LAB # 8

COLLABORATIVE FILTERING

OBJECTIVE

Implementing collaborative filtering for a recommender system

Lab Tasks:

For the given dataset, build a recommender system using item based collaborative filtering, which recommends movies for a selected user. If we enter a user name into the recommender, the recommender is supposed to return the list of recommended movies which have the highest predicted ratings. Use Nearest Neighbors to calculate the distance between movies by using the cosine similarity.

```
import pandas as pd

# Create the ratings DataFrame
ratings_data = {
    'UserID': [1, 1, 2, 2, 3, 3, 3, 4],
    'MovieID': [1, 2, 1, 3, 2, 3, 4, 4],
    'Rating': [5, 3, 4, 2, 4, 5, 3, 5]
}
ratings_df = pd.DataFrame(ratings_data)

# Save to ratings.csv
ratings_df.to_csv('ratings.csv', index=False)

# Create the movies DataFrame
movies_data = {
    'MovieID': [1, 2, 3, 4],
    'Title': ['The Dark Knight', 'Inception', 'Interstellar', 'The Matrix']
}
movies_df = pd.DataFrame(movies_data)

# Save to movies.csv
movies_df.to_csv('movies.csv', index=False)

print("CSV files 'ratings.csv' and 'movies.csv' have been created.")
```

```
CSV files 'ratings.csv' and 'movies.csv' have been created.
```

```python
import pandas as pd
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics.pairwise import cosine_similarity

# Step 1: Load the dataset (ratings and movie data)
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')

# Step 2: Create a User-Item matrix (pivot table)
user_item_matrix = ratings.pivot_table(index='UserID', columns='MovieID', values='Rating')

# Step 3: Calculate the cosine similarity between movies (Item-based collaborative filtering)
movie_similarity = cosine_similarity(user_item_matrix.T.fillna(0))  # Transpose for item-based filtering

# Create a DataFrame for movie similarities
movie_similarity_df = pd.DataFrame(movie_similarity, index=user_item_matrix.columns, columns=user_item_matrix.columns)

# Step 4: Function to recommend movies for a specific user
def recommend_movies(user_id, top_n=3):
    # Get the user's ratings
    user_ratings = user_item_matrix.loc[user_id]

    # Find movies rated by the user
    rated_movies = user_ratings[user_ratings > 0].index.tolist()

    # Predict ratings for all movies not rated by the user
    predicted_ratings = {}
    for movie_id in user_item_matrix.columns:
        if movie_id not in rated_movies:
            # Get similar movies to the ones the user rated
            similar_movies = movie_similarity_df[movie_id]
            predicted_rating = sum(user_ratings[rated_movie] * similar_movies[rated_movie] for rated_movie in rated_movies) / sum
            predicted_ratings[movie_id] = predicted_rating
```

```python
    # Sort the predicted ratings for all unrated movies
    recommended_movie_ids = sorted(predicted_ratings, key=predicted_ratings.get, reverse=True)[:top_n]

    # Get movie titles for the recommended movie IDs
    recommended_movies = movies[movies['MovieID'].isin(recommended_movie_ids)]

    return recommended_movies

# Step 5: Test the recommender system with a user (e.g., UserID = 3)
user_id = 3
recommended_movies = recommend_movies(user_id)

# Print the recommended movies
print(f"Recommended movies for User {user_id}:")
print(recommended_movies[['MovieID', 'Title']])
```

```
Recommended movies for User 3:
   MovieID            Title
0        1  The Dark Knight
```

Home Tasks:

A music streaming platform wants to recommend songs to users based on their listening history. The goal is to use item-based collaborative filtering to recommend songs that are similar to the ones a user has already listened to.

```
import pandas as pd

# Create the music ratings DataFrame
ratings_data = {
    'UserID': [1, 1, 2, 2, 3, 3, 3, 4],
    'SongID': [1, 2, 1, 3, 2, 3, 4, 4],
    'Rating': [5, 3, 4, 2, 4, 5, 3, 5]
}
ratings_df = pd.DataFrame(ratings_data)

# Save to 'music_ratings.csv'
ratings_df.to_csv('music_ratings.csv', index=False)

# Create the songs DataFrame
songs_data = {
    'SongID': [1, 2, 3, 4],
    'Title': ['Song A', 'Song B', 'Song C', 'Song D']
}
songs_df = pd.DataFrame(songs_data)

# Save to 'songs.csv'
songs_df.to_csv('songs.csv', index=False)

print("CSV files 'music_ratings.csv' and 'songs.csv' have been created.")
```

```
CSV files 'music_ratings.csv' and 'songs.csv' have been created.
```

```python
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

# Step 1: Load the dataset (ratings and song data)
ratings = pd.read_csv('music_ratings.csv')
songs = pd.read_csv('songs.csv')

# Step 2: Create a User-Item matrix (pivot table)
user_item_matrix = ratings.pivot_table(index='UserID', columns='SongID', values='Rating')

# Step 3: Calculate the cosine similarity between songs (Item-based collaborative filtering)
song_similarity = cosine_similarity(user_item_matrix.T.fillna(0))  # Transpose for item-based filtering

# Create a DataFrame for song similarities
song_similarity_df = pd.DataFrame(song_similarity, index=user_item_matrix.columns, columns=user_item_matr

# Step 4: Function to recommend songs for a specific user
def recommend_songs(user_id, top_n=3):
    # Get the user's ratings
    user_ratings = user_item_matrix.loc[user_id]

    # Find songs rated by the user
    rated_songs = user_ratings[user_ratings > 0].index.tolist()

    # Predict ratings for all songs not rated by the user
    predicted_ratings = {}
    for song_id in user_item_matrix.columns:
        if song_id not in rated_songs:
            # Get similar songs to the ones the user rated
            similar_songs = song_similarity_df[song_id]
            predicted_rating = sum(user_ratings[rated_song] * similar_songs[rated_song] for rated_song in
            predicted_ratings[song_id] = predicted_rating

    # Sort the predicted ratings for all unrated songs
    recommended_song_ids = sorted(predicted_ratings, key=predicted_ratings.get, reverse=True)[:top_n]

    # Get song titles for the recommended song IDs
    recommended_songs = songs[songs['SongID'].isin(recommended_song_ids)]

    return recommended_songs
```

```python
# Step 5: Test the recommender system with a user (e.g., UserID = 3)
user_id = 3
recommended_songs = recommend_songs(user_id)

# Print the recommended songs
print(f"Recommended songs for User {user_id}:")
print(recommended_songs[['SongID', 'Title']])
```

```
Recommended movies for User 3:
   MovieID           Title
0        1  The Dark Knight
```

GitHub File Upload: