

**EX.NO.:5**

**DATE:**

**Implement collaborative filter techniques.**

---

**Aim:**

To implement collaborative filter techniques.

**ALGORITHM:**

**User-based collaborative filtering:-**

1. Import the necessary libraries
2. Sample user-item rating data
3. Create a user-item matrix
4. Calculate cosine similarity between users
5. Perform to get collaborative filtering recommendations for a user ( Users with highest similarity, items that the similar users liked and the current user hasn't)
6. Print User-based recommendations

**Item-based collaborative filtering: -**

1. Calculate cosine similarity between items
2. Perform the function to get item-based collaborative filtering recommendations (Find items that user has not rated, calculate the average weighted average of item ratings, sort items by predicted score, get the top recommendations)
3. Print Item-based recommendations

**PROGRAM:**

**User-based collaborative filtering:-**

```
import pandas as pd

from sklearn.metrics.pairwise import cosine_similarity
```

**# Sample user-item rating data (replace this with your data)**

```
ratings_data = pd.DataFrame({  
    'user_id': [1, 1, 2, 2, 3, 3, 4, 4],  
    'item_id': [1, 2, 2, 3, 3, 4, 1, 4],  
    'rating': [5, 4, 5, 3, 4, 2, 3, 5]  
})
```

**# Create a user-item matrix**

```
user_item_matrix = ratings_data.pivot_table(index='user_id', columns='item_id', values='rating',  
fill_value=0)
```

**# Calculate cosine similarity between users**

```
user_similarity_matrix = cosine_similarity(user_item_matrix)
```

**# Function to get collaborative filtering recommendations for a user**

```
def get_user_based_recommendations(user_id, user_item_matrix, user_similarity_matrix,  
n_recommendations=2):
```

```
    user_index = user_id - 1 # Adjust index to start from 0
```

```
    similarities = user_similarity_matrix[user_index]
```

**# Find the indices of users with highest similarity (excluding the user itself)**

```
    similar_users_indices = similarities.argsort()[::-1][1:n_recommendations+1]
```

**# Get items that the similar users liked and the current user hasn't**

```
    recommended_items = []
```

```
    for index in similar_users_indices:
```

```
        liked_items = user_item_matrix.iloc[index][user_item_matrix.iloc[index] > 0].index
```

```
        user_items = user_item_matrix.iloc[user_index][user_item_matrix.iloc[user_index] > 0].index
```

```
        new_items = set(liked_items) - set(user_items)
```

```
        recommended_items.extend(new_items)
```

```
    return recommended_items[:n_recommendations]
```

### **# Example: Get user-based collaborative filtering recommendations for user 1**

```
user_id_to_recommend = 1

user_based_recommendations = get_user_based_recommendations(user_id_to_recommend,
user_item_matrix, user_similarity_matrix)

print(f"User-Based Collaborative Filtering Recommendations for User {user_id_to_recommend}:
{user_based_recommendations}")
```

### **Item-based collaborative filtering:-**

#### **# Calculate cosine similarity between items**

```
item_similarity_matrix = cosine_similarity(user_item_matrix.T)
```

#### **# Function to get item-based collaborative filtering recommendations for a user**

```
def get_item_based_recommendations(user_id, user_item_matrix, item_similarity_matrix,
n_recommendations=2):
```

```
    user_index = user_id - 1 # Adjust index to start from 0
```

#### **# Find items that the user has not rated**

```
unrated_items = user_item_matrix.columns[user_item_matrix.iloc[user_index] == 0]
```

#### **# Calculate the weighted average of item ratings based on similarity**

```
    item_scores = item_similarity_matrix.T.dot(user_item_matrix.iloc[user_index]) /
(item_similarity_matrix.T.dot(user_item_matrix.iloc[user_index].abs()) + 1e-10)
```

#### **# Sort items by predicted score and get top recommendations**

```
    recommended_items = item_scores[unrated_items]
```

```
    return recommended_items
```

### **# Example: Get item-based collaborative filtering recommendations for user 1**

```
item_based_recommendations = get_item_based_recommendations(user_id_to_recommend,
user_item_matrix, item_similarity_matrix)

print(f"Item-Based Collaborative Filtering Recommendations for User {user_id_to_recommend}:
{item_based_recommendations}")
```

## **OUTPUT:**

### **User-based collaborative filtering: -**

User-Based Collaborative Filtering Recommendations for User 1: [3, 4]

### **Item-based collaborative filtering: -**

Item-Based Collaborative Filtering Recommendations for User 1: [1]

## **RESULT:**

Thus the implementation of collaborative filter techniques was executed successfully.