Final Project Report: Introduction to Data Mining (CSE 5243)

Project Title: Movie Recommendation System

Collaborators: Debanjan Adak (adak.3), Ashwini Joshi (joshi.313)

Introduction:

Recommendation Systems (RS) have rigorously been used in various applications as a way to suggest items that a customer would likely be interested in by predicting customer preference. The most popular applications using recommendation systems are movies, music, news, grocery shopping. Netflix earns a revenue of around 150 million dollars every year. It is observed that users cancel their subscription if they do not find the right movie to watch. Hence, suggesting good movies depending upon user's interest plays an important role here. The report summarizes such an implementation of a Movie Recommendation System (MRS) using Collaborative Filtering approach suggesting movies to a user based on the past preferences of movies rated by all users. Collaborative Filtering (CF) can be broadly categorized into two main methods as memory-based and model-based. In memory-based method, we have built both user-based and item-based prediction models. We have also implemented Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF) with gradient descent as model-based techniques.

The rest of the report is divided as follows: We introduce the problem statement and background followed by Data Preprocessing and Exploratory Data Analysis (EDA) in the first two sections. Third section gives overall architecture of the Recommendation System followed by implementation in fourth section. Last section walks through a detailed analysis of the results obtained and comparison between different approaches taken. This is followed by the contributions of team members and a short conclusion.

Problem Statement:

Given a set of users with their previous ratings for a set of movies, the proposed system aims to recommend movies to users based on user-movie (item-based) ratings providing related content out of massive collection of both relevant and irrelevant items available in the dataset.

Background:

Recommendation systems can be broadly categorized as content-based filtering, collaborative filtering, and hybrid approach. Collaborative Filtering (CF) can be categorized into two main methods as memory-based and model-based. In memory-based approach, there are two types namely- user-based and item-based. User-based collaborative filtering approach is to predict items to the target user that are already of interest to other users who are similar to the target user. Item-based collaborative filtering approach is to predict items by inquiring into similarities between the items and other items that are already associated with the user. User profile normally contains less ratings than an Item profile. In user-based CF model, similarity between users is dynamic and hence pre-computing user neighbourhood can lead to poor performance. On the other hand, in

Item-based CF model, similarity between items is more static and hence prediction process involves only a table lookup for the similarity values & computation of the weighted sum. We have implemented both these approaches. Model-based systems are superior to memory-based because they are based on matrix factorization, thereby, deal better with sparse data, work faster and scale out well for large datasets. Hence we have also built SVD and Non-negative Matrix Factorization with gradient descent models to measure the performance.

Section 1: Data Preprocessing

This section talks about processing of the data before actually building the training model for prediction. We have used the MovieLens dataset for this project. MovieLens is a web-based movies recommendation system with 43,000 users and over 3500 movies. The dataset consists of 100,000 user rating in the range of 1 to 5 from 943 users on 1682 movies. Each user has rated at least 20 movies. Other than that, the dataset has simple demographic info for users (age, gender, occupation, zip) and Genre information of movies. Following steps have been taken for cleaning and consolidating data from the MovieLens Dataset –

1. Removed IMDB URLs and Video Release date from the item table.
2. The quote character is ignored while preparing the genre table. (for example – "Children").
3. Removed zip code from the user table as this is not a useful information for the Recommendation System.
4. Assigned column names to different data tables prepared from raw data for the ease of processing.
5. The format of the date field "Release_Date" in the item table has been fixed to "%d-%b-%Y" as per R format for the ease of data processing.
6. Some movies are having multiple ratings from the same users. These entries are de-duplicated before using it for analysis.
7. All genres with "NA" have been removed. Additionally, Genre field has been transformed into single variable for the ease of processing of the data.

Section 2: Exploratory Data Analysis

This section summarizes main characteristics for extracting patterns and trends with movie preferences in the data using visualization and other statistics in R. After performing the initial data preprocessing, relevant libraries and data has been loaded in R for the Exploratory Data Analysis of the MovieLens dataset.

1. First, we will analyze the trends of the ages of all the users in the data. From figure 2.1, we can observe that the users tend to be mostly in their late teens and mid-thirties although there seems to be another peak that occurs in the late forties.

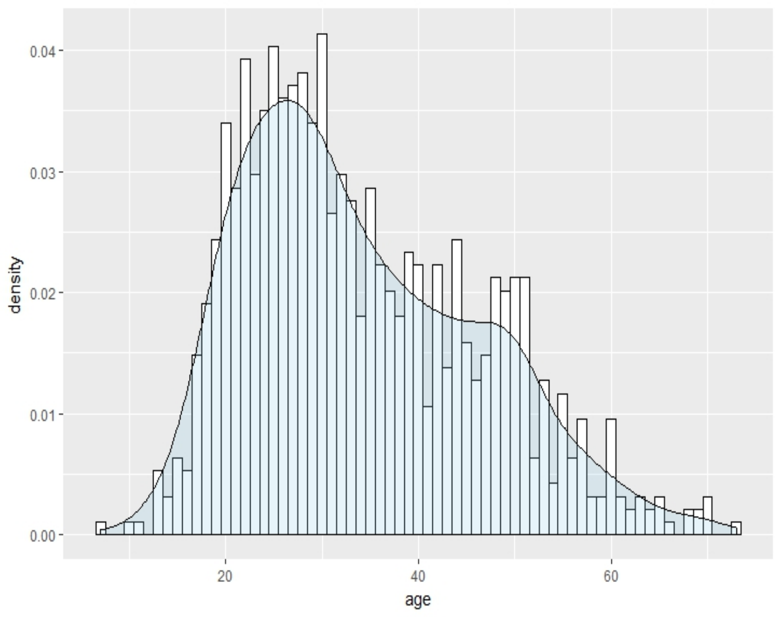


Figure 2.1: Density vs Age Histogram

2. Figure 2.2 analyses the trends of the release dates of the movies in the data. We are getting the below histogram from R. We can see that the release dates of most of the movies are after 1990. There is a significantly long tale which indicates that there must be some movies from the past starting from as early as 1922.

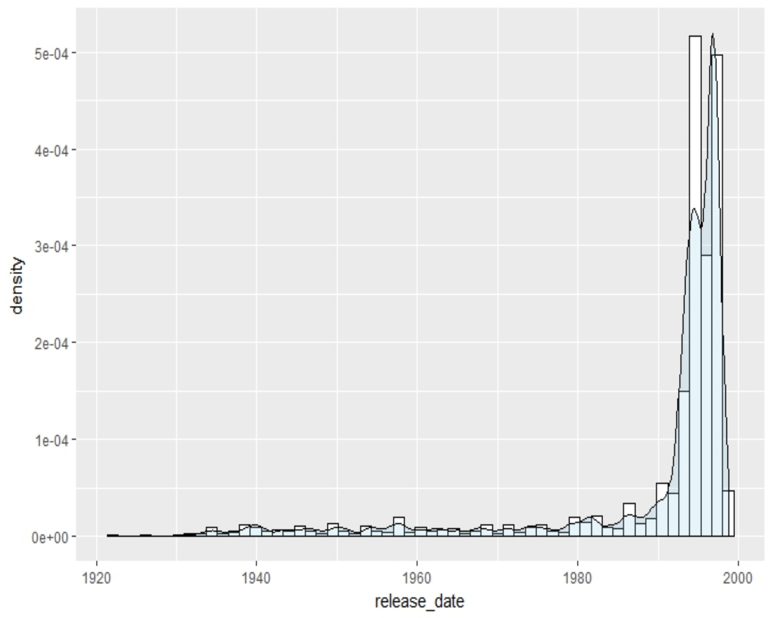


Figure 2.2: Density vs Release Date Histogram

3. Figure 2.3 to 2.6 represents the dataset with respect to the professions of the users. From each profession, we will analyze the number of users, gender bias, ages of the users from and the ratings of the users.

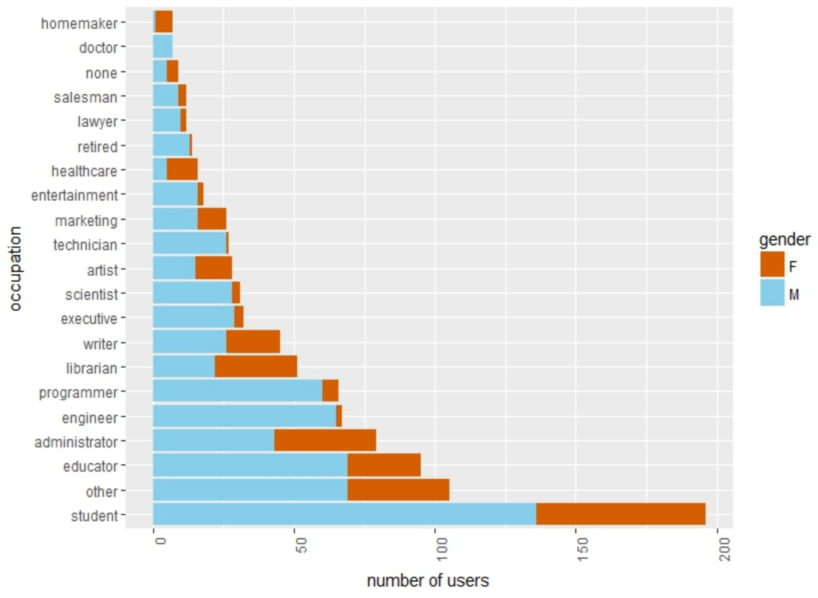


Figure 2.3: Occupation vs Number of Users and Gender Bias Graph

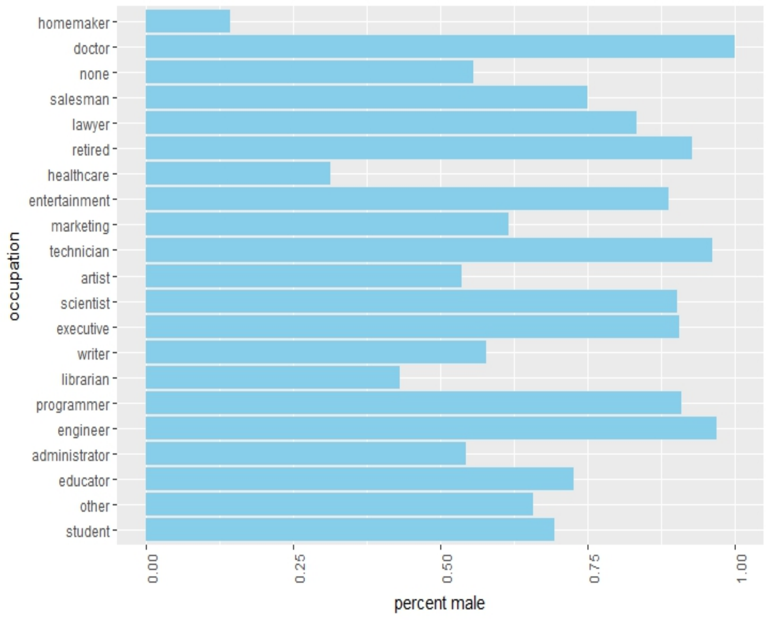


Figure 2.4: Occupation vs Percent Male Graph

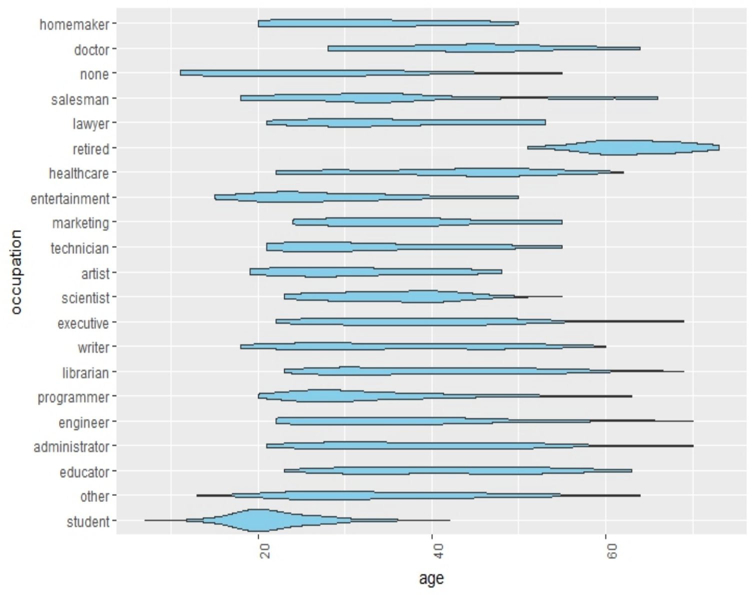


Figure 2.5: Occupation vs Age Graph

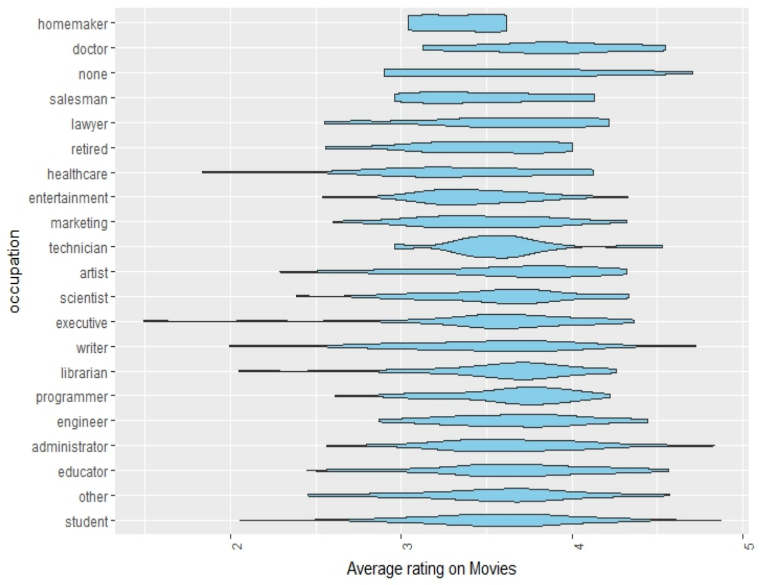


Figure 2.6: Occupation vs Average Rating on Movies Graph

From the above graphs, it is observed that there are very few doctors and homemakers among the users. We probably cannot say anything about these groups with much confidence. For most of the professions, the percentage of males are greater than the percentage of females. Some professions like doctor, technician and engineering are highly male dominated. Students have a very low average age in contrast to the retired which seems to be quite natural. There are some other interesting observations like average programmer is younger than the average health care workers. We are not getting an even distribution of the movie ratings across different professions. Some professions appear to be more picky; for example executives seem to sometime rank movies very low and healthcare workers seem to have a very low average rating.

4. Figure 2.7 shows the count of the movies from each genre.

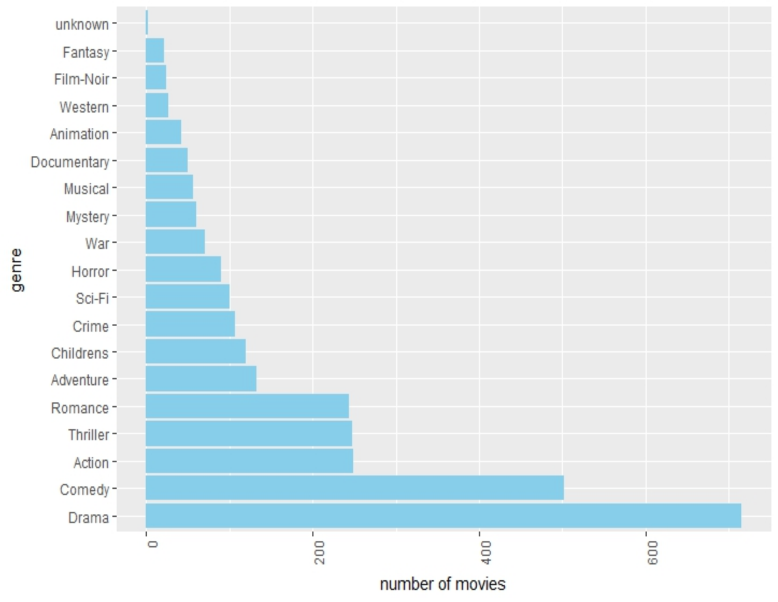


Figure 2.7: Genre vs Movie Count Graph

From the above histogram, we can see that Drama has the highest number of movies in the dataset. There are very few movies in the genre of Fantasy and Film-Noir. The distribution is highly skewed.

5. Next, we will investigate the trends in movies with respect to rating and other factors within the dataset.

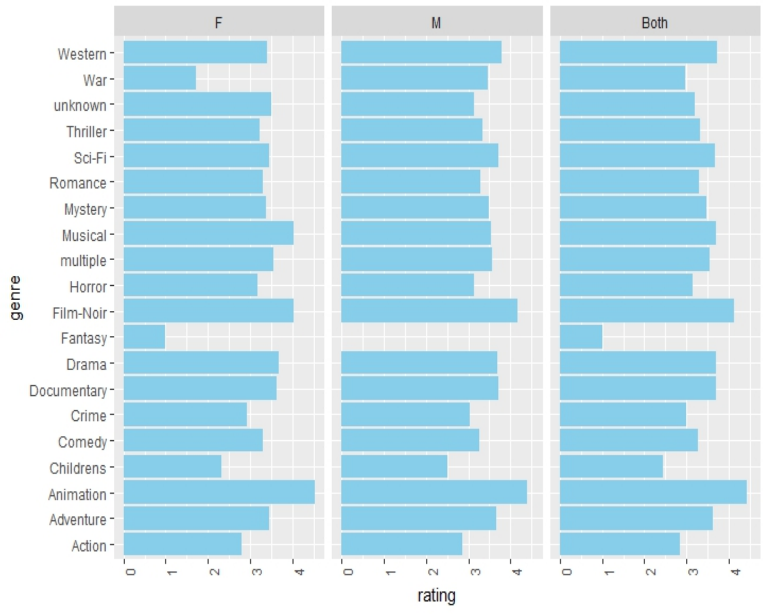


Figure 2.8: Genre vs Rating Graph

From the above histogram of Figure 2.8, we can see that Horror and Fantasy movies tend to get low ratings and they are typically not impactful. Noir and Animation seem to be the highest rated surprisingly. Low sample sizes for some of these might be a problem; the only reason fantasy and war seem like they make a difference in terms of gender is that the sample size for both is quite small. The only gender difference noticeable here is that women seem to like musicals more than men.

6. Figure 2.9 shows heatmap will demonstrate the average rating of the users from each profession across movies on different genres.

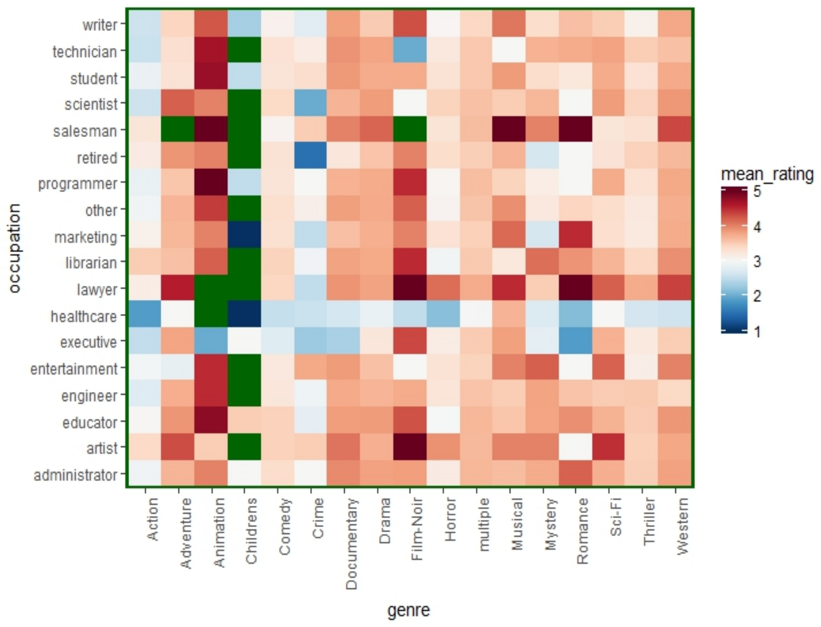


Figure 2.9: Heatmap of Average Rating of users with Occupation and Genre

From the above heatmap, it can be observed that retired and likely elderly people do not like pure crime movies. People who work in healthcare (not doctors) have extremely high standards. They mostly like the musical genres and for rest of the genres the average rating is much lower. Executives tend to dislike many movie genres but still they prefer the noir films. Lawyers tend to give high ratings to the movies they like. They prefer noir, romance and adventure films mostly. Animation movies are mostly liked by students, technician, educator and programmers.

Section 3: Flow Diagram

This section discusses overall architecture of the implemented Recommender System. Figure 3.1 shows the flow diagram of Recommender System. The dataset contains 100k records containing user ratings of different movies.

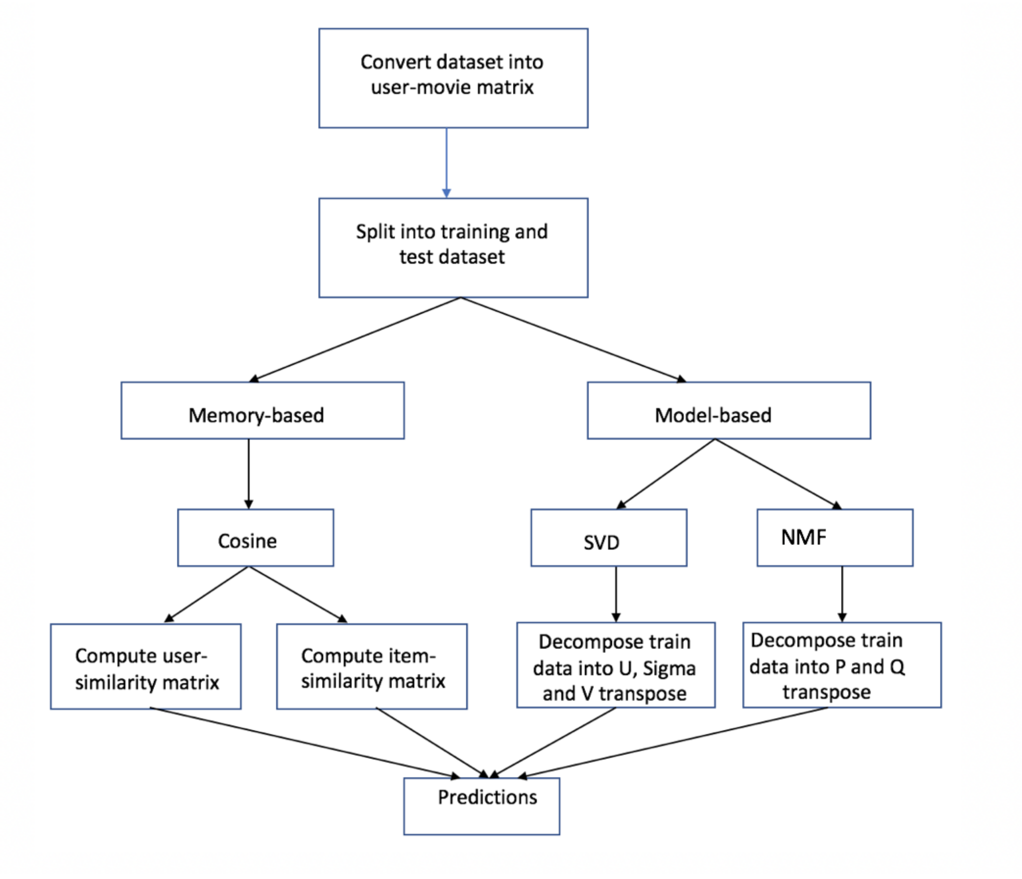

Convert dataset into user-movie matrix
Split into training and test dataset
Memory-based
Model-based
Cosine
SVD
NMF
Compute user-similarity matrix
Compute item-similarity matrix
Decompose train data into U, Sigma and V transpose
Decompose train data into P and Q transpose
Predictions

Figure 3.1: Flow Diagram for Movie Recommendation System

Steps:

1. The dataset contains these attributes - 'user_id', 'movie_id', 'Rating' and 'Timestamp'. The dataset containing users and their ratings of movies is transformed into user-movie matrix where each row is a user and each column is a movie. There are 943 distinct users and 1682 movies hence, the matrix is of size 943×1682 where entry in cell(i, j) corresponds to rating of user i for movie j. It is a sparse matrix since not all users have rated all existing movies in the dataset.
2. This data is then used for making predictions with four different approaches - user-based Cosine, item-based Cosine, Singular Value Decomposition and NMF with Gradient Descent.
3. In cosine similarity, both user-based and item-based similarity matrices are computed for making predictions.
4. In SVD, matrix is decomposed into three different matrices namely U, Sigma (Diagonal Matrix) and V.
5. In NMF, matrix is decomposed into two different matrices P and Q transpose.
6. Using above computed matrices of similarity and decomposition, movie recommendations are suggested for the users. Top 5 movies are suggested for each user where this number can easily be changed to recommend top 10 or any number of movies. Detailed explanation

and implementation of each approach is given in the next section (Section 4: Implementation).

Section 4: Implementation

As shown in figure 3.1 (flow diagram), both memory-based (Cosine with item-based and user-based) and model-based (SVD and NMF) techniques are implemented for making predictions to compare the performance.

Cosine Similarity: For building similarity matrix dataset is split and reshaped into user-movie matrix as discussed in the above section where 943 distinct users and 1682 distinct movies are present. Cosine similarity is computed from scratch in python.

For user-based model, user-similarity matrix is calculated which gives similarity between different users in terms of their ratings. Prediction is done using ratings of users who are similar to the given user using formula given below:

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$

Where r_{xi} is the predicted rating of item i by user *x*. S_{xy} is the cosine similarity between users x and y and r_{yi} is the rating by user y for item i. *N* is set of users.

In case of item, instead of calculating similarity between users, item-similarity matrix is computed which stores similarity between items which is movies in this case. This similarity between two items is measured by considering users who have rated both items by following formula:

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

Where r_{xi} is the predicted rating of user s for item i S_{ij} **is** the similarity between items i and j and r_{xj} is rating of user x for item j. *N(i;x)* is set of items.

After building similarity matrix, multiple different approaches are performed in order to find out which works best for the dataset.

1. Ratings from all users/items: In this approach, prediction of movies is made using weights of all existing users in the system and their ratings where weight is the cosine similarity between two items. In case of item-based, weights of all items are considered depending upon the value of cosine similarity between them.

2. Ratings from top-k users/items: Considering weights of all users can be expensive in terms of time taken. Also, ratings of users which have a very low similarity with the given user might be irrelevant in some cases. Hence, for each user, top-k similar users are extracted and rating calculation is done based on it. In the formulae mentioned above, instead of considering whole set of users/items, summation is taken only over top-k users/items which are most similar to the given value. Default value of k is 15 for item and 50 for user. Explanation about determining optimal value of k is discussed in the analysis part (Section 5: Evaluation).

3. No-bias ratings: In case of user-based rating, certain users may tend to always give high or low ratings to all movies. For example, one user might give 5 stars to all movies he likes and 1 to all he dislikes. Another user might give 4 and 2 for the movies he likes and dislikes. To remove this bias, mean rating is subtracted each because relative difference in the ratings that these users give is more important than the absolute rating values. In case of item-based filtering, this issue does not arise because we consider all ratings for items from a single user.

It is observed that, considering top-k users/items and removing bias by subtracting mean rating because gives the best output. This performance evaluation is discussed further in section 5. Hence, in later sections, wherever user-based or item-based CF approach using Cosine are referred, they are implemented using this approach.

Singular Value Decomposition: SVD is a matrix factorization method used in model-based Collaborative Filtering. Movie-item matrix created is factorized into three separate matrices using scikit package called as ‘svds’ which is specially used sparse matrices. It takes the original matrix which needs to be decomposed along with a parameter k which is the number of singular values you want your matrix to be decomposed in.
Using SVD, user-movie matrix is decomposed by the following formula:

$$A_{[m\times n]} = U_{[m\times r]} S_{[r\times r]} (V_{[n\times r]})^T$$

Where U is user-concepts matrix, S is a diagonal matrix containing singular values which give strength of each concept. V is movie-concept matrix. U and V are column orthogonal. In the given dataset, concept corresponds to genre of the movies. Since there are 19 genres, value of k is set to 19. Hence, matrix U is user-genre similarity matrix where each entry tells us how much a user corresponds to the given genre. S contains 19 diagonal entries which give us strength of each of the 19 genres. Reducing the value of k might give better results but as the value keeps on decreasing, the model becomes more and more prone to overfitting. Hence, k = 19 is used as the default value. Matrix V is movie-genre similarity matrix which tell us which genre each movie belongs to. By taking dot product of U and S and then multiplying it with transpose of V, ratings of users for movies are predicted.

Non-negative Matrix Factorization with Regularization: Non-negative Matrix Factorization is another method used in model-based Collaborative Filtering. Here the original user-item matrix R of dimension m x n is factorized into two separate matrices P of dimension m x r and the transpose of matrix Q of dimension n x r where m is the number of users, n is the number of movies and r is the rank of the matrix R which represents the latent features or concepts in the data. In this way, each row of P would represent the strength of the associations between a user and features. Similarly, each row of Q would represent the strength of the associations between an item and the features. To find P and Q, we first initialize the two matrices with some values, calculate how different their product is to M, and then try to minimize this difference iteratively. The difference which is usually called the error between estimated rating and real rating along with regularization term and can be calculated by the following equation for each user-item pair:

$$e_{ij}^{2} = (r_{ij} - \sum_{k=1}^{k} p_{ik}q_{kj})^2 + \frac{\beta}{2}\sum_{k=1}^{k}(\|P\| + \|Q\|)^2$$

Here r_{ij} is the actual rating of a movie n_j by user m_i. The predicted rating of the movie n_j by user m_i is given by $\sum_{k=1}^{K} p_{ik}q_{kj}$. $\|P\|$ and $\|Q\|$ represents the Frobenius norms of matrix P and Q respectively. The Frobenius norm, sometimes also called the Euclidean norm of a matrix is defined as the square root of the sum of the absolute squares of its elements. The parameter β is used to control the magnitudes of the user-feature and item-feature vectors such that P and Q would give a good approximation of R without having to contain large numbers. In practice β is set to some values in the range of 0.02. Having obtained the gradient, we can now formulate the update rules for both p_{ik} and q_{kj} :

$$p'_{ik} = p_{ik} + \alpha\frac{\delta}{\delta p_{ik}}e_{ij}^{2} = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik})$$

$$q'_{kj} = q_{kj} + \alpha\frac{\delta}{\delta q_{kj}}e_{ij}^{2} = q_{ik} + \alpha(2e_{ij}p_{ik} - \beta q_{kj})$$

Here, α is a constant whose value determines the rate of approaching the minimum. Usually, we choose a small value for α, say 0.0002. This is because if we make too large a step towards the minimum, we may run into the risk of missing the minimum and end up oscillating around the minimum. In our case, we have used value of 10 for the number of steps for the iterations. In the given dataset, concept corresponds to genre. Since there are 19 genres, value of k is set to 19.

Section 5: Evaluation

This section does thorough analysis and comparison of results obtained from different approaches taken for making predictions. Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are used as measures for evaluation.

Performance of various methods implemented for predicting movies using Cosine are tested by comparing predicted ratings and actual ratings using Root Mean Squared Error (RMSE) in Table 5.1 calculated using formula below:

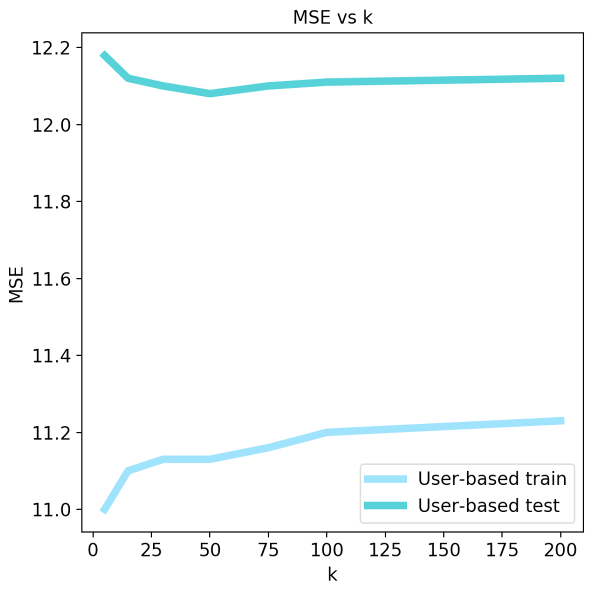
$$\sqrt{\sum_{xi}(r_{xi} - r^*_{xi})^2}$$

Where r_{xi} is the predicted rating and $r*_{xi}$ is the true rating given by the user

	Description	RMSE
Item	Ratings of all items	2.6802
	Ratings of top-k items	1.9643
User	Ratings from all users	2.8115
	Ratings from top-k users	2.6572
	Ratings after removing bias	2.3472
	Ratings from top-k after removing bias	2.1099

Table 5.1: Comparison of RMSE for all and top-k items

For implementing top-k approach, it is necessary to choose optimal value of k. A few different values of k are tested and Mean Squared Error is calculated. Figure number shows graphs of RMSE vs k for both user and item. From figure 5.1, k= 15 is chosen as optimal value for items since it is giving minimum error. For users, k = 50 gives a nice minimum error hence it is considered to be optimal.



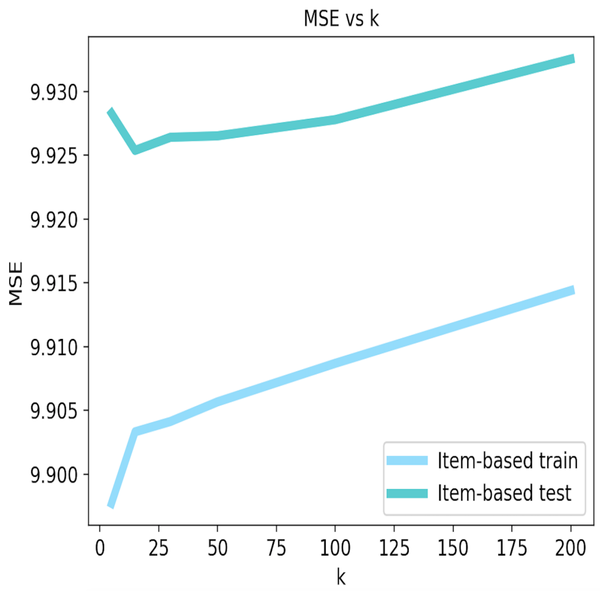


Figure 5.1: Mean Squared Error vs k for user-based and item-based predictions

Now let's compare output from model based to memory-based to find out which works the best for prediction of movies. Table 5.2 shows RMSE values for all the implemented approaches. It is seen that model-based methods work quite well as compared to memory-based. Comparing SVD with NMF tells us that NMF is better than SVD. In case of memory-based approaches, item-item similarity predicts superior results as compared to user-based Cosine similarity.

Type	Approach	RMSE
Memory-based	User-based Cosine Similarity	2.1099
	Item-based Cosine Similarity	1.9643
Model-based	Singular Value Decomposition	1.6033
	Non-negative Matrix Factorization	1.1486

Figure 5.2: RMSE values for the implemented approaches

Contributions:

- **Debanjan:**
    1. Data Preprocessing
    2. Exploratory Data Analysis of the MovieLens dataset
    3. Non-negative Matrix Factorization with regularization using gradient descent method from scratch (Model-based Collaborative Filtering)

- **Ashwini:**
    1. Cosine Similarity (Memory-based Collaborative Filtering)
        1.1. User-based from scratch
        1.2. Item-based from scratch
    2. Singular Value Decomposition (Model-based Collaborative Filtering)
    3. Evaluation of results

Conclusion:

Overall, model-based matrix factorization techniques (SVD and NMF) give finer results than memory-based approaches since they learn latent features in the data Also, in memory-based techniques, item-based works better than user-based collaborative filtering. Though implemented algorithms are giving reasonable output, they do not take into consideration the cold start problem. If a new user has joined or a new movie has released which is not rated yet by anyone, then such methods will perform poorly. Hence, implementing hybrid recommendation systems can be thought of as a future scope to solve the problem.

References

[1] Anand Rajaraman and Jeffrey David Ullman. 2011. Mining of Massive Datasets. Cambridge University Press, New York, NY, USA.

[2] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872