EX NO.:3

DATE:

Implement user-profile learning.

AIM:

To implement user-profile learning.

ALGORITHM:

1. Import the necessary libraries
2. Get the sample user-data
3. Convert the user-data into matrix
4. Calculate the cosine similarity between users
5. Perform functions to get the personalized recommendations for a given user
6. Find the highest similarity
7. Aggregate items liked by similar users
8. Remove the items already liked by user
9. Print the recommendations

PROGRAM:

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
# Sample user-item interaction data (replace this with your data)
user_item_data =
{
    'user1': 'item1 item2 item4',
    'user2': 'item2 item3 item5',
    'user3': 'item1 item3 item4',
    'user4': 'item2 item4 item5',
}
```

```
# Convert the user-item interactions into a matrix representation (Bag-of-Words)
vectorizer = CountVectorizer(binary=True)
user_item_matrix = vectorizer.fit_transform(user_item_data.values())
# Calculate cosine similarity between users
user_similarity_matrix = cosine_similarity(user_item_matrix, user_item_matrix)
# Function to get personalized recommendations for a given user
def get_recommendations(user, user_similarity_matrix, user_item_data, n_recommendations=2):
    user_index = list(user_item_data.keys()).index(user)
    similarities = user_similarity_matrix[user_index]
        # Find the indices of users with highest similarity (excluding the user itself)
    similar_users_indices = np.argsort(similarities)[::-1][1:n_recommendations+1]
    # Aggregate items liked by similar users
    recommended_items = set()
    for index in similar_users_indices:
        items = user_item_data[list(user_item_data.keys())[index]].split()
        recommended_items.update(items)
    # Remove items already liked by the user
    user_items = user_item_data[user].split()
    recommended_items -= set(user_items)
    return recommended_items
# Example: Get recommendations for 'user1'
user_to_recommend = 'user1'
recommendations = get_recommendations(user_to_recommend, user_similarity_matrix, user_item_data)
print(f"Recommendations for {user_to_recommend}: {recommendations}")
```

OUTPUT:

Recommendations for user1: {'item5', 'item3'}

RESULT:

Thus the implementation of user-profile learning. was executed successfully.