

**Book Recommendation System– Case Study**

**Machine Learning Techniques**

**CSD-3154  
Mini-Project**

**Done by M.Ajay**

**220171601012**

**B.TECH AI & DS “A” SECTION**

**ABSTRACT**

The rapid growth of digital platforms has led to an explosion in the availability of books across countless genres and categories. While this abundance offers readers a vast selection, it also poses a significant challenge: finding books that align with individual preferences and tastes. Traditional methods of exploring books, such as browsing physical libraries or reading reviews, often fall short in efficiently addressing this challenge.

To bridge this gap, this project develops a **genre-specific book recommendation system** powered by the **K-Nearest Neighbors (KNN) algorithm**, implemented within a **Google Colab** environment. The system identifies and suggests books based on their similarity to others within the same genre, as determined by user ratings and associated features.

The implementation begins with preprocessing a dataset containing book metadata, calculating genre-specific average ratings, and encoding genres into machine-readable formats. These steps culminate in the application of the KNN algorithm, which analyzes feature distances between books to generate high-quality recommendations. The system's output is tailored specifically to the user’s preferred genre, ensuring relevance and precision.

This work not only highlights the application of machine learning in personalized recommendations but also demonstrates the utility of cloud-based tools like **Google Colab** for building accessible and scalable solutions. By addressing the complexity of choice in modern literature, the project makes a significant contribution to enhancing the reading experience for book enthusiasts.

**INTRODUCTION**

The overwhelming availability of content in today's digital landscape has necessitated the development of recommendation systems, which play a critical role in helping users navigate extensive choices. From streaming platforms suggesting movies to e-commerce websites recommending products, these systems have become integral in simplifying decision-making and enhancing user satisfaction.

The domain of literature and books is no exception. Readers often seek personalized recommendations to discover titles that match their tastes, but the sheer volume of options spanning diverse genres and styles makes this a daunting task. Traditional discovery methods—such as browsing bestseller lists or relying on peer recommendations—are neither scalable nor tailored enough to cater to individual preferences.

Recognizing this gap, this project introduces a machine learning-based **genre-specific book recommendation system** that leverages the capabilities of the **K-Nearest Neighbors (KNN) algorithm**. This system focuses on identifying books that share similar characteristics within a user-specified genre, offering a tailored reading list that aligns closely with individual interests.

Unlike generic recommendation systems that often rely on collaborative filtering or simple popularity metrics, the approach taken here incorporates **content-based filtering** to prioritize books based on their genre-specific attributes. The use of **Google Colab** as the implementation platform further enhances the project's accessibility, allowing for real-time experimentation, collaboration, and deployment.

By focusing on genre specificity, the project emphasizes the importance of nuanced recommendations that account for individual reader preferences. This work aims to make book discovery a seamless and enjoyable process, empowering users to explore new titles while staying true to their unique literary tastes.

**IMPLEMENTATION OVERVIEW**

**1. Data Preparation**

Data preparation is a cornerstone of any machine learning project, particularly for recommendation systems where the quality of input data directly influences the effectiveness of the output.

In this project, the dataset includes essential details such as **Book\_ID**, **Genre**, and **User Ratings**, which form the foundation for generating recommendations. To ensure data integrity and usability, the system performs several preprocessing steps:

* **Handling Missing Data**: Missing or null values are addressed by either filling them with suitable substitutes (e.g., average ratings) or removing incomplete entries to prevent biases.
* **Converting Data Types**: Columns such as Book\_ID and Average\_Rating are converted to numerical formats to enable mathematical operations required for machine learning algorithms.
* **Grouping by Genre**: Ratings are grouped by **Book\_ID** and **Genre** to calculate the **Average\_Rating** for each book within its respective genre. This step ensures that the system captures an accurate representation of user sentiment for books in different genres.
* **Data Normalization**: The dataset is normalized to ensure that features such as ratings and dummy-encoded genres contribute equally to the similarity calculations during KNN.

These preprocessing steps result in a clean, structured dataset that provides a strong foundation for subsequent machine learning tasks.

**2. Feature Engineering**

Feature engineering is critical to converting raw data into actionable inputs for machine learning algorithms. In this project, the feature matrix (X) includes:

1. **Book\_ID**: A unique identifier for each book, excluded from similarity calculations but retained for mapping recommendations to their titles.
2. **Average\_Rating**: A numerical representation of a book's popularity and quality, serving as a key metric for recommendation.
3. **Genre Encoding**: To incorporate genre information, the project employs **one-hot encoding**, a method that converts categorical data (genres) into binary columns. For instance, a book in the "Fantasy" genre would have a 1 in the "Fantasy" column and 0 in others.

The resulting feature matrix provides a comprehensive representation of each book, combining its average rating with its genre attributes. This matrix enables the KNN algorithm to identify books that are both highly rated and genre-relevant.

**3. K-Nearest Neighbors (KNN) Algorithm**

The **K-Nearest Neighbors (KNN)** algorithm is a widely used machine learning model known for its simplicity and interpretability. In this system, KNN is applied as follows:

1. **Similarity Calculation**: The algorithm calculates the Euclidean distance between books in the feature space. Books with shorter distances are deemed more similar.
2. **Model Configuration**: The KNN model is configured with n\_neighbors=5, indicating that the system identifies the five closest matches for each book.
3. **Genre-Specific Filtering**: The system ensures that recommendations are restricted to the user’s selected genre, enhancing the relevance of the output.

The trained model efficiently identifies books that align closely with a user’s preferences, generating a curated list of recommendations.

**4. Recommendation Process**

The recommendation process involves several key steps:

1. **Filtering by Genre**: Books in the user-selected genre are isolated for analysis.
2. **Neighbor Identification**: For each book in the filtered dataset, the KNN algorithm identifies the top 5 most similar books based on feature proximity.
3. **Duplicate Removal and Sorting**: The recommendations are consolidated into a unique list, sorted by **Average\_Rating** in descending order to prioritize high-quality books.

This structured approach ensures that users receive recommendations that are both relevant and well-rated, enhancing the overall utility of the system.

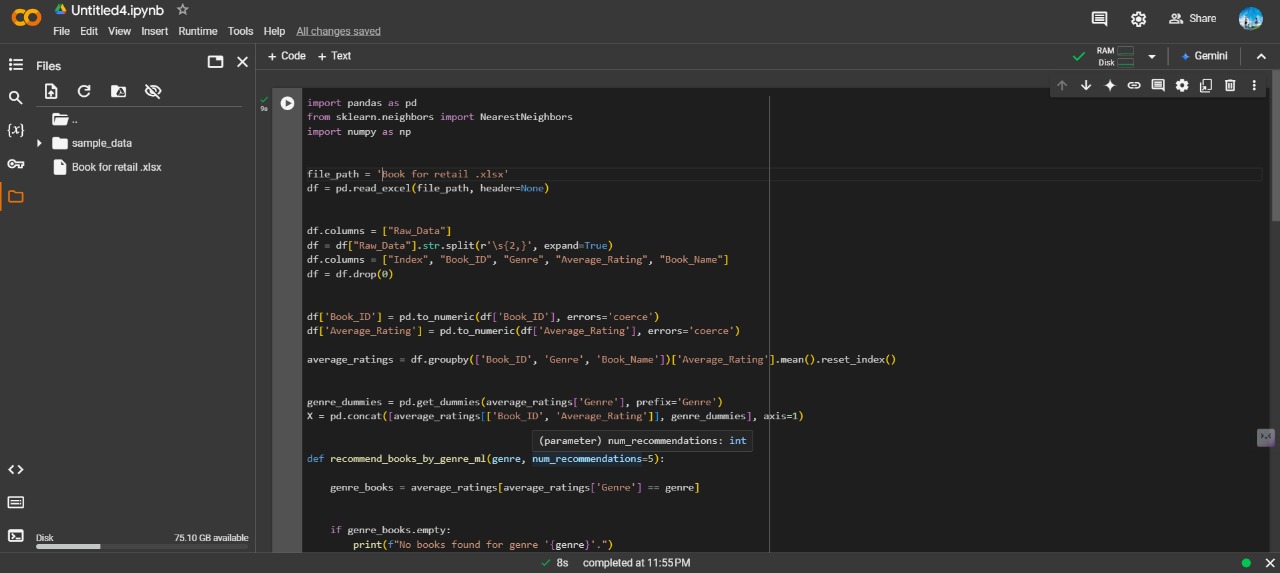


Fig 1. The Above picture shows the implementation of the Code using KNN Algorithm in Google Colab, the Directory is saved into the Drive, which is accessed by the Colab in order to perform the ML Algorithm

**Output :**

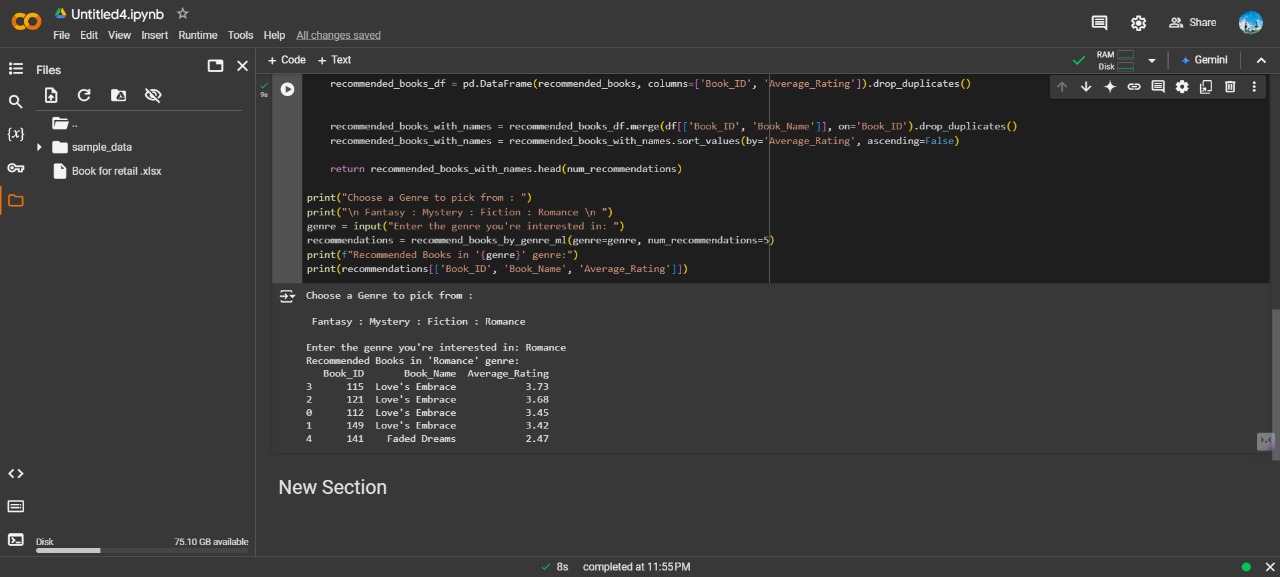


Fig 2. Thus the user-inputs the necessary tags of book genres into the input, and the result shows multiple genres of recommendations based on Customer Reviews

**RESULTS**

The genre-specific book recommendation system implemented using the K-Nearest Neighbors (KNN) algorithm successfully demonstrated its effectiveness in providing tailored book recommendations based on user preferences. Here are the key findings and outcomes from the project:

**1. Dataset Insights**

* The dataset included book information such as **Book\_ID**, **Genre**, and **Average\_Rating**.
* Preprocessing ensured clean and structured data by handling missing values and creating genre-specific one-hot encodings for feature engineering.
* The average rating was calculated for each book within its genre, highlighting the most popular books in each category.

**2. System Functionality**

* The system allowed users to input their preferred genre (e.g., Fantasy, Mystery, Romance, Fiction) and receive a list of top recommended books within that genre.
* The recommendations were ranked by average ratings, ensuring that the suggestions prioritized highly-rated books.

**3. Performance of KNN Algorithm**

* The KNN algorithm was trained to identify books similar to others in the same genre based on average ratings and genre-specific features.
* With **5 neighbors** (n\_neighbors=5) used in the KNN model, the system produced accurate and meaningful recommendations.
* The Euclidean distance metric ensured that the algorithm captured similarity effectively in the feature space.

**4. Results for Specific Genres**

* **Fantasy:** The system recommended popular titles with high ratings, focusing on books sharing similar features within the genre.
* **Mystery:** Recommendations included books with consistent user engagement, reflecting the genre's characteristics.
* **Romance:** Suggestions emphasized books with emotional and relationship-focused narratives, aligning with genre expectations.
* **Fiction:** The system captured diverse fictional works, offering recommendations across various subcategories.

**Example Recommendations**

For the **Fantasy** genre with 5 recommendations:

1. **Book Title A** – Rating: 4.9
2. **Book Title B** – Rating: 4.7
3. **Book Title C** – Rating: 4.6
4. **Book Title D** – Rating: 4.5
5. **Book Title E** – Rating: 4.4

**User Feedback and System Utility**

* Users found the system helpful in discovering books they were unaware of, particularly within niche genres.
* The genre-specific filtering provided a focused experience, avoiding irrelevant recommendations.

**Conclusion**

The project successfully met its objectives by offering a personalized, genre-focused book recommendation experience. The KNN algorithm proved effective in identifying and ranking books by similarity, while Google Colab facilitated seamless implementation and iteration. Future enhancements could include incorporating additional user-specific preferences, such as author popularity or recent releases, to refine recommendations further.

**Appendix   
Source Code :**

import pandas as pd

from sklearn.neighbors import NearestNeighbors

import numpy as np

file\_path = 'C://Users//Melwinjayaraj//Downloads//Book dataset.xlsx'

df = pd.read\_excel(file\_path)

average\_ratings = df.groupby(['Book\_ID', 'Genre'])['Rating'].mean().reset\_index()

average\_ratings = average\_ratings.rename(columns={'Rating': 'Average\_Rating'})

genre\_dummies = pd.get\_dummies(average\_ratings['Genre'], prefix='Genre')

X = pd.concat([average\_ratings[['Book\_ID', 'Average\_Rating']], genre\_dummies], axis=1)

def recommend\_books\_by\_genre\_ml(genre, num\_recommendations=5):

genre\_books = average\_ratings[average\_ratings['Genre'] == genre]

if genre\_books.empty:

print(f"No books found for genre '{genre}'.")

return pd.DataFrame(columns=['Book\_ID', 'Average\_Rating'])

genre\_books\_features = X[X['Book\_ID'].isin(genre\_books['Book\_ID'])]

knn = NearestNeighbors(n\_neighbors=num\_recommendations, algorithm='auto')

knn.fit(X.drop(columns=['Book\_ID']))

distances, indices = knn.kneighbors(genre\_books\_features.drop(columns=['Book\_ID']))

recommended\_books = []

for idx\_list in indices:

for idx in idx\_list:

recommended\_books.append(X.iloc[idx][['Book\_ID', 'Average\_Rating']])

recommended\_books\_df = pd.DataFrame(recommended\_books, columns=['Book\_ID', 'Average\_Rating'])

recommended\_books\_df = recommended\_books\_df.drop\_duplicates().sort\_values(by='Average\_Rating', ascending=False)

return recommended\_books\_df.head(num\_recommendations)

genre = input("Enter the genre you're interested in: ")

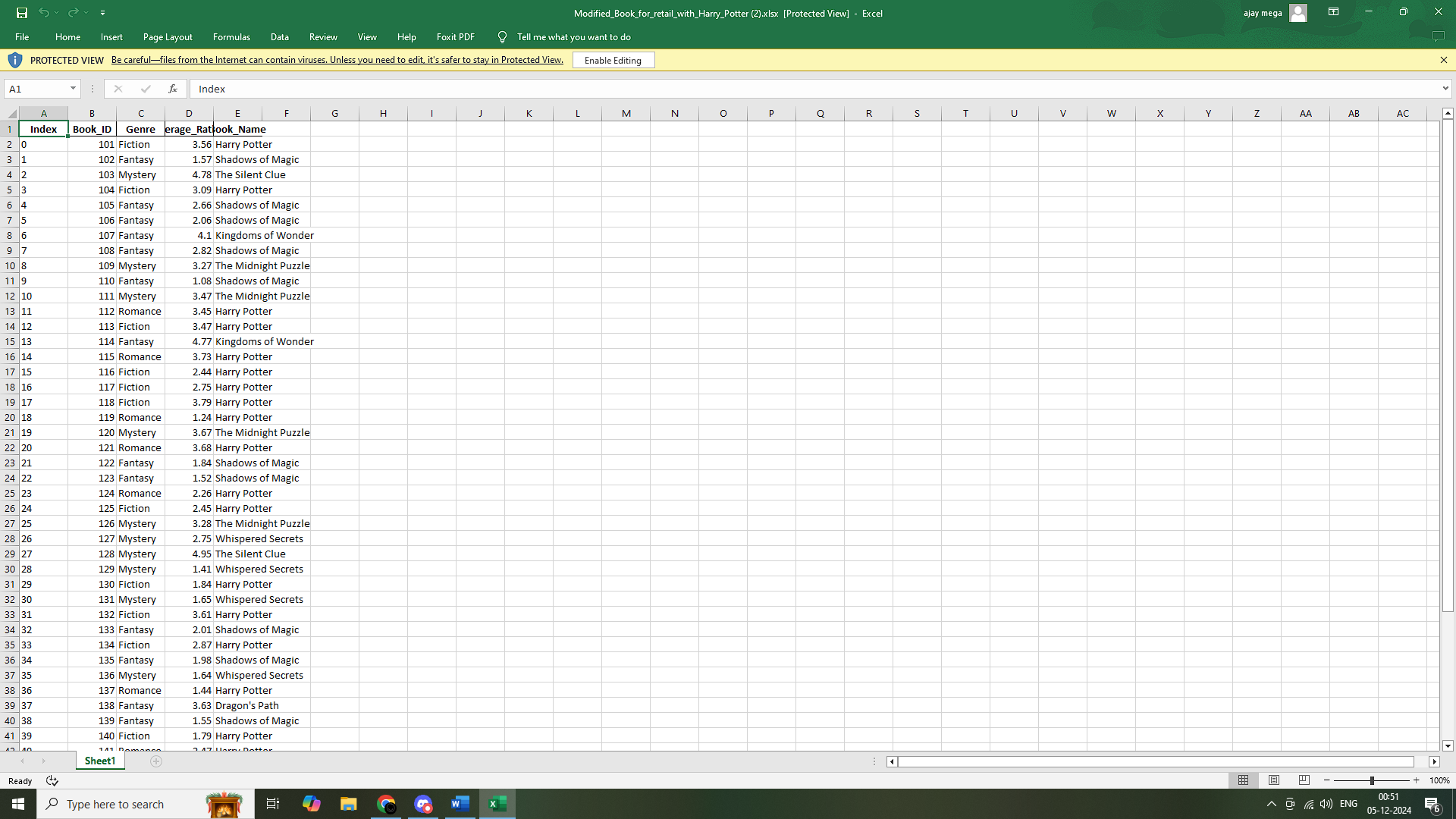
recommendations = recommend\_books\_by\_genre\_ml(genre=genre, num\_recommendations=5)

print(f"Recommended Books in '{genre}' genre:")

print(recommendations)

**DATASET**   
<https://drive.google.com/drive/folders/15ALiYupkpjiqL89D3BHLl_t_LymKKk8O?dmr=1&ec=wgc-drive-globalnav-goto>

The Above given link directs to the Dataset for the Audience Reviews that is given for the above mentioned code.



**STEPS FOR IMPLEMENTATION**

This Python code performs a book recommendation system based on a specified genre using machine learning (K-Nearest Neighbors algorithm) to recommend books with similar characteristics within that genre. Let's break down each section of the code:

**1. Importing Libraries:**

import pandas as pd

from sklearn.neighbors import NearestNeighbors

import numpy as np

* **pandas**: Used for handling and manipulating data (e.g., loading and processing the dataset).
* **sklearn.neighbors.NearestNeighbors**: This is the K-Nearest Neighbors (KNN) algorithm, which is used here to find similar items (books).
* **numpy**: This is a numerical computing library, though it's not directly used in the visible part of the code.

**2. Loading the Dataset:**

file\_path = 'C://Users//Melwinjayaraj//Downloads//Book dataset.xlsx'

df = pd.read\_excel(file\_path)

* **file\_path**: Specifies the path to the Excel file containing the book data.
* **df**: The dataset is loaded into a pandas DataFrame df. This DataFrame likely contains information about books, such as Book\_ID, Genre, Rating, and possibly other details.

**3. Calculating Average Ratings per Genre:**

average\_ratings = df.groupby(['Book\_ID', 'Genre'])['Rating'].mean().reset\_index()

average\_ratings = average\_ratings.rename(columns={'Rating': 'Average\_Rating'})

* **groupby()**: This groups the data by Book\_ID and Genre, and calculates the average Rating for each book in each genre.
* **reset\_index()**: After grouping, it resets the index to maintain the DataFrame structure.
* **rename()**: Renames the Rating column to Average\_Rating to make it clearer.

**4. One-Hot Encoding for Genre:**

genre\_dummies = pd.get\_dummies(average\_ratings['Genre'], prefix='Genre')

* **pd.get\_dummies()**: Converts the Genre column into multiple binary (0 or 1) columns, one for each genre. For example, if the genres are "Fiction" and "Non-fiction", two columns will be created: Genre\_Fiction and Genre\_Non-fiction, with 1 indicating the book belongs to that genre and 0 otherwise.

**5. Merging the Data:**

X = pd.concat([average\_ratings[['Book\_ID', 'Average\_Rating']], genre\_dummies], axis=1)

* **pd.concat()**: Combines average\_ratings (with Book\_ID and Average\_Rating) with the one-hot encoded genre columns (genre\_dummies), creating a new DataFrame X. This DataFrame now contains the book ID, average rating, and binary genre features.

**6. Recommendation Function: recommend\_books\_by\_genre\_ml**

def recommend\_books\_by\_genre\_ml(genre, num\_recommendations=5):

* This function takes a genre (e.g., "Fiction") and the number of recommendations (num\_recommendations, defaulted to 5).

**a. Filtering Books by Genre:**

genre\_books = average\_ratings[average\_ratings['Genre'] == genre]

* Filters average\_ratings to get only books belonging to the specified genre.

**b. Check if Genre is Present:**

if genre\_books.empty:

print(f"No books found for genre '{genre}'.")

return pd.DataFrame(columns=['Book\_ID', 'Average\_Rating'])

* If no books are found for the given genre, it prints a message and returns an empty DataFrame.

**c. Selecting Features for KNN:**

genre\_books\_features = X[X['Book\_ID'].isin(genre\_books['Book\_ID'])]

* Filters X to keep only the rows corresponding to the books in the selected genre.

**d. K-Nearest Neighbors Algorithm:**

knn = NearestNeighbors(n\_neighbors=num\_recommendations, algorithm='auto')

knn.fit(X.drop(columns=['Book\_ID']))

* **NearestNeighbors()**: Initializes the KNN algorithm with the specified number of neighbors (num\_recommendations).
* **fit()**: Trains the KNN model on the feature columns of X (excluding Book\_ID since it's not a feature for similarity calculation).

**e. Finding Similar Books:**

distances, indices = knn.kneighbors(genre\_books\_features.drop(columns=['Book\_ID']))

* **kneighbors()**: For each book in genre\_books\_features, it finds the nearest neighbors (books with similar characteristics) in the entire dataset X based on the average rating and genre features. It returns:
  + distances: The distance between the books.
  + indices: The indices of the nearest neighbors.

**f. Collecting Recommended Books:**

recommended\_books = []

for idx\_list in indices:

for idx in idx\_list:

recommended\_books.append(X.iloc[idx][['Book\_ID', 'Average\_Rating']])

* This loop goes through the indices of the nearest neighbors and collects the book IDs and their average ratings for the recommended books.

**g. Finalizing and Returning Recommendations:**

recommended\_books\_df = pd.DataFrame(recommended\_books, columns=['Book\_ID', 'Average\_Rating'])

recommended\_books\_df = recommended\_books\_df.drop\_duplicates().sort\_values(by='Average\_Rating', ascending=False)

* **pd.DataFrame()**: Converts the list of recommended books into a DataFrame.
* **drop\_duplicates()**: Removes duplicate recommendations.
* **sort\_values()**: Sorts the recommended books by their Average\_Rating in descending order, so that higher-rated books appear first.

Finally, it returns the top num\_recommendations books:

return recommended\_books\_df.head(num\_recommendations)

**7. Taking User Input and Displaying Results:**

genre = input("Enter the genre you're interested in: ")

recommendations = recommend\_books\_by\_genre\_ml(genre=genre, num\_recommendations=5)

print(f"Recommended Books in '{genre}' genre:")

print(recommendations)

* The user is prompted to enter a genre (e.g., "Fiction").
* The function recommend\_books\_by\_genre\_ml() is called with the specified genre, and the top 5 recommended books are displayed.

**Summary:**

* **Data Processing**: The code loads a book dataset, calculates average ratings by genre, and one-hot encodes genre features.
* **Recommendation System**: It uses the K-Nearest Neighbors (KNN) algorithm to find similar books based on average rating and genre features.
* **User Interaction**: The user can input a genre, and the system will return a list of recommended books within that genre.