**PROJECT STEP 2: DATA ANALYSIS**

**COURSE PROJECT**

**KANISHK KUMAR**

**IFT 511: ANALYZING BIG DATA**

**PROF. ASMAA ELBADRAWY**

**DATE: 01st DECEMBER 2024**

**TASK 1: BUILDING A RECOMMENDER SYSTEM**

**COLLABORATIVE FILTERING WITH PROXIMITY CALCULATIONS**

**CODE:**

from sklearn.datasets import load\_svmlight\_file

from sklearn.neighbors import NearestNeighbors

import numpy as np

import pandas as pd

libsvm\_file\_1311 = 'Ratings\_final\_output.libsvm'

sparse\_matrix\_1311, \_ = load\_svmlight\_file(libsvm\_file\_1311)

knn\_1311 = NearestNeighbors(metric='cosine', algorithm='brute')

knn\_1311.fit(sparse\_matrix\_1311)

def recommend\_books\_for\_all\_users\_no\_metadata\_1311(k\_1311=10,

top\_n\_1311=5,

output\_file\_1311="recommendations\_all\_users\_1311.csv"):

"""

Generate book recommendations for all users in the dataset without metadata and save results to a CSV file.

Parameters:

k\_1311 (int): Number of similar users to consider.

top\_n\_1311 (int): Number of top recommendations to return for each user.

output\_file\_1311 (str): Path to save the recommendations CSV file.

Returns:

None

"""

all\_recommendations\_1311 = []

num\_users\_1311 = sparse\_matrix\_1311.shape[0]

for user\_id\_1311 in range(1, num\_users\_1311 + 1):

zero\_indexed\_user\_id\_1311 = user\_id\_1311 - 1

distances\_1311, indices\_1311 = knn\_1311.kneighbors(sparse\_matrix\_1311[zero\_indexed\_user\_id\_1311], n\_neighbors=k\_1311 + 1)

similar\_users\_1311 = indices\_1311.flatten()[1:]

similarity\_scores\_1311 = 1 - distances\_1311.flatten()[1:]

similar\_users\_matrix\_1311 = sparse\_matrix\_1311[similar\_users\_1311].toarray()

weighted\_ratings\_1311 = np.dot(similar\_users\_matrix\_1311.T, similarity\_scores\_1311) / (similarity\_scores\_1311.sum() + 1e-8)

user\_rated\_books\_1311 = sparse\_matrix\_1311[zero\_indexed\_user\_id\_1311].toarray().flatten() > 0

weighted\_ratings\_1311[user\_rated\_books\_1311] = 0

recommended\_indices\_1311 = np.argsort(-weighted\_ratings\_1311)[:top\_n\_1311]

recommendation\_scores\_1311 = weighted\_ratings\_1311[recommended\_indices\_1311]

for idx\_1311, score\_1311 in zip(recommended\_indices\_1311, recommendation\_scores\_1311):

all\_recommendations\_1311.append({

"User\_ID": user\_id\_1311,

"Book\_ID": idx\_1311,

"Recommendation\_Score": score\_1311

})

if user\_id\_1311 % 1000 == 0:

print(f"Processed {user\_id\_1311}/{num\_users\_1311} users.")

pd.DataFrame(all\_recommendations\_1311).to\_csv(output\_file\_1311, index=False)

print(f"All recommendations saved to {output\_file\_1311}")

recommend\_books\_for\_all\_users\_no\_metadata\_1311(

k\_1311=10,

top\_n\_1311=5,

output\_file\_1311="recommendations\_all\_users.csv"

)

**WHAT THE CODE DOES**

The code above is used to recommend 5 books to different user, using the libSVM file that we generated in task 1 using the sparse matrix.

The first part is where we upload the file using the load\_svmlight\_file. Then use the cosine similarity for finding the nearest neighbor.

The recommendation logic:

1. For each user, find the nearest neighbor.
2. Aggregate the similarity score.
3. Exclude the books that the users have already read.
4. Recommend the top 5 books for every user

Save the CSV file as “Recommend\_all\_user”.

**CODE SCREENSHOT**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**OUTPUT**

A screenshot of a computer

Description automatically generated

SELECTED FILE ON THE JUPYTER HOME SCREEN IS THE GENERATED CSV FILE THAT RECOMMENDS BOOK FOR ALL THE USERS USING THE LIBSVM FILE THAT WE CREATED IN PHASE 1

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The CSV file recommends the best 5 books for each user, and all provides the recommended store for each book.