SINDH MADRESSATUL ISLAM UNIVERSITY

ENTRE TO LEARN
GO FORTH TO SERVE

1885

Machine Learning

Project Topics

Name: Danish Ahmed
ID: CSC-21F-032
Section: 8A
Instructor: Miss Aqsa Umer

Assignment part 2

1. Dataset Overview

You have 3 datasets:

- **Books** (271k rows): Contains book details like ISBN, title, author, year, publisher.
- **Users** (278k rows): User data with locations and ages.
- **Ratings** (1.1M rows): User-book interactions with ratings (0–10).

Dataset Link: Kaggle - Book Recommendation Dataset

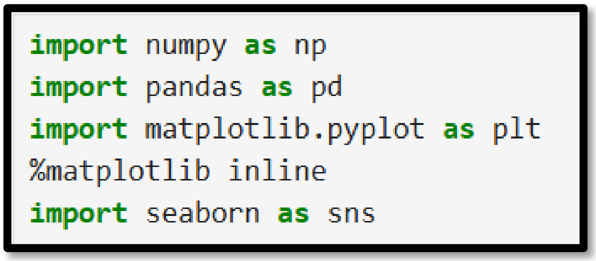
I downloaded and extracted these files into my project folder to start working.

2. Preprocessing Steps

Before building any machine learning model, it is important to clean and prepare the data properly.
Here are the detailed preprocessing steps I performed on the dataset:

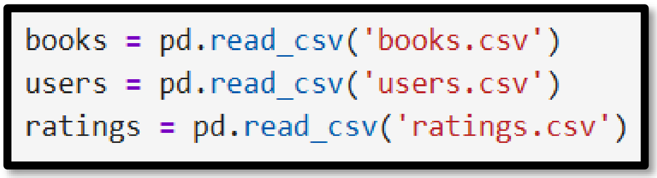
2.1 Loading the Data

First, I imported the necessary libraries:



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Then I loaded the three CSV files:



```
books = pd.read_csv('books.csv')
users = pd.read_csv('users.csv')
ratings = pd.read_csv('ratings.csv')
```

2.2 Understanding the Data

To get a basic understanding, I displayed the first few rows of each dataset using .head():

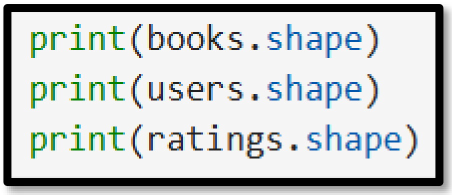
Books Dataset (books.head())

Users Dataset (users.head())

Ratings Dataset (ratings.head())

2.3 Checking Shape of the Data

I checked the number of rows and columns in each dataset:



```
print(books.shape)
print(users.shape)
print(ratings.shape)
```

```
(271360, 8)
(278858, 3)
(1149780, 3)
```

2.4 Checking for Missing Values

Using **.isnull().sum()**, I checked missing values:

1. Books Missing Values:

Column	Missing Values
Book-Author	2
Publisher	2
Image-URL-L	3

2. Users Missing Values:

Column	Missing Values
Age	110,762

3. Ratings Missing Values:

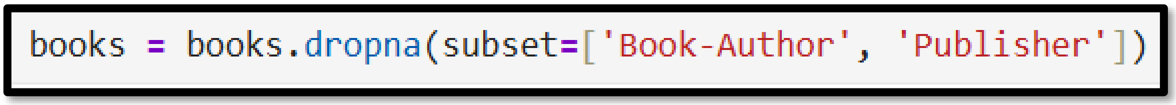
Column	Missing Values
None	0

No missing values in the **Ratings** dataset.

2.5 Handling Missing Values

Books Dataset:

Since Book Author and Publisher are important for our system, I dropped rows where these fields were missing:



```
books = books.dropna(subset=['Book-Author', 'Publisher'])
```

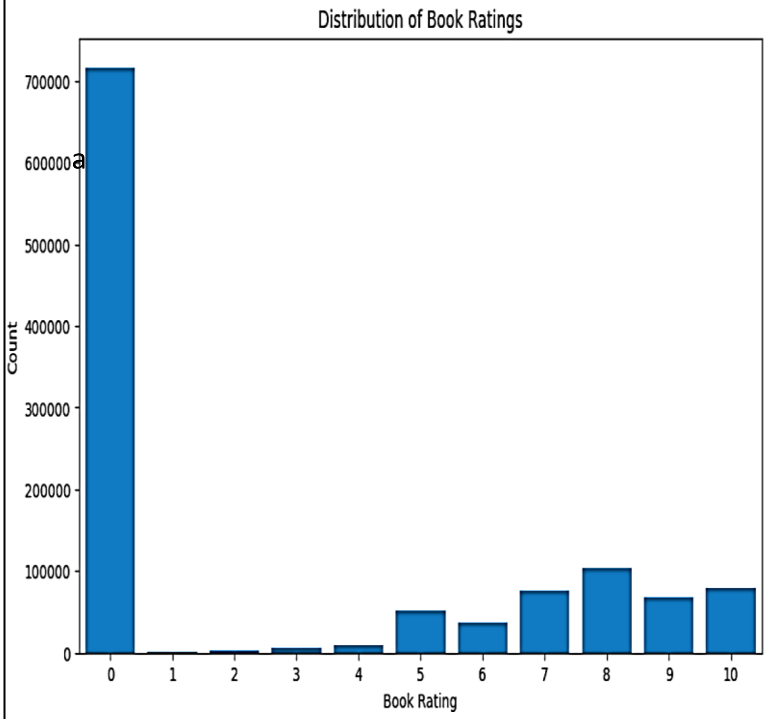
Corrected Preprocessing for Users Dataset (Age Column)

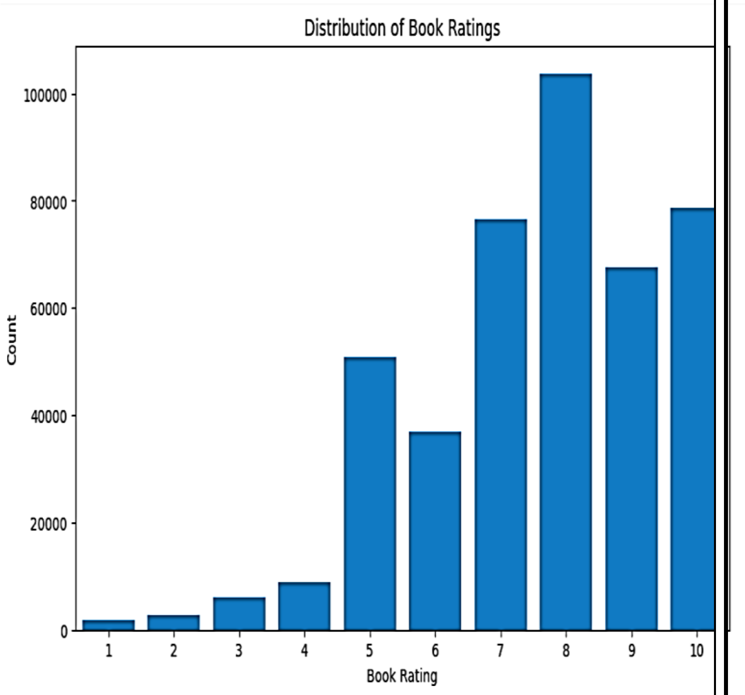
Step 1: Filtering Ratings

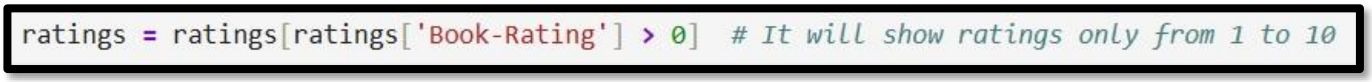
First, I filtered out the ratings to only include users who have given ratings greater than 0: such as (1-10), because the graph was looking bad if starting with 0 ratings.

BEFORE:

AFTER:



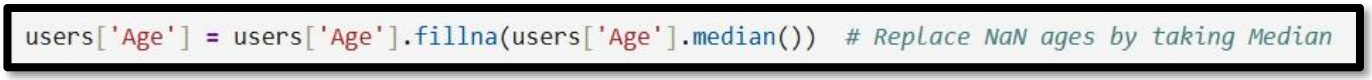




```
ratings = ratings[ratings['Book-Rating'] > 0]   # It will show ratings only from 1 to 10
```

Step 2: Filling Missing Age Values

Instead of leaving the missing ages, I replaced the missing (NaN) values by using the **median** of the Age column:

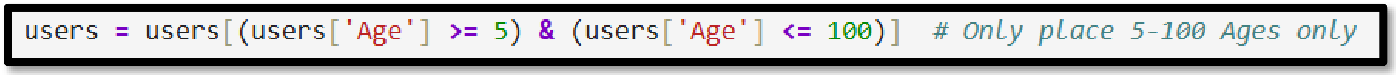


```
users['Age'] = users['Age'].fillna(users['Age'].median())  # Replace NaN ages by taking Median
```

Reason:
In this case, Median is better than mean because it is not affected by outliers (e.g., some users who might have mistakenly entered 200 years old).

Step 3: Filtering Realistic Ages

Some users had entered unrealistic ages (like less than 5 years or more than 100 years).

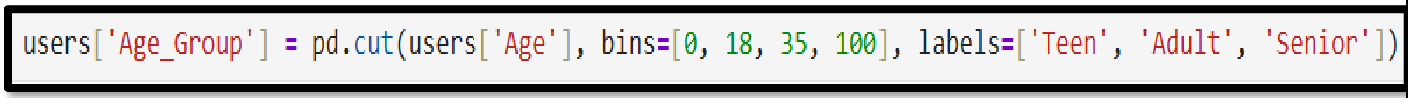


```
users = users[(users['Age'] >= 5) & (users['Age'] <= 100)]  # Only place 5-100 Ages only
```

Step 4: Creating Age Groups

I then created **age groups** to categorize users into:

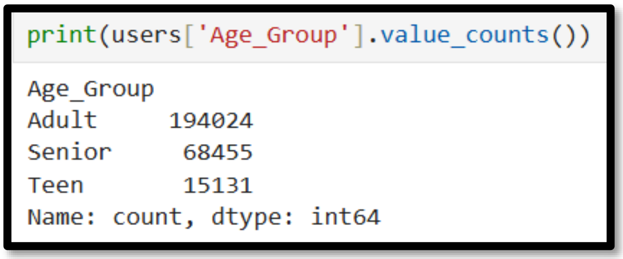
- Teen (0-18 years)
- Adult (19-35 years)
- Senior (36-100 years)



```
users['Age_Group'] = pd.cut(users['Age'], bins=[0, 18, 35, 100], labels=['Teen', 'Adult', 'Senior'])
```

This step will help later if I want to recommend books based on the user's **age group**.

Step 5: Checking How Many in Each Age Group



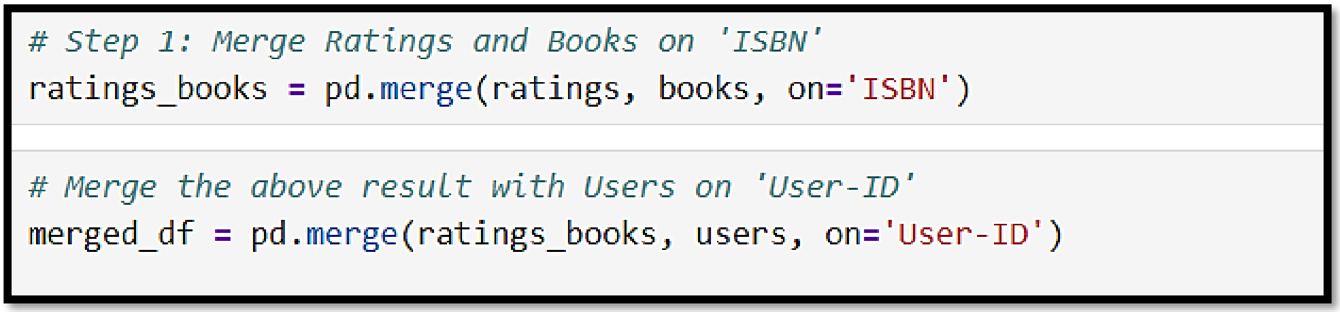
```
print(users['Age_Group'].value_counts())

Age_Group
Adult     194024
Senior     68455
Teen       15131
Name: count, dtype: int64
```

2.6 Checking Duplicates:

No duplicate rows found— data is clear

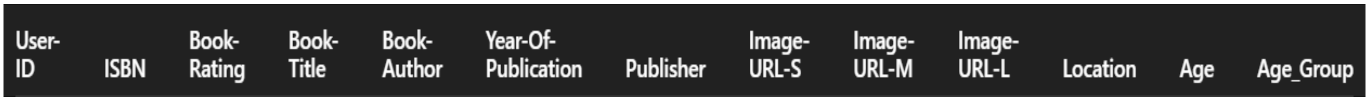
2.7 Merging



```
# Step 1: Merge Ratings and Books on 'ISBN'
ratings_books = pd.merge(ratings, books, on='ISBN')

# Merge the above result with Users on 'User-ID'
merged_df = pd.merge(ratings_books, users, on='User-ID')
```

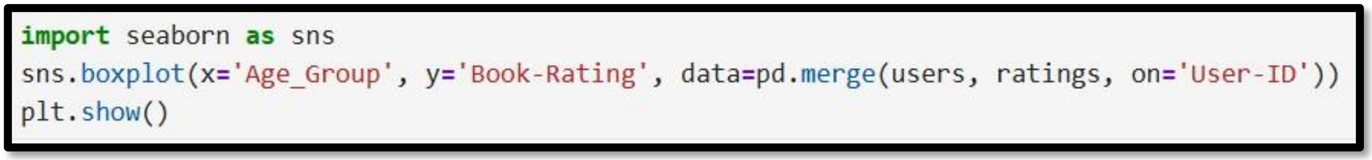
By writing merged_df the merged data will be shown in Tabular form



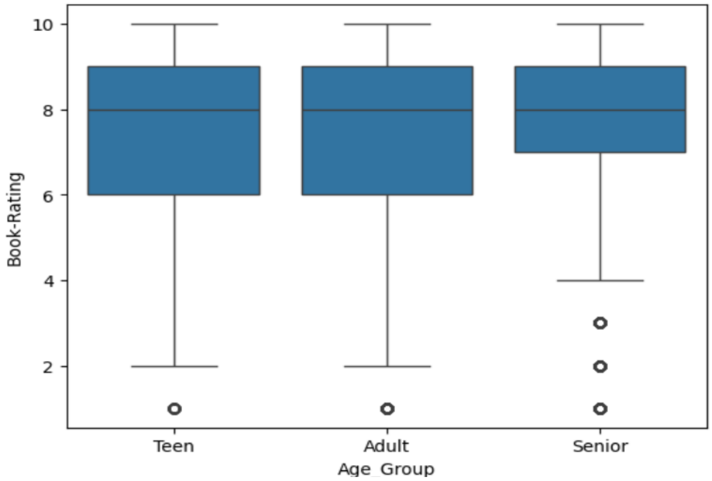
User-ID	ISBN	Book-Rating	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	Image-URL-M	Image-URL-L	Location	Age	Age_Group

2.8 How Different Age Groups Rate Books

Then in the Last I showed the ratings given by different age groups by grapgh.



```
import seaborn as sns
sns.boxplot(x='Age_Group', y='Book-Rating', data=pd.merge(users, ratings, on='User-ID'))
plt.show()
```



3. Research Paper Summary

Title: A Survey of Collaborative Filtering Techniques
Link: Research Paper PDF

Overview: This paper explains different collaborative filtering methods used in recommendation systems. It covers:

- Memory-based techniques (like user-user and item-item collaborative filtering)
- Model-based techniques (like matrix factorization, SVD, etc.)
- Challenges like scalability, sparsity, and cold-start problems.

Importance: Understanding collaborative filtering methods is crucial for building an effective book recommendation system. It helped me plan how I will build the model (Collaborative Filtering approach).