Topic: Book Recommendation (Machine Learning)

Name: Shivam Vishwakarma

Roll no: 6709

Book Recommendation System

A book recommendation system is an intelligent system designed to suggest books to users based on their interests, preferences, and past behavior. Since large online platforms (like Amazon or Goodreads) contain millions of books, users often find it difficult to choose what to read next. Recommendation systems solve this problem by automatically identifying and ranking books that are most relevant to each user.

Types of Recommendation Systems

1. **Content-Based Filtering**
    - Uses book features such as title, author, genre, or description.
    - Example: If a user likes "The Hobbit", recommend other fantasy books by J.R.R. Tolkien or similar authors.
2. **Collaborative Filtering**
    - Uses user behavior (ratings, interactions) instead of book features.
    - Two main types:
        - **User-based CF:** Finds users with similar tastes and recommends books they liked.
        - **Item-based CF:** Finds books that were rated similarly by many users (used in your model).
3. **Hybrid Systems**
    - Combine both content-based and collaborative filtering to overcome limitations of each.
    - Example: Recommend books similar in genre and also highly rated by similar users.

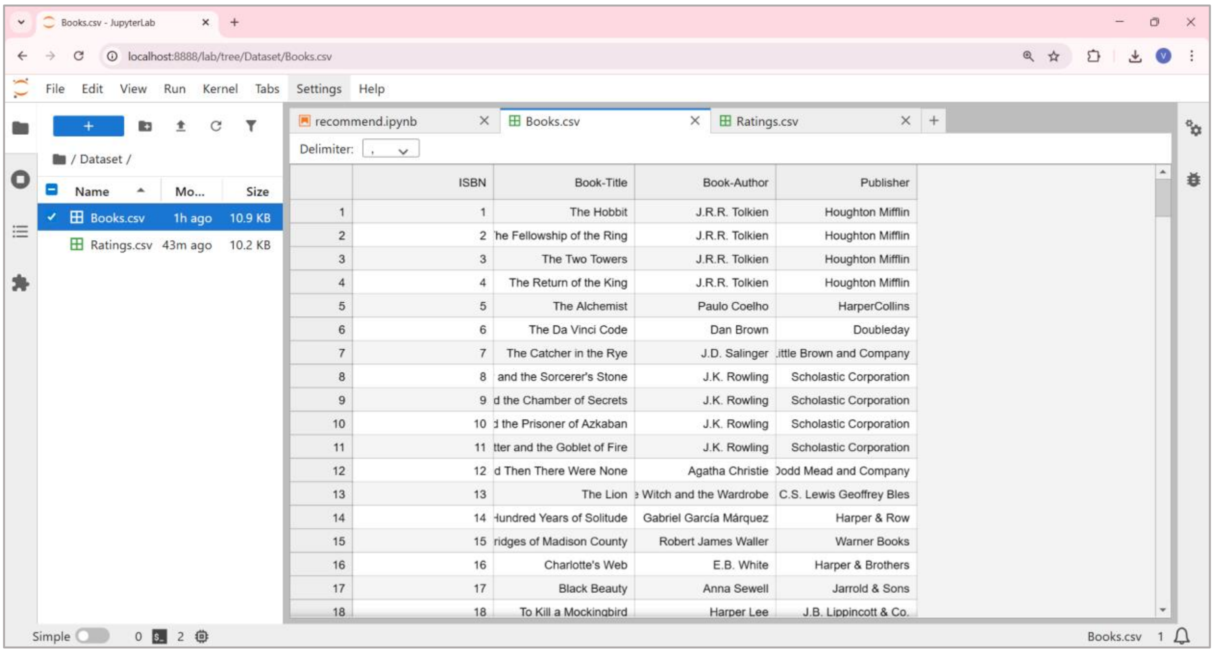
How our model works

In this project, I used an Item-Based Collaborative Filtering approach with K-Nearest Neighbors (KNN):

- I built a User–Item Matrix where rows = users and columns = books (ISBN).
- Each book is represented as a vector of user ratings.
- Using cosine similarity, we measure how similar two books are based on user ratings.
- For a given input book, the model finds the k most similar books and recommends them.

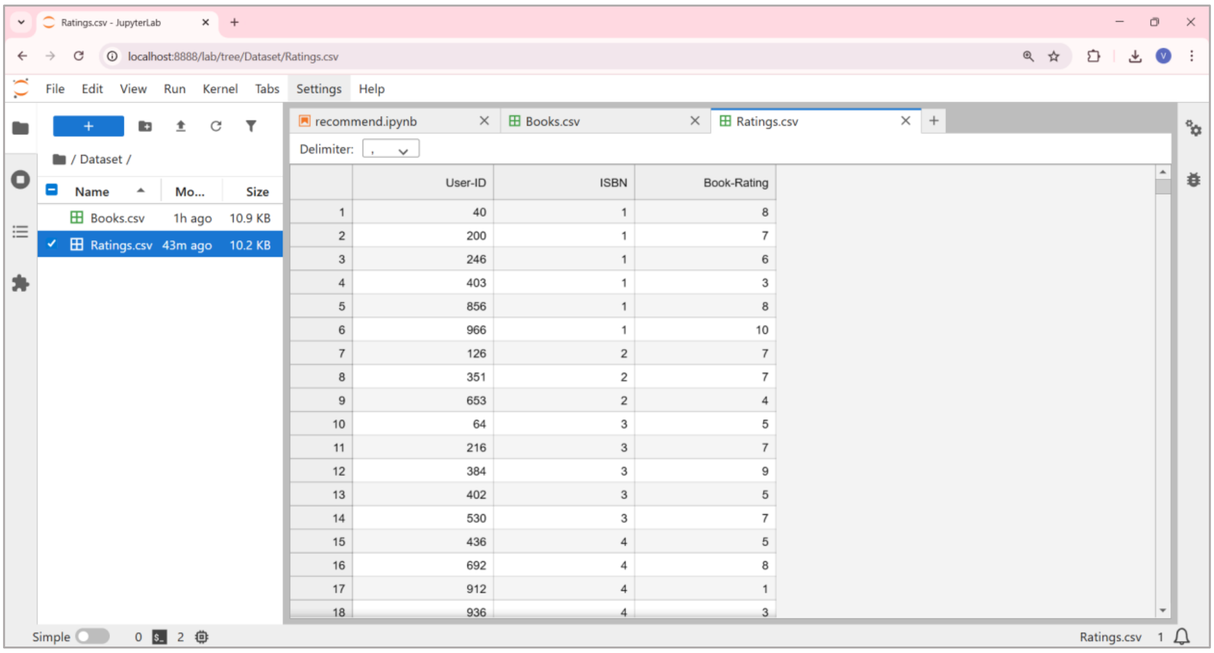
Data Set

Books.csv



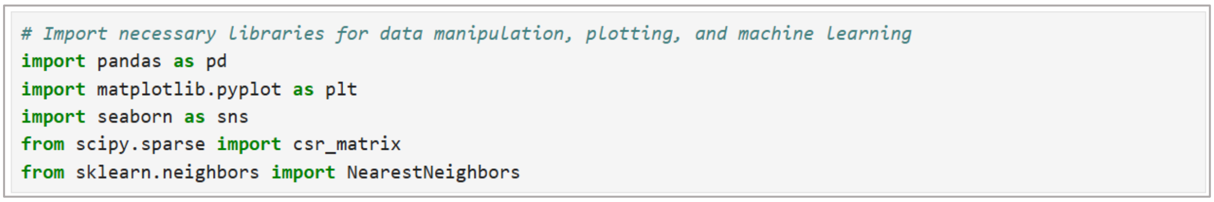
	ISBN	Book-Title	Book-Author	Publisher
1	1	The Hobbit	J.R.R. Tolkien	Houghton Mifflin
2	2	'he Fellowship of the Ring	J.R.R. Tolkien	Houghton Mifflin
3	3	The Two Towers	J.R.R. Tolkien	Houghton Mifflin
4	4	The Return of the King	J.R.R. Tolkien	Houghton Mifflin
5	5	The Alchemist	Paulo Coelho	HarperCollins
6	6	The Da Vinci Code	Dan Brown	Doubleday
7	7	The Catcher in the Rye	J.D. Salinger	.ittle Brown and Company
8	8	and the Sorcerer's Stone	J.K. Rowling	Scholastic Corporation
9	9	d the Chamber of Secrets	J.K. Rowling	Scholastic Corporation
10	10	d the Prisoner of Azkaban	J.K. Rowling	Scholastic Corporation
11	11	tter and the Goblet of Fire	J.K. Rowling	Scholastic Corporation
12	12	d Then There Were None	Agatha Christie	Dodd Mead and Company
13	13	The Lion	e Witch and the Wardrobe	C.S. Lewis Geoffrey Bles
14	14	Hundred Years of Solitude	Gabriel García Márquez	Harper & Row
15	15	ridges of Madison County	Robert James Waller	Warner Books
16	16	Charlotte's Web	E.B. White	Harper & Brothers
17	17	Black Beauty	Anna Sewell	Jarrold & Sons
18	18	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.

Ratings.csv



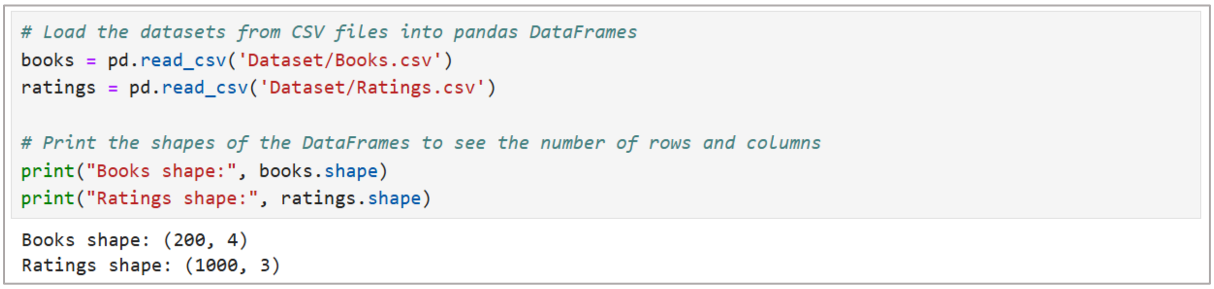
	User-ID	ISBN	Book-Rating
1	40	1	8
2	200	1	7
3	246	1	6
4	403	1	3
5	856	1	8
6	966	1	10
7	126	2	7
8	351	2	7
9	653	2	4
10	64	3	5
11	216	3	7
12	384	3	9
13	402	3	5
14	530	3	7
15	436	4	5
16	692	4	8
17	912	4	1
18	936	4	3

Step 1: Importing required libraries



```
# Import necessary libraries for data manipulation, plotting, and machine learning
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
```

Step 2: Loading the Dataset and checking the dimension of data

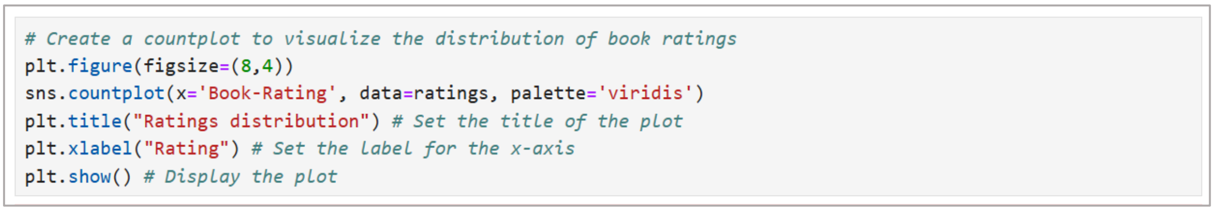


```
# Load the datasets from CSV files into pandas DataFrames
books = pd.read_csv('Dataset/Books.csv')
ratings = pd.read_csv('Dataset/Ratings.csv')

# Print the shapes of the DataFrames to see the number of rows and columns
print("Books shape:", books.shape)
print("Ratings shape:", ratings.shape)
```

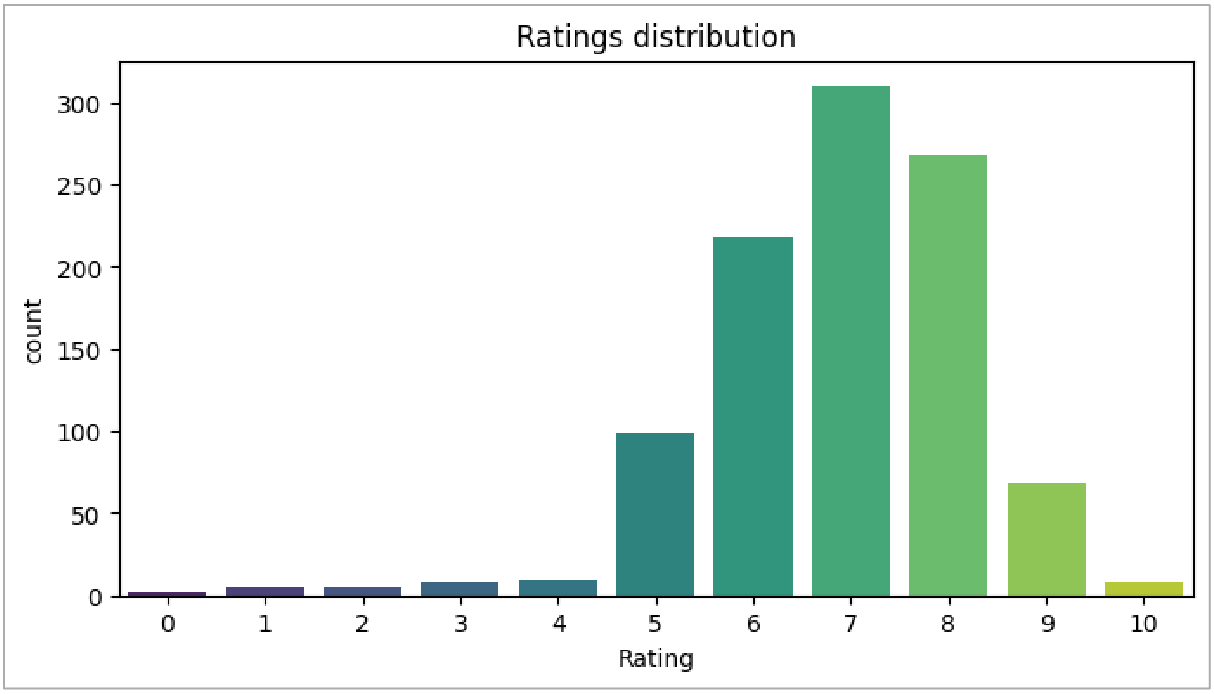
```
Books shape: (200, 4)
Ratings shape: (1000, 3)
```

Step 3: Analysing the Data

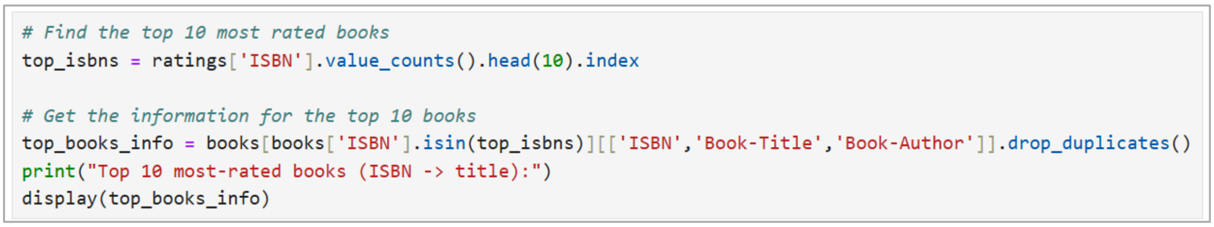
1. Ratings distribution of Books: Majorly books have been rated 6, 7 and 8



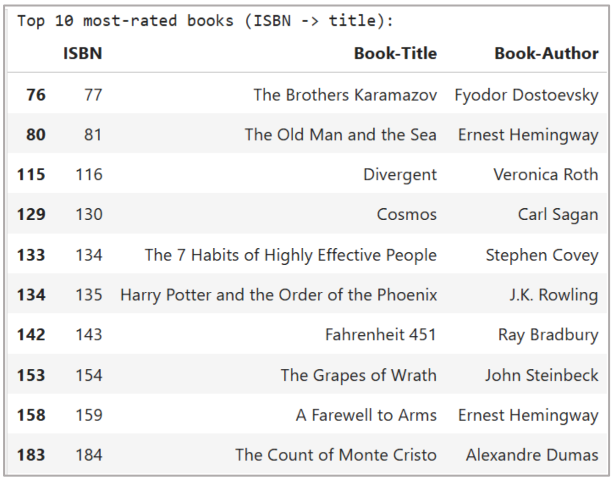
```
# Create a countplot to visualize the distribution of book ratings
plt.figure(figsize=(8,4))
sns.countplot(x='Book-Rating', data=ratings, palette='viridis')
plt.title("Ratings distribution") # Set the title of the plot
plt.xlabel("Rating") # Set the label for the x-axis
plt.show() # Display the plot
```



2. Top 10 most rated books



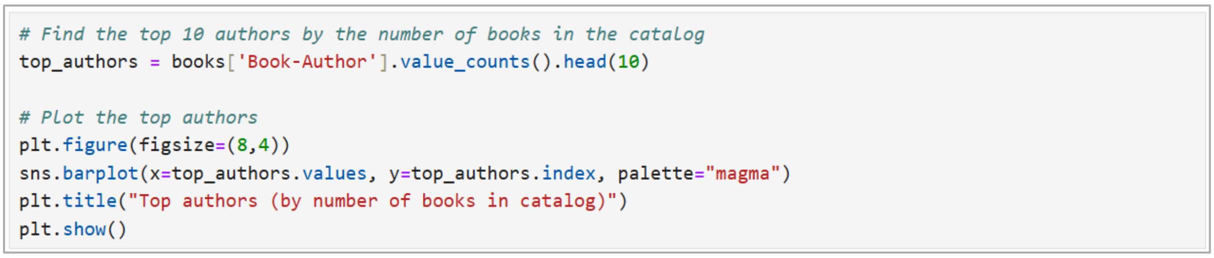
```
# Find the top 10 most rated books
top_isbns = ratings['ISBN'].value_counts().head(10).index

# Get the information for the top 10 books
top_books_info = books[books['ISBN'].isin(top_isbns)][['ISBN','Book-Title','Book-Author']].drop_duplicates()
print("Top 10 most-rated books (ISBN -> title):")
display(top_books_info)
```



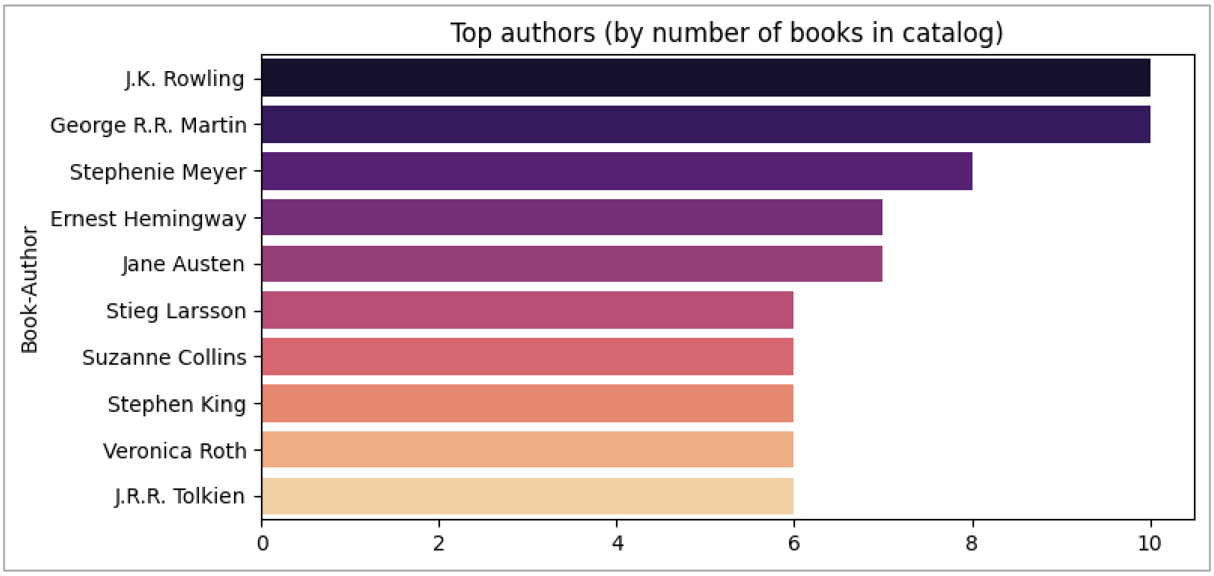
```
Top 10 most-rated books (ISBN -> title):
```

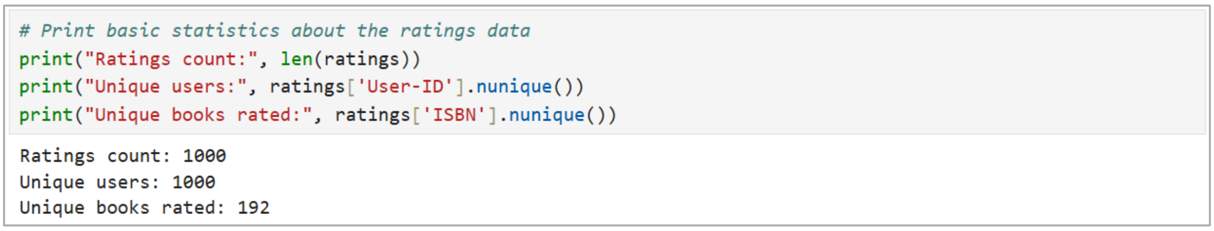
	ISBN	Book-Title	Book-Author
76	77	The Brothers Karamazov	Fyodor Dostoevsky
80	81	The Old Man and the Sea	Ernest Hemingway
115	116	Divergent	Veronica Roth
129	130	Cosmos	Carl Sagan
133	134	The 7 Habits of Highly Effective People	Stephen Covey
134	135	Harry Potter and the Order of the Phoenix	J.K. Rowling
142	143	Fahrenheit 451	Ray Bradbury
153	154	The Grapes of Wrath	John Steinbeck
158	159	A Farewell to Arms	Ernest Hemingway
183	184	The Count of Monte Cristo	Alexandre Dumas

3. Most Books by Author



```
# Find the top 10 authors by the number of books in the catalog
top_authors = books['Book-Author'].value_counts().head(10)

# Plot the top authors
plt.figure(figsize=(8,4))
sns.barplot(x=top_authors.values, y=top_authors.index, palette="magma")
plt.title("Top authors (by number of books in catalog)")
plt.show()
```

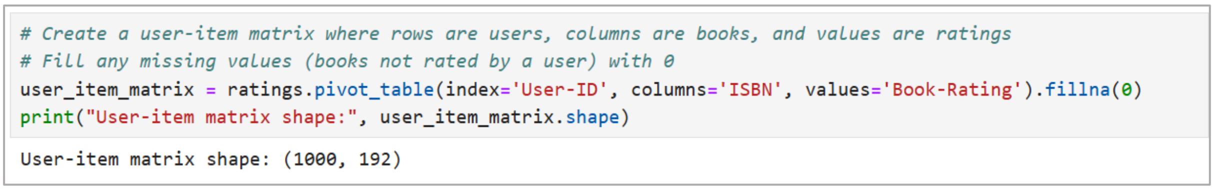




```
# Print basic statistics about the ratings data
print("Ratings count:", len(ratings))
print("Unique users:", ratings['User-ID'].nunique())
print("Unique books rated:", ratings['ISBN'].nunique())
```

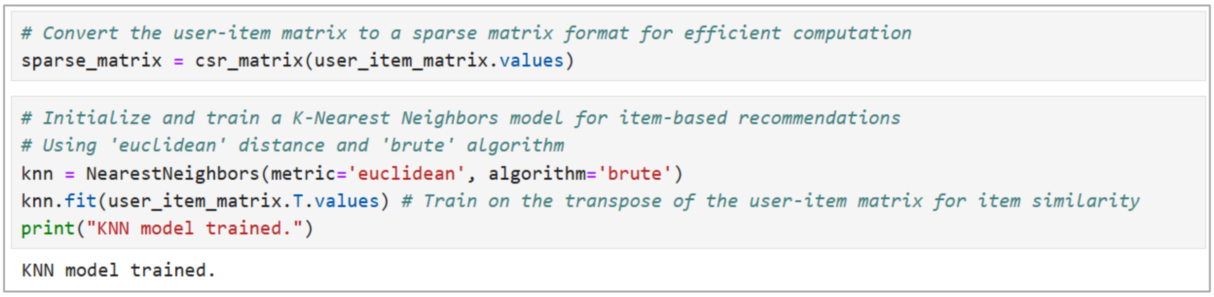
```
Ratings count: 1000
Unique users: 1000
Unique books rated: 192
```

Step 4: Building Model & training the model



```
# Create a user-item matrix where rows are users, columns are books, and values are ratings
# Fill any missing values (books not rated by a user) with 0
user_item_matrix = ratings.pivot_table(index='User-ID', columns='ISBN', values='Book-Rating').fillna(0)
print("User-item matrix shape:", user_item_matrix.shape)
```

```
User-item matrix shape: (1000, 192)
```

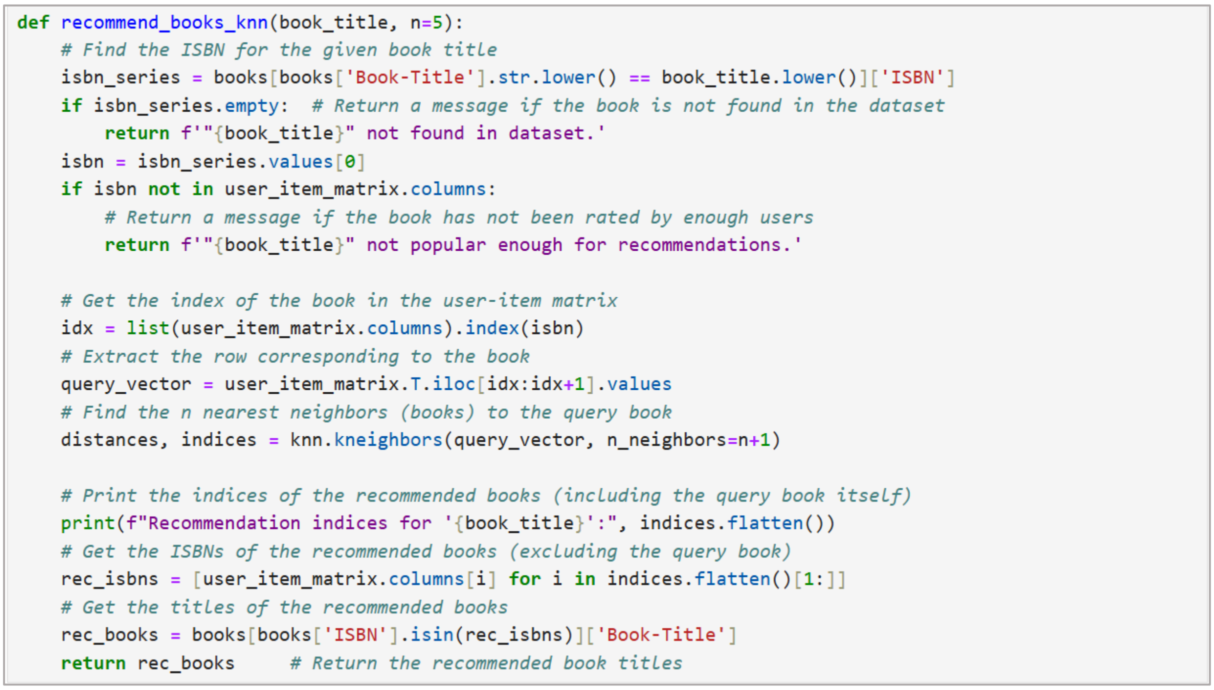


```
# Convert the user-item matrix to a sparse matrix format for efficient computation
sparse_matrix = csr_matrix(user_item_matrix.values)
```

```
# Initialize and train a K-Nearest Neighbors model for item-based recommendations
# Using 'euclidean' distance and 'brute' algorithm
knn = NearestNeighbors(metric='euclidean', algorithm='brute')
knn.fit(user_item_matrix.T.values) # Train on the transpose of the user-item matrix for item similarity
print("KNN model trained.")
```

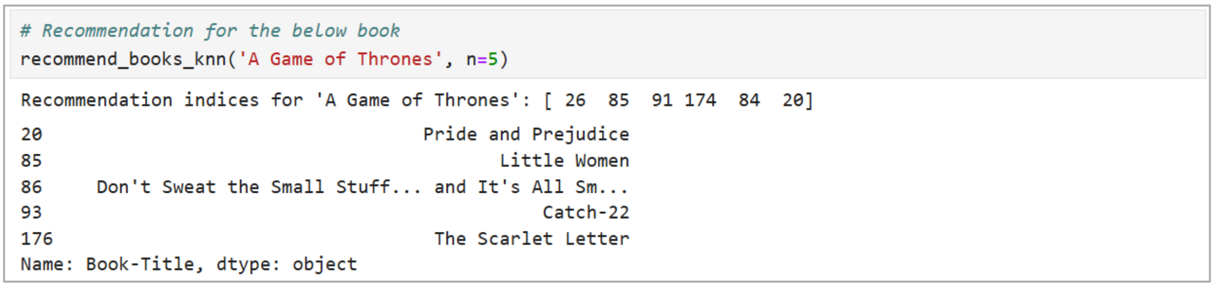
```
KNN model trained.
```

Step 5: Defining function to get recommended book.



```
def recommend_books_knn(book_title, n=5):
    # Find the ISBN for the given book title
    isbn_series = books[books['Book-Title'].str.lower() == book_title.lower()]['ISBN']
    if isbn_series.empty:  # Return a message if the book is not found in the dataset
        return f'"{book_title}" not found in dataset.'
    isbn = isbn_series.values[0]
    if isbn not in user_item_matrix.columns:
        # Return a message if the book has not been rated by enough users
        return f'"{book_title}" not popular enough for recommendations.'

    # Get the index of the book in the user-item matrix
    idx = list(user_item_matrix.columns).index(isbn)
    # Extract the row corresponding to the book
    query_vector = user_item_matrix.T.iloc[idx:idx+1].values
    # Find the n nearest neighbors (books) to the query book
    distances, indices = knn.kneighbors(query_vector, n_neighbors=n+1)

    # Print the indices of the recommended books (including the query book itself)
    print(f"Recommendation indices for '{book_title}':", indices.flatten())
    # Get the ISBNs of the recommended books (excluding the query book)
    rec_isbns = [user_item_matrix.columns[i] for i in indices.flatten()[1:]]
    # Get the titles of the recommended books
    rec_books = books[books['ISBN'].isin(rec_isbns)]['Book-Title']
    return rec_books     # Return the recommended book titles
```

Step 6: Checking 5 recommendation for book “A Game of Thrones”



```
# Recommendation for the below book
recommend_books_knn('A Game of Thrones', n=5)
```

```
Recommendation indices for 'A Game of Thrones': [ 26  85  91 174  84  20]
20                                    Pride and Prejudice
85                                           Little Women
86     Don't Sweat the Small Stuff... and It's All Sm...
93                                               Catch-22
176                                    The Scarlet Letter
Name: Book-Title, dtype: object
```