistical analysis: has python code of statistical analysis for the Book-Crossing dataset.• Graphs for comparison: has python code for the MAR, MAR, precision-recall and ROC -curve graphs.
Result	Chrome HTML Document (.html)	This file contains seven files as following: <ul style="list-style-type: none">• Enhanced user-based collaborative filtering (EUBCF): has results of implementation and evaluation of EUBCF algorithm• Item based-collaborative filtering (IBCF): has results of implementation and evaluation of IBCF algorithm.• User based-collaborative filtering (UBCF): has results of implementation and evaluation of UBCF algorithm.

		<ul style="list-style-type: none"> • Singular value decomposition (SVD): has results of implementation and evaluation of SVD algorithm. • tuning SVD in Book-Crossing: has results of tuning the factor number for the function (svd) in SVD algorithm. • pre-processing and statistical analysis: has results of statistical analysis for the Book-Crossing dataset. • Graphs for comparison: has results of the MAR, MAR, precision-recall and ROC -curve graphs.
--	--	---

Appendix H

MovieLens file

File name: MovieLens recommender system

This file contains 3 files

File name	Type	Description
MovieLens-1m dataset	Text Document (.txt)	This file contains four files which are: <ul style="list-style-type: none">• Ratings: has ratings dataset• movies: has movies information dataset• Users: has users information dataset• ReadMe: has details about the dataset
code	Python file (.py)	This file contains seven files as following: <ul style="list-style-type: none">• Enhanced user-based collaborative filtering (EUBCF): has python code of implementation and evaluation of EUBCF algorithm• Item based-collaborative filtering (IBCF): has python code of implementation and evaluation of IBCF algorithm.• User based-collaborative filtering (UBCF): has python code of implementation and evaluation of UBCF algorithm.• Singular value decomposition (SVD): has python code of implementation and evaluation of SVD algorithm.• tuning SVD in ML-1m: has python code of tuning the factor number for the function (svd) in SVD algorithm.• pre-processing and statistical analysis: has python code of statistical analysis for the MovieLens dataset.• Graphs for comparison: has python code for the MAR, MAR, precision-recall and ROC -curve graphs.
Result	Chrome HTML Document (.html)	This file contains seven files as following: <ul style="list-style-type: none">• Enhanced user-based collaborative filtering (EUBCF): has results of implementation and evaluation of EUBCF algorithm• Item based-collaborative filtering (IBCF): has results of implementation and evaluation of IBCF algorithm.

		<ul style="list-style-type: none"> • User based-collaborative filtering (UBCF): has results of implementation and evaluation of UBCF algorithm. • Singular value decomposition (SVD): has results of implementation and evaluation of SVD algorithm. • tuning SVD in ML-1m: has results of tuning the factor number for the function (svd) in SVD algorithm. • pre-processing and statistical analysis: has results of statistical analysis for the MovieLens dataset. • Graphs for comparison: has results of the MAR, MAR, precision-recall and ROC -curve graphs.
--	--	---

Appendix I

Test file

File name: test by small Sample

This file contains 5 files

File name	Type	Description
made-up sample dataset for testing	Text Document (.txt)	This file contains one file: manual-test: has sample of 15 ratings of movies
Item-based prediction for sample dataset	Python file (.py)	has python code of implementation of IBCF algorithm.
Item-based prediction for sample dataset	Chrome HTML Document (.html)	has results of implementation and ratings prediction for all users of IBCF algorithm
User-based prediction for sample dataset	Python file (.py)	has python code of implementation of UBCF algorithm.
User-based prediction for sample dataset	Chrome HTML Document (.html)	has results of implementation and ratings prediction for all users of UBCF algorithm