Please complete the assigned problems to the best of your abilities. Ensure that the work you do is entirely your own, external resources are only used as permitted by the instructor, and all allowed sources are given proper credit for non-original content.

1. Practicum Problems

These problems will primarily reference the lecture materials and the examples given in class using Python. It is suggested that a Jupyter/IPython notebook be used for the programmatic components.

1.1 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?

In this task, the goal is to predict user 1's potential rating for movie 508 using a user-based collaborative filtering approach. First, I loaded the original rating data from the MovieLens 100k dataset and constructed a user-item utility matrix, where each cell represents a user's rating for a particular movie. Since different users have different rating standards, I followed the task requirement to center each user's ratings by subtracting their individual average rating, thereby standardizing the evaluation scale. Afterward, missing values in the matrix were filled with zeros to enable cosine similarity computation. In the similarity calculation phase, I selected user 1 as the target and computed the cosine similarity between user 1 and all other users based on the centered ratings, then identified the top 10 most similar users as neighbors. I then extracted the ratings these similar users gave to movie 508 and applied two different methods to predict user 1's rating: a simple average and a similarity-weighted average. The simple average takes the mean of the neighbors' ratings, while the weighted average multiplies each rating by its corresponding similarity score and normalizes the result, yielding a more fine-grained prediction. Finally, I compared the outcomes of the two methods and checked whether user 1 had actually rated movie 508 to assess the practical significance of the prediction.

The following library functions are used:
pandas: Used to read the rating data and construct the user-item rating matrix using DataFrame operations.
numpy: Used for performing numerical operations, such as calculating averages or handling missing data.
scipy.spatial.distance.cosine: Used to compute the cosine similarity between user rating vectors.

The following is the program's output:

```
[943 rows x 1682 columns]
The 10 users most similar to User 1:
User ID: 773, Similarity: 0.2048
User ID: 868, Similarity: 0.2023
User ID: 592, Similarity: 0.1966
User ID: 880, Similarity: 0.1958
User ID: 429, Similarity: 0.1907
User ID: 276, Similarity: 0.1875
User ID: 916, Similarity: 0.1864
User ID: 222, Similarity: 0.1824
User ID: 457, Similarity: 0.1823
User ID: 8, Similarity: 0.1809
```

Figure 1

The 10 most similar users and their similarity scores

```
Rating details of similar users for movie 508:
User ID: 592, Rating for Movie 508: 5.0, Average User Rating: 3.81
User ID: 880, Rating for Movie 508: 4.0, Average User Rating: 3.43
User ID: 429, Rating for Movie 508: 4.0, Average User Rating: 3.39
User ID: 276, Rating for Movie 508: 5.0, Average User Rating: 3.47
User ID: 222, Rating for Movie 508: 3.0, Average User Rating: 3.05
```

Figure 2

Ratings from the top ten similar users who rated the movie 508

```
Weighted details of similar user ratings:
User ID: 592, Rating: 5.0, Similarity: 0.1966, Weighted Scoring: 0.9830
User ID: 880, Rating: 4.0, Similarity: 0.1958, Weighted Scoring: 0.7832
User ID: 429, Rating: 4.0, Similarity: 0.1907, Weighted Scoring: 0.7626
User ID: 276, Rating: 5.0, Similarity: 0.1875, Weighted Scoring: 0.9374
User ID: 222, Rating: 3.0, Similarity: 0.1824, Weighted Scoring: 0.5472
```

Figure 3

Similarity-weighted scores from the top ten users who rated movie 508

For simple prediction scoring:

$$\text{Predicted Rating}_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} r_i$$

Where:

r_i :Rating of movie 508 by the i-th similar user.

N :Number of similar users with ratings.

So the simple average score is:

$$\frac{5.0+4.0+4.0+5.0+3.0}{5}=\frac{21.0}{5}=4.2000$$

For similarity-weighted scoring:

$$\text{Predicted Rating}_{\text{weighted}}=\frac{\sum_{i=1}^{N}\text{sim}_i\cdot r_i}{\sum_{i=1}^{N}\text{sim}_i}$$

Where:

sim_i:The similarity between user i and user 1

r_i:The similarity between user i and user 1

Numerator (weighted score sum):
0.9830+0.7832+0.7626+0.9374+0.5472=4.0134
Denominator (sum of similarities):
0.1966+0.1958+0.1907+0.1875+0.1824=0.9530

Final weighted score:

$$\frac{4.0134}{0.9530}\approx 4.2116$$

The difference is not significant, so User A predicts a rating of about 4.2 for Movie 508.

1.2 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?

For Question 2, I adopted a recommendation system method based on user-centered ratings and cosine similarity. First, for each user, I subtracted their average rating from their original ratings, converting them into deviation values relative to the user's own rating center. This eliminates the impact of individual rating habits and makes subsequent similarity calculations more accurate and fair. When constructing user profiles, I treated each user as a multidimensional vector, with each dimension representing the user's centered rating for a different movie. As required by the problem, I used cosine similarity to compare users with a specific movie (movie 95). In the actual calculation, I first extracted the centered rating vectors for user 200 and user 15, and identified the movies they both rated. Using these commonly rated movies as the basis for calculation, I then computed the cosine similarity between each user and movie 95. The closer the cosine similarity is to 1, the more similar the user's rating pattern is to the target movie; the closer it is to 0, the greater the difference in rating patterns. Specifically, I measured similarity by calculating the cosine value of the angle between a user's centered rating vector and the centered rating vector of movie 95. Finally, I

compared the cosine similarities of user 200 and user 15 to movie 95 and selected the user with the higher similarity as the recommendation target.

The following library functions are used:
pandas: Used for reading data files, handling tabular data, and constructing the user-item rating matrix.
numpy: Used for performing numerical computations, such as calculating means and handling vectors and arrays.
cosine_similarity: Provide a commonly used cosine similarity calculation function in machine learning, used to measure the degree of similarity between vectors.

Based on the final program output, we can see that the cosine similarity between user 200 and movie 95 is 0.0890, while the cosine similarity between user 15 and movie 95 is 0.8695. Therefore, the recommendation system should suggest movie 95 to user 15, as the cosine similarity is higher.

-

END