Homework 5.0

1 Introduction

In the age of information overload, recommender systems have become indispensable tools for filtering vast amounts of data and providing personalized suggestions to users. These systems enhance user experience on platforms ranging from e-commerce and media streaming to social networks. Various techniques exist for building recommender systems, broadly categorized into collaborative filtering, content-based filtering, and hybrid methods.

This report focuses on implementing and analyzing two fundamental approaches using the widely-used MovieLens 100k dataset:

Problem 1: Load the Movielens 100k dataset (ml-100k.zip) into Python using Pandas data frames. Convert the ratings data into a utility matrix representation and find the 10 most similar users for user 1 based on the cosine similarity of the centered user ratings data. Based on the average of the ratings for item 508 from similar users, what is the expected rating for this item for user 1?

Problem 2: Load the Movielens 100k dataset (ml-100k.zip) into Python using Pandas data frames. Build a user profile on centered data (by user rating) for both users 200 and 15, and calculate the cosine similarity and distance between the user's preferences and the item/movie 95. Which user would a recommender system suggest this movie to?

The remainder of this report is structured as follows: Section 2 describes the dataset. Section 3 details the methodology, implementation, and results for Problem 1. Section 4 covers the methodology, implementation, and results for Problem 2. Finally, Section 5 concludes the report with a summary of findings.

2 Dataset Description

The MovieLens 100k dataset is a benchmark dataset for evaluating recommender systems. It contains 100,000 ratings (1-5) from 943 users on 1682 movies. The data was collected by the GroupLens research group between September 1997 and April 1998.

The dataset is provided in several files; this analysis primarily utilizes:

- `u.data`: Contains the core rating information in the format user_id \ item_id \ rating \ timestamp.
- `u.item`: Provides item (movie) information. It includes 19 columns representing movie genres, where a value of 1 indicates the movie belongs to that genre and 0 otherwise.

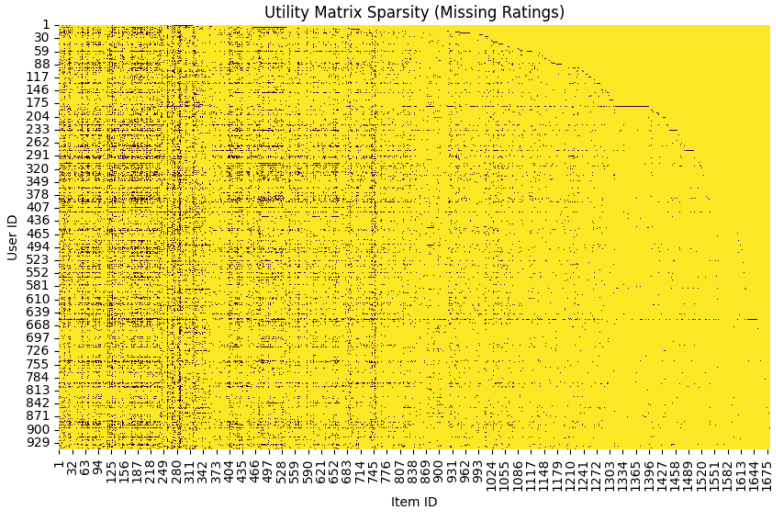

Figure 1: Sparsity of the MovieLens 100k utility matrix

Figure 2 illustrates the sparsity of the MovieLens 100k utility matrix. The blank spaces indicate missing ratings.

3 Problem 1

3.1 Objective

The goal of Problem 1 is to load the MovieLens 100k data, represent ratings in a utility matrix, find the 10 most similar users to User 1 based on the cosine similarity of centered ratings, and predict User 1's rating for Item 508 using the average rating of these similar users.

3.2 Methodology

Data Loading: Load the u.data file into a Pandas DataFrame.

Utility Matrix Creation: Convert the DataFrame into a user-item utility matrix using pivot_table. Missing values (items not rated by a user) are represented by NaN.

Rating Centering: To account for users' individual rating scales, ratings are centered by subtracting each user's average rating from their individual ratings.

User Similarity Calculation: Cosine similarity is computed between the centered rating vectors of all pairs of users. Cosine similarity measures the angle between two vectors; a value closer to 1 indicates high similarity.

Finding Nearest Neighbors: For a target user (User 1), the 10 users with the highest cosine similarity scores (excluding the user itself) are identified as the "nearest neighbors."

Rating Prediction: The predicted rating for the target user on a specific item (Item 508) is calculated as the average of the actual ratings given to that item by the identified nearest neighbors.

3.3 Implementation

The implementation utilized Python with the Pandas library for data manipulation and Scikit-learn's cosine_similarity function. The raw u.data was loaded into a DataFrame. A utility matrix was created using pivot_table. User-specific average ratings were calculated and used to subtract from the original ratings, creating the centered ratings. Missing values were filled with 0 before computing the cosine similarity matrix between all users.

For User 1, the row corresponding to User 1's similarities was selected, and the top 10 similar users were identified.

3.4 Results and Analysis

The top 10 users most similar to User 1, based on the cosine similarity of their centered ratings, were found to be in Figure 2.

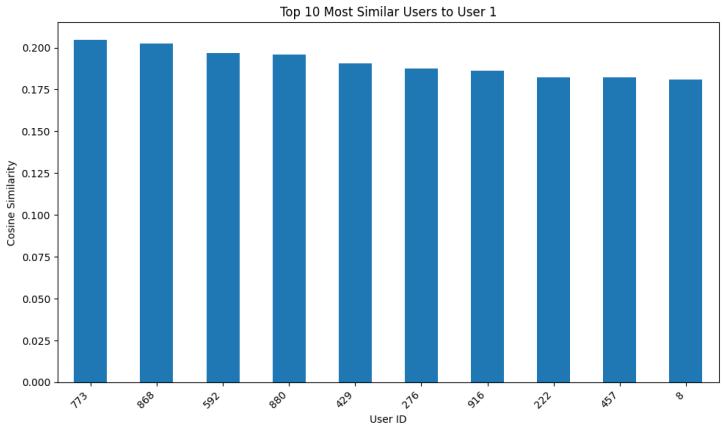

Figure 2: Top 10 Most Similar Users to User 1

This list and their scores demonstrate that even the most similar users may have moderate cosine similarity scores, reflecting the diversity in user rating patterns.

The distribution of original ratings and centered ratings are shown in Figure 3 and Figure 4, respectively.

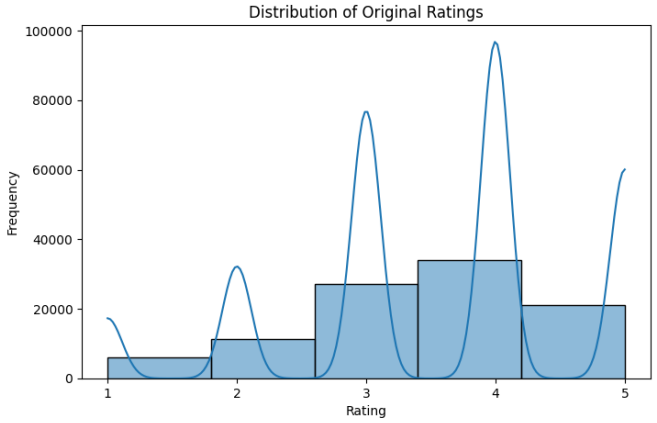

Figure 3: Distribution of Original Ratings

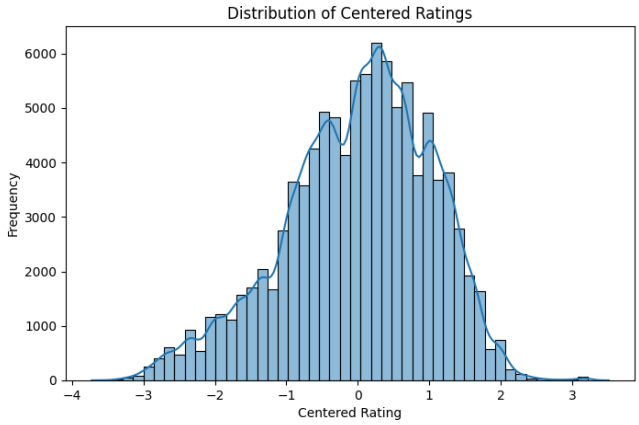

Figure 4: Distribution of Centered Ratings

Figure 3 shows that centering shifts the distribution to be centered around 0, highlighting deviations from a user's personal average.

We can also display the similarity matrix of the top 20 users in Figure 5.

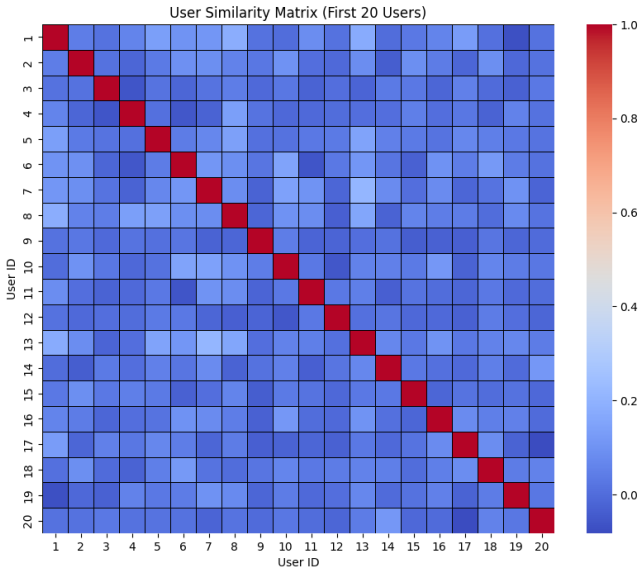

Figure 5: User Similarity Matrix.png

The predicted rating for User 1 on Item 508 is 4.200, based on the average of the ratings given by the 10 most similar users.

4 Problem 2

4.1 Objective

The goal of Problem 2 is to load the MovieLens 100k data, build user profiles based on centered ratings and item genre information for users 200 and 15, calculate the cosine similarity and distance between each user's profile and Item 95's genre features, and determine which user the system would suggest this movie to.

4.2 Methodology

Data Loading: Load u.data and u.item using Pandas.

Rating Centering: As in Problem 1, user ratings are centered to normalize preferences.

Item Feature Representation: Items (movies) are represented by a feature vector. In this case, Item 95 is represented by its 19-dimensional binary genre vector.

User Profile Construction: A user profile is created by taking the genre vectors of all movies rated by the user and performing a weighted sum.

Similarity Calculation: Cosine similarity is computed between the user's genre preference profile vector and the target item's genre feature vector.

4.3 Implementation

The implementation began by loading `u.data` and `u.item` into Pandas DataFrames. User ratings were then **centered** by subtracting each user's average rating. **Movie genre information** was extracted from `u.item` to form a `movie_genres` DataFrame, indexed by `movie_id`.

A custom `build_user_profile` function was developed to construct a user's genre preference vector. This function takes a `user_id`, filters their centered ratings, merges them with the `movie_genres` DataFrame, and calculates a **weighted sum** of the genre features, where weights are derived from the user's centered ratings for each movie. This function was applied to **Users 200 and 15** to generate their respective genre profiles.

Subsequently, **Item 95's genre vector was extracted**. For similarity calculation, both the generated user profile Series and Item 95's genre Series were converted to (1, -1) **NumPy arrays**. Finally, their **cosine similarity was computed using** `sklearn.metrics.pairwise.cosine_similarity`.

4.4 Results and Analysis

First, we look at the distribution of genres across the dataset, which forms the basis of item content in Figure 6.

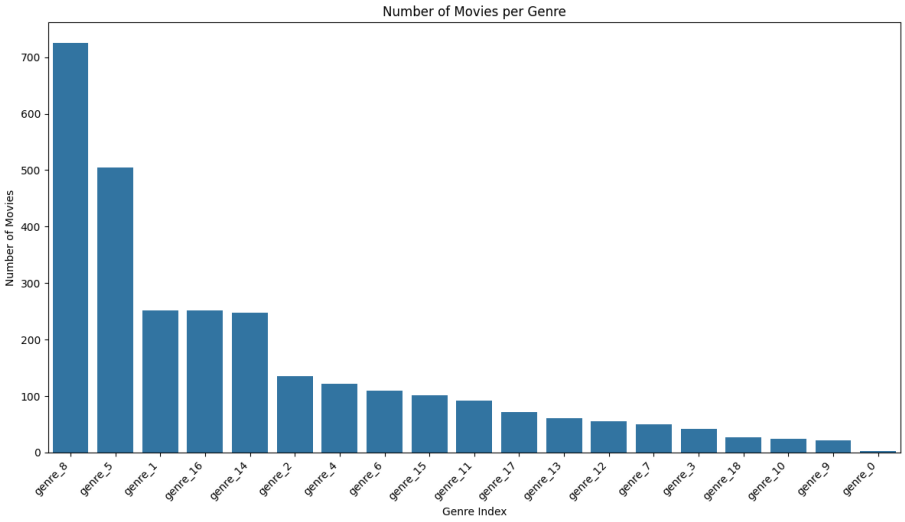

Figure 6: Number of Movies per Genre.png

Next, we visualize the genre preference profiles built for User 200 and User 15 in Figure 7.

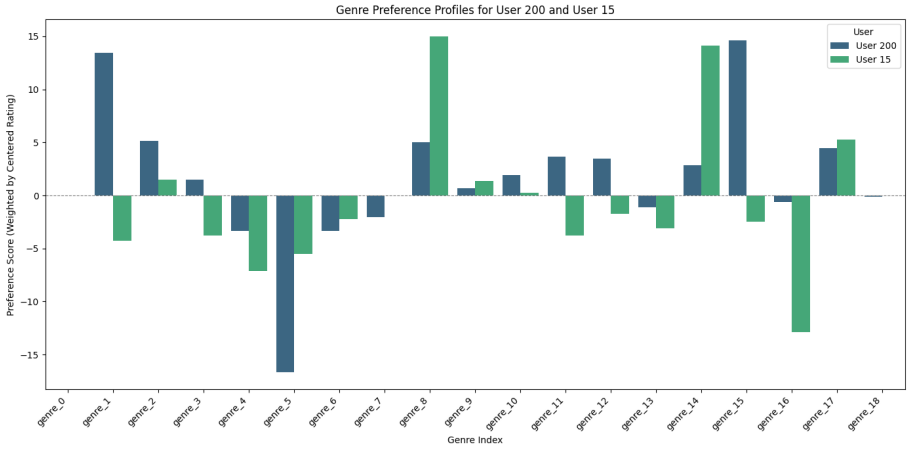

Figure 7: Genre Preference Profiles for User 200 and User 15

Positive scores indicate a preference above the user's average rating, while negative scores indicate a preference below it. Figure 7 clearly illustrates the distinct genre preferences of the two users based on their centered ratings. For instance, **User 200** shows strong positive preferences for **genre_1** and **genre_15**, alongside a negative preference for **genre_5** and **genre_4**. In contrast, **User 15** demonstrates preferences for **genre_8** and **genre_14**, while disliking **genre_16** and **genre_4**.

Item 95's content features (genres) are illustrated in Figure 8. As shown in Figure 8, Movie 95 belongs to the genres **genre_3**, **genre_4**, **genre_5**, and **genre_12**.

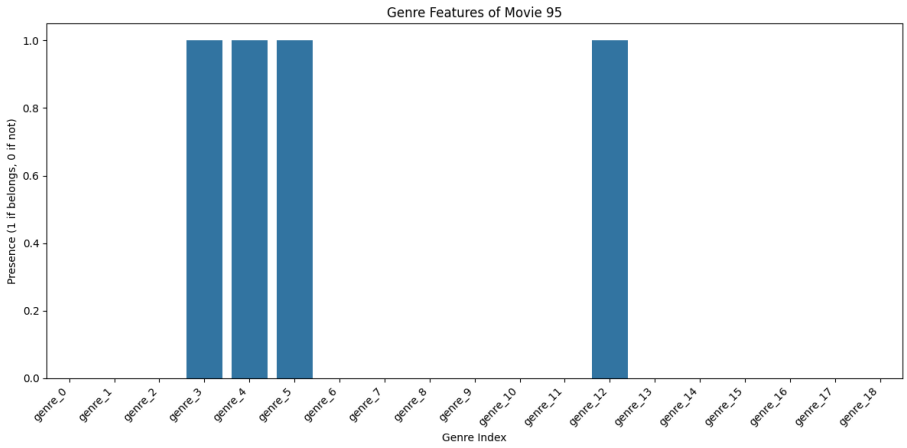

Figure 8: Genre Features of Movie 95

Finally, the **cosine similarity** and **distance** between each user profile and Item 95's genre features were calculated:

- **User 200 vs. Item 95:** Cosine Similarity = **-0.2652**, Distance = **1.2652**.
- **User 15 vs. Item 95:** Cosine Similarity = **-0.3259**, Distance = **1.3259**.

Analyzing the results, **User 200 has a higher cosine similarity (-0.2652) with Item 95 than User 15 (-0.3259)**. This **indicates a stronger match** between the genre preferences of User 200 and the genres of Movie 95.

Therefore, based on this content-based matching approach, a recommender system would be more inclined to **suggest Movie 95 to User 200** based on our result.

5 Conclusion

For **Problem 1**. By identifying the **10 most similar users** to User 1 based on centered rating patterns, the predicted rating was calculated as the average of their actual ratings for Item 508, resulting in a value of **4.200**.

For **Problem 2**. User profiles reflecting genre preferences were built for **User 200 and User 15** using centered ratings and movie genre data. By calculating the **cosine similarity** between these user profiles and Item 95's genre features, it was determined that **User 200 had a higher similarity score (-0.2652) with Item 95**. Consequently, a recommender system utilizing this content-based approach would **recommend Movie 95 to User 200**.