Machine Learning Assignment (8)

RECOMMENDER SYSTEM.

Name: Mohamed El-Sayed Eid

Sec: 2, **BN:** 11

Submitted to: Eng. Mirna Atef

1. Generation of Utility Matrix

Function Description: create_utility_matrix

- **Purpose:**
  - The create_utility_matrix function creates a sparse matrix representing user-item ratings from a given pandas DataFrame. This is useful for handling large datasets with many zero entries (i.e., when most users have not rated most items). Additionally, the function generates mappings between user and movie IDs and their respective indices, which are essential for efficient matrix operations and lookups. Here's how it works:

1. **Input Data:** The function takes a DataFrame containing user IDs, movie IDs, and ratings. Each row represents a rating given by a user to a movie.

2. **Identify Unique Users and Movies:** It identifies all unique users and movies from the dataset to determine the dimensions of the resulting matrix.

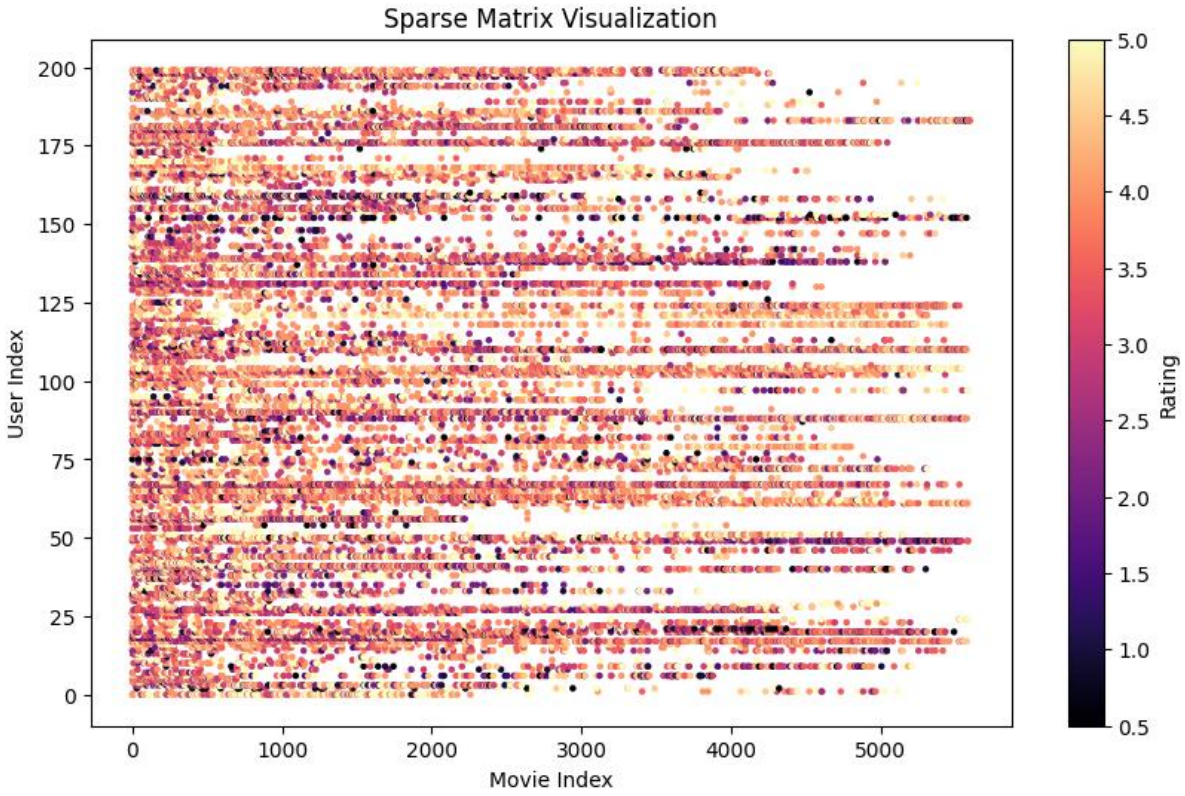
3. **Mapping Creation:**

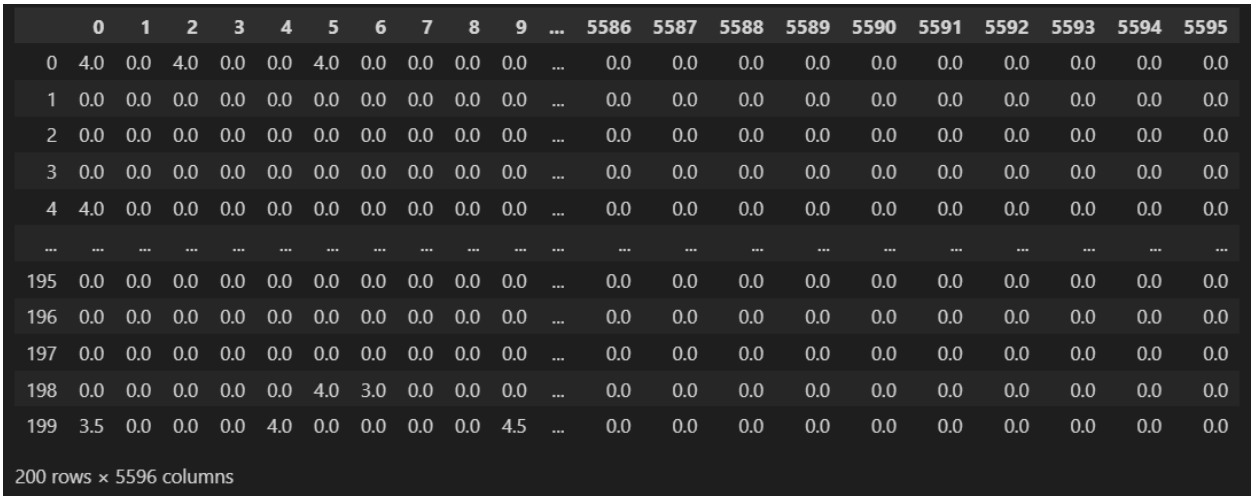
- User Mapping: It creates a dictionary that maps each unique user ID to a unique integer index. Another dictionary is created to reverse this mapping, converting indices back to user IDs.
- Movie Mapping: Similarly, it creates a dictionary that maps each unique movie ID to a unique integer index, along with a reverse mapping dictionary.

4. **Index Conversion:** Using the created mappings, the function converts the user IDs and movie IDs in the dataset to their respective integer indices.

5. **Sparse Matrix Construction:** It constructs a sparse matrix using `csr_matrix` form SciPy module where rows correspond to users, columns correspond to movies, and the values are the ratings. A sparse matrix is used because it efficiently stores and processes data with many missing values (most users haven't rated most movies).

6. **Return Values:** The function returns the sparse matrix along with the four dictionaries that provide the mappings between user/movie IDs and their integer indices.





	0	1	2	3	4	5	6	7	8	9	...	5586	5587	5588	5589	5590	5591	5592	5593	5594	5595
0	4.0	0.0	4.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
195	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
196	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
197	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
198	0.0	0.0	0.0	0.0	0.0	4.0	3.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
199	3.5	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	4.5	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

200 rows × 5596 columns

Utility Matrix: Each row is UserId and each column is MovieId

2. Recommending Similar Movies

Function Description: find_similar_movies

- **Purpose:**
  - This function finds and returns a list of movies that are similar to a given movie based on user ratings. It leverages the k-nearest neighbors (kNN) algorithm to identify similar movies. Here's how it works:

1. **Input Data:** The function takes a specific movie ID, a user-item matrix (where rows represent users and columns represent movies with their ratings), dictionaries mapping movie names to IDs and IDs to names, the number of similar movies to find, and a distance metric for similarity calculations (defaulting to cosine similarity).

2. **Transpose the Matrix:** The user-item matrix is transposed so that rows represent movies and columns represent users. This way, the similarity between movies is calculated based on user ratings.

3. **Retrieve the Movie Vector:** The function retrieves the vector of ratings for the specified movie from the transposed matrix. This vector represents how different users have rated the movie.

4. **Initialize k-Nearest Neighbors (kNN):** A kNN model is created using the specified distance metric (e.g., cosine similarity). This model is used to find the nearest neighbors of the specified movie in terms of user ratings.

5. **Find Nearest Neighbors:** The kNN model is applied to the movie vector to find the nearest neighbors. These neighbors are the movies that have the most similar rating patterns to the specified movie.

6. **Sort Neighbors:** The neighbors are sorted by their similarity to the specified movie. The closest neighbor is the movie itself, which is excluded from the final list of similar movies.

7. **Map Indices to Movie Titles:** The indices of the nearest neighbors are mapped back to their corresponding movie titles using the provided dictionary mappings.

8. **Return Results:** The function returns a list of similar movie titles, ordered from most similar to least similar. Additionally, it returns a similarity matrix that shows the similarity relationships between the movies.

```
Because you watched Toy Story (1995):

['Toy Story 2 (1999)',
 'Jurassic Park (1993)',
 'Independence Day (a.k.a. ID4) (1996)',
 'Star Wars: Episode IV - A New Hope (1977)',
 'Forrest Gump (1994)',
 'Lion King, The (1994)',
 'Star Wars: Episode VI - Return of the Jedi (1983)',
 'Mission: Impossible (1996)',
 'Groundhog Day (1993)',
 'Back to the Future (1985)']
```

Recommended 10 Movies similar to ‘Toy Story (1995)’

```
Because you watched Waiting to Exhale (1995):

['Before and After (1996)',
 'Walking Dead, The (1995)',
 'Nothing Personal (1995)',
 'Bliss (1997)',
 'Gordy (1995)',
 'Spirits of the Dead (1968)',
 'Sum of Us, The (1994)',
 'I Like It Like That (1994)',
 'Poetic Justice (1993)',
 'Georgia (1995)']
```

Recommended 10 Movies similar to ‘Waiting to Exhale (1995)’