Building a Memory-Based Book Recommendation System Using Pearson Collaborative Filtering

Section 1. Homework Overview

The objective of this project is to build a complete memory-based recommendation system for books using **collaborative filtering based exclusively on the Pearson correlation coefficient**, as described on the official Wikipedia page for memory-based collaborative filtering (https://en.wikipedia.org/wiki/Collaborative_filtering).

You will work with a Book rating dataset (Books.csv) and build a fully functional recommender that can produce meaningful book recommendations both at the **user level** and at the **book level**.

The project is divided into two major components:

1. **Data Cleaning and Preparation**
2. **Collaborative Filtering and Recommendation Generation**

All work must follow the exercise requirements described below. No external machine-learning models, KNN libraries, cosine similarity, or hybrid methods may be used. Only memory-based collaborative filtering with Pearson correlation is allowed.

At the end of the project, you will submit the cleaned datasets, your similarity calculations, your recommendations, and written explanations supporting each decision and method used in a notebook (**cells with no output won't be graded**).

NOTE: All questions must be answered by using Spark Dataframes, DON'T use pandas

SECTION 2 : Data Cleaning and Preparation

Your objective is to produce clean dataframes from the raw input data ready for collaborative filtering using Pearson correlation.

This process must be completed **before** any collaborative filtering is attempted.
All cleaning must be preceded by inspection and followed by verification.

At the end of this section, you must build the following Data Frames:

- a cleaned interaction dataframe named **`ratings_clean`**,
    - Only valid `(User-ID, ISBN, Rating)` triples
    - Users with enough ratings (≥5 ratings)
    - Books with enough ratings ≥5 ratings
    - Cleaned rating values ∈ [0,10]
    - No malformed ISBNs

- a cleaned metadata dataframe named **`books_clean`**:
    - Only books corresponding to ISBNs present in `ratings_clean`
    - Clean ISBN
    - Clean and usable metadata (Title, Author)
- a list of active users named **`users_active`**.

Exercise 1 : Inspect and clean the Raw Data

You must inspect the data before each cleaning step and verify the results afterwards.

Load the data from Books.csv, examine it thoroughly and clean it. Note: ratings should be between 0 and 10, a rating with a value of 0 means “no rating”, your recommendation algorithm should ignore 0 values.

1. ISBN is the **primary key** for books. If it is wrong, ratings for the same book look like different items. Inspect the ISBN values, to detect any inconsistencies. Before cleaning, inspect some examples. Clean ISBN values so they only contain numerical values. Drop ISBNs that are **empty** after cleaning or that don’t contain any numbers. Drop ISBNs that occur **fewer than 2 times.**

2. Remove demographics: This project does *not* use demographics, keep only user ids.

3. Keep only **active users**: An *active user* is a user who has rated **enough books** to be useful (user similarity requires **overlap** between users). We often filter users based on number of ratings:
    - 0 ratings → useless

- 1 rating → cannot compute meaningful similarities
- at least 5 ratings → considered “active”

If only 1,000 users have >5 ratings, your model should focus on these 1,000.

4. Among the active users, keep only users that have rated at least 1 book in common with at least 1 other user.

5. Inspect and remove entries with invalid ratings (e.g., negative, 100, NaN, text). Convert all ratings to float (if not already done).

6. Remove the books with not enough ratings (less than 5) or with unusually high/low ratings. Unusually high ratings means books with extremely high average score (e.g., 10/10) but very few ratings (e.g., 2 people). Unusually low ratings means: books with extremely low average score (0–1) often because only a handful of users rated them. Compute the number of ratings per ISBN: inspect the distribution, analyze how many books have 0, 1, 2, ... ratings.

7. Remove duplicate books. The dataset may contain many editions of the same book. This affects recommendations: the **same book** in multiple editions might be recommended several times.

8. Remove duplicates [User Id, ISBN]: users that have rated the same book several times. If a user rated the same book twice, keep the **highest rating** or the **latest** (justify choice)

9. Inspect and clean titles/authors: Search for common issues: missing titles, “Unknown” authors, whitespaces or inconsistent formatting. Replace missing titles with `"Unknown Title",` replace missing authors with `"Unknown Author".` Strip whitespaces.

Exercise 2 : Final Inspection

You must confirm:

- No ratings outside [0,10]
- No users with < 5 ratings
- No books with < 5 ratings

- all ISBN values are properly formatted, non-empty strings
- No duplicated `(User-ID, ISBN)` rows
- Clean titles/authors
- confirm that the number of valid books in `books_clean` matches the number of ISBNs in `ratings_clean`

Write a summary explaining:

- how many users remain
- how many books remain
- how many ratings remain
- what percentage of the original data was removed
- why each cleaning rule was important for collaborative filtering

SECTION 3 : Data Cleaning and Preparation

Using the cleaned dataframes from the previous section, you will now implement user-based and item-based collaborative filtering following the mathematical definitions of Pearson correlation and memory-based recommendation.

Exercise 9: Compute Pearson Similarity Between Users

Using `ratings_clean`, inspect and justify how you represent missing ratings.
Verify that the dimensions correspond to the final set of users and ISBNs from Section 2.

9.1 Compute similarity between the target user and all others

Using **Pearson correlation**, for each user u find the **k** most similar users. These are the **"neighbors" of u**.

Pearson correlation

Pearson correlation measures **how strongly two users' rating behaviors move together**.

It takes into account:

- rating magnitude
- rating direction (likes vs dislikes)

- rating style (strict vs generous raters)

Formula you will implement:

$$\text{sim}(u,v) = \frac{\sum_i (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_i (r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_i (r_{v,i} - \bar{r}_v)^2}}$$

Where:

- $r_{u,i}$ = rating of user u for book i
- $\bar{r}_u$ = average rating of user u
- Summation runs over books that **both users rated**

The Full Similarity Vector

For a user u, a **similarity vector** is:

[sim(u,1), sim(u,2), sim(u,3),...]

A list of similarity scores between user u and *every* other user. This vector is used in rating prediction.

Full Similarity Matrix

Compute a full matrix:

- rows = users
- columns = users
- value = Pearson similarity

Exercise 10: Predict ratings for books the target user has not read

For a chosen target user chosen, identify the books this user has not rated.

For each unread book, identify neighbors (the k most similar ones) who have rated it and compute a predicted rating using the weighted Pearson-based formula.

Experiment with several values of k.

Prediction formula:

$$\hat{r}_{u,j} = \bar{r}_u + \frac{\sum_{v \in N} sim(u,v)\,(r_{v,j} - \bar{r}_v)}{\sum_{v \in N} |sim(u,v)|}$$

Where:

- u: target user
- v: neighbor user (as defined previously). Only consider the k most similar users
- j: book to predict

Recommend top-N books

- Keep only books user u has not rated
- Predict ratings for these books
- Recommend the highest predicted scores

Exercise 11: Compute Pearson Similarity Between Books (Item–Item Collaborative Filtering)

Select a target book and examine its rating vector across users. Compute the Pearson similarity between this book and all other books, considering only books that share enough common raters. Produce a similarity vector for the target book and store all the similarity vectors into a book similarity vector.

Exercise 12: Produce “Because you liked book X, you may like book Y” recommendations

For a chosen book:

- find top similar books (the k most similar books, experiment with several k values)
- return highest correlated books

SECTION 4: Final Deliverables

By the end of this project, you must submit:

1. A notebook containing your answers to all exercises, including all inspections, analyses, justifications, and interpretations (code cells with empty output will not be graded) and the following:

    a. The cleaned datasets : `ratings_clean`, `books_clean`, and `users_active`.

    b. The user and item matrices used for collaborative filtering.

    c. The recommendation lists generated for both user-user and item-item collaborative filtering methods.