|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Kingdom of Saudi Arabia**  **Ministry of Education**  **University of Jeddah**  **College of Computer Science and Engineering**  **Department of Computer Science and Artificial Intelligence** | Logo, company name  Description automatically generated | **المملكة العربية السعودية**  **وزارة التعليم**  **جامعة جدّة**  **كلية علوم وهندسة الحاسب**  **قسم علوم الحاسب والذكاء الاصطناعي** | |  |  |

|  |
| --- |
| **Lab 3** |
| **CCAI 422 Recommender Systems** |
| **Second semester 2023/2024**   |  |  | | --- | --- | | **Lab Date/Time: 13-2-2024.** |  | | **Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | |

|  |  |
| --- | --- |
| **Instructor Name** | **Section** |
|  |  |

**Instructions**:

The lab must be completed in the lab individually

Student should invest in understanding the lab rather than just copying and pasting the code

This lab will be part of the lab final exam

|  |  |
| --- | --- |
| **PLO/CLO** | **SO** |
| **PLO S2 (CLO 2):** Demonstrate the ability of applying tools, techniques and practices required for problem solving in the domain of recommender systems | **SO 2:** Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline |

|  |  |  |
| --- | --- | --- |
|  | **Max Score** | **Student Score** |
| **PLO S2 / CLO 2 / SO 2** | **5** |  |
| **Total** |  |  |

**User-based Recommender**

**Introduction and Lab Description**

This lab contains a tutorial showing the implementation of a user-based recommender using the related python libraries. It is an important task because it allows students to familiarize themselves with and prepare for the lab final exam.

**Data Description**

Each student has access to a data file which will be used for the task in this lab:

* *Part-1 data:* “ccai422\_lab03\_part1\_data.csv” contains 100,000 data points. Specifically, it contains the ratings of 943 users to 1682 items (movies in this data). The average number of ratings per user is about 106 ratings. This means a random user has on average about 1576 unrated items (1682-106).

**Grading**

In this lab, students will not submit anything on the blackboard. The grading will be completed in the lab. The assessment will be individually, students can always ask for help and clarifications if needed.

**Tasks: [PLO S2 / CLO 2 / SO 2] [5 marks]**

Please complete the below tutorial:

1. Import the libraries

import pandas as pd  
import numpy as np

1. Load the dataset into a datafrme

*# Use the read\_csv function to store the dataset into a dataframe called ratings*  
ratings = pd.read\_csv('ccai422\_lab03\_part1\_data.csv')  
  
*# Keep only the columns that we need in this task: "user\_id", "movie\_id" and "rating"*  
ratings = ratings[["user\_id", "movie\_id", "rating"]]

1. Explore the data

*# The total number of data points*  
print('The number of data points in this dataset: ' + str(len(ratings)))  
  
*# The number of items (i.e. movies) in the dataset*  
print('The number of items (i.e. movies) in the dataset: ' + str(ratings['movie\_id'].nunique()))  
  
*# The number of users in the dataset*  
print('The number of users in the dataset: ' + str(ratings['user\_id'].nunique()))  
  
*# The average ratings per user*  
ratings\_per\_users = ratings.groupby('user\_id').count()  
print('The average ratings per user: '+ str(round(ratings\_per\_users.mean()[0],2)))  
  
*# The number of ratings/user*  
print('The below table shows the number of ratings per user\n')  
print(ratings\_per\_users)

1. Rating matrix

*#Build the ratings matrix using pivot\_table function*  
r\_matrix = ratings.pivot\_table(values='rating', index='user\_id', columns='movie\_id')  
  
*#Create a dummy ratings matrix which will have all null values imputed to 0*  
r\_matrix\_dummy = r\_matrix.copy()  
  
*# rename the axis of the new matrix*  
r\_matrix\_dummy = r\_matrix\_dummy.rename\_axis('user\_id', axis=1).rename\_axis(None, axis=0)  
  
*# Impute all the NaN values to 0*  
r\_matrix\_dummy = r\_matrix\_dummy.fillna(0)  
r\_matrix\_dummy.head()  
  
r\_matrix\_dummy = r\_matrix\_dummy.fillna(0)  
r\_matrix\_dummy.head()

1. Compute the correlation

*#Get the transpose of rating matrix to compute the pearson correlation between the users not the items*  
users\_rating\_matrix = r\_matrix\_dummy.T  
  
*# Use the corr function of pandas to compute the pearson correlation on the users\_ratings\_matrix*  
pearson\_sim = users\_rating\_matrix.corr()

1. Retrieve the prediction variables

*# Randomly select a user to rate an item for him/her*  
userX= 5  
  
*# Get the rating data of the specified user and compute the mean value*  
rXmean=r\_matrix[userX].mean()  
  
*# Specify the n neighbors to be used in the prediction*  
n = 2  
  
*# Retrieve the top n based on the pearson sim (ignore the first one since it is the item with itself)*  
topn = pearson\_sim[[userX]].nlargest(n+1,userX).index.tolist()[1:]  
  
*# Retrieve the similarity values to be used*  
neighbors\_sim = pearson\_sim[[userX]].nlargest(n+1,userX)[1:]  
  
*# Get the rating data for the top n neighbors*  
r\_matrix\_topn = r\_matrix[topn]  
  
*# Compute the mean rating's value per neighbor*  
neighbors\_means = r\_matrix\_topn.mean()  
  
*# Compute the differences between the mean rating's value per user and his/her actual ratings*  
averaged\_neighbors\_ratings = r\_matrix\_topn.sub(neighbors\_means,axis=1)

1. Unrated items for the target user

*# Select all unrated items for the target user*  
unrated\_target = r\_matrix[r\_matrix[userX].isna()][topn]  
  
*# rename the axis of the unrated item matrix*  
unrated\_target = unrated\_target.rename\_axis('movie\_id', axis=1).rename\_axis(None, axis=0)  
  
*# Remove items that are not rated by all top n neighbors*  
unrated\_target.dropna(axis = 0, how = 'all', inplace = True)  
  
  
unrated\_target.head()

1. Predict the rating

*# Randomly select the item to be rated*  
itemX = 7  
  
*# Predict the rating value for the unrated item*  
predicted\_value = rXmean + ((neighbors\_sim.T.dot(averaged\_neighbors\_ratings.loc[itemX].T).values[0]) / neighbors\_sim.sum())