**“Code:”**

import math

from collections import defaultdict

from tqdm import tqdm

def load\_data():

"""Load and process book and rating data"""

print("Loading book details...")

isbn\_to\_title = {}

book\_id\_to\_isbn = {}

# Read book data from Books.csv and map ISBNs to titles

with open('Downloads/archive (1)/Books.csv', 'r', encoding='utf-8') as file:

next(file) # Skip the header row

for idx, line in enumerate(file, start=1):

try:

parts = line.strip().split(';')

isbn = parts[0].strip()

title = parts[1].strip()

isbn\_to\_title[isbn] = title

book\_id\_to\_isbn[idx] = isbn

except Exception as error:

print(f"Error reading line: {line}, Error: {error}")

continue

print("Loading user ratings from LIBSVM format...")

user\_ratings = defaultdict(dict)

# Read user ratings from the ratings.libsvm file

with open('Downloads/user\_book\_ratings\_custom (2).libsvm', 'r') as file:

for user\_id, line in enumerate(file, start=1):

ratings = line.strip().split()

for rating in ratings:

try:

if ':' not in rating:

continue

book\_id, score = map(float, rating.split(':'))

user\_ratings[user\_id][int(book\_id)] = score

except ValueError as error:

print(f"Invalid rating skipped: {rating}, Error: {error}")

continue

# Calculate user norms (used for cosine similarity)

print("Calculating user norms...")

user\_norms = {

user: math.sqrt(sum(rating \*\* 2 for rating in ratings.values()))

for user, ratings in user\_ratings.items()

if sum(rating \*\* 2 for rating in ratings.values()) > 0

}

return user\_ratings, isbn\_to\_title, user\_norms, book\_id\_to\_isbn

def compute\_similarity(user1, user2, user\_ratings, user\_norms):

"""Calculate cosine similarity between two users"""

if user1 not in user\_norms or user2 not in user\_norms:

return 0.0

shared\_books = set(user\_ratings[user1].keys()) & set(user\_ratings[user2].keys())

if not shared\_books:

return 0.0

dot\_product = sum(user\_ratings[user1][book] \* user\_ratings[user2][book] for book in shared\_books)

denominator = user\_norms[user1] \* user\_norms[user2]

return dot\_product / denominator if denominator > 0 else 0.0

def generate\_recommendations(target\_user, user\_ratings, user\_norms, k=10):

"""Recommend top 5 books for a user based on similar users"""

similarity\_scores = []

for other\_user in user\_ratings:

if other\_user != target\_user:

sim = compute\_similarity(target\_user, other\_user, user\_ratings, user\_norms)

if sim > 0:

similarity\_scores.append((other\_user, sim))

top\_similar\_users = sorted(similarity\_scores, key=lambda x: x[1], reverse=True)[:k]

if not top\_similar\_users:

return []

target\_user\_books = set(user\_ratings[target\_user].keys())

recommendations = {}

for similar\_user, similarity in top\_similar\_users:

for book in user\_ratings[similar\_user]:

if book not in target\_user\_books:

numerator = sum(

user\_ratings[sim\_user][book] \* sim for sim\_user, sim in top\_similar\_users if book in user\_ratings[sim\_user]

)

denominator = sum(sim for \_, sim in top\_similar\_users)

if denominator > 0:

recommendations[book] = numerator / denominator

return sorted(recommendations.items(), key=lambda x: x[1], reverse=True)[:5]

def main():

"""Main function to generate recommendations"""

user\_ratings, isbn\_to\_title, user\_norms, book\_id\_to\_isbn = load\_data()

print("Generating recommendations for all users...")

total\_users = len(user\_ratings)

with open('recommendationsFILE.csv', 'w', encoding='utf-8', newline='') as file:

file.write('User\_ID,Book\_ID,Book\_Title,Recommendation\_Score\n')

for user in tqdm(user\_ratings.keys(), total=total\_users, desc="Processing users"):

recommendations = generate\_recommendations(user, user\_ratings, user\_norms)

for book\_id, score in recommendations:

title = isbn\_to\_title.get(book\_id\_to\_isbn.get(book\_id, str(book\_id)), f"Book\_{book\_id}")

scaled\_score = min(max(round(score \* 2), 1), 10)

file.write(f'{user},{book\_id},"{title}",{scaled\_score}\n')

print("Recommendations successfully generated!")

if \_name\_ == "\_main\_":

main()

**“Code Screenshots:”**

A computer screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**“CSV File Screenshots:”**

**A screenshot of a computer

Description automatically generated**

**“Explanation of the Above Results”**

This project develops a user-based collaborative filtering recommendation system to provide personalized book suggestions. It processes book information from Books.csv and user rating data from user\_book\_ratings\_custom (2).libsvm. Using cosine similarity, the system identifies users with similar reading preferences and predicts ratings for unrated books based on input from these similar users.

The system generates recommendations by combining weighted ratings from similar users, scaling scores between 1 and 10. The results, saved in recommendationsFILE.csv, include user IDs, book titles, and corresponding recommendation scores, providing up to five personalized book suggestions per user.

Designed to handle diverse user preferences, the system is scalable and adaptable for larger datasets. It enhances user engagement by offering relevant recommendations and promoting book discovery. Future enhancements could incorporate user feedback to refine recommendations or include additional attributes like genres to improve personalization. Overall, the project showcases an effective implementation of collaborative filtering to create a versatile and user-focused recommendation engine.

Here are the top five books with the highest recommendation scores from the output:

1. **"Death of a DJ: A Mystery"**
   * **Recommendation Score:** 10
   * **User ID:** 115
2. **"Iron Chef: The Official Book"**
   * **Recommendation Score:** 10
   * **User ID:** 115
3. **"The Voyage of Qv. 66"**
   * **Recommendation Score:** 10
   * **User ID:** 117
4. **"Dorothy Parker: What Fresh Hell Is This"**
   * **Recommendation Score:** 10
   * **User ID:** 121
5. **"Quotation Quizzlers: Puzzling Your Way Through Famous Quotations"**
   * **Recommendation Score:** 10
   * **User ID:** 117

These books all have the maximum recommendation score of 10, indicating they are highly relevant and recommended for the respective users.