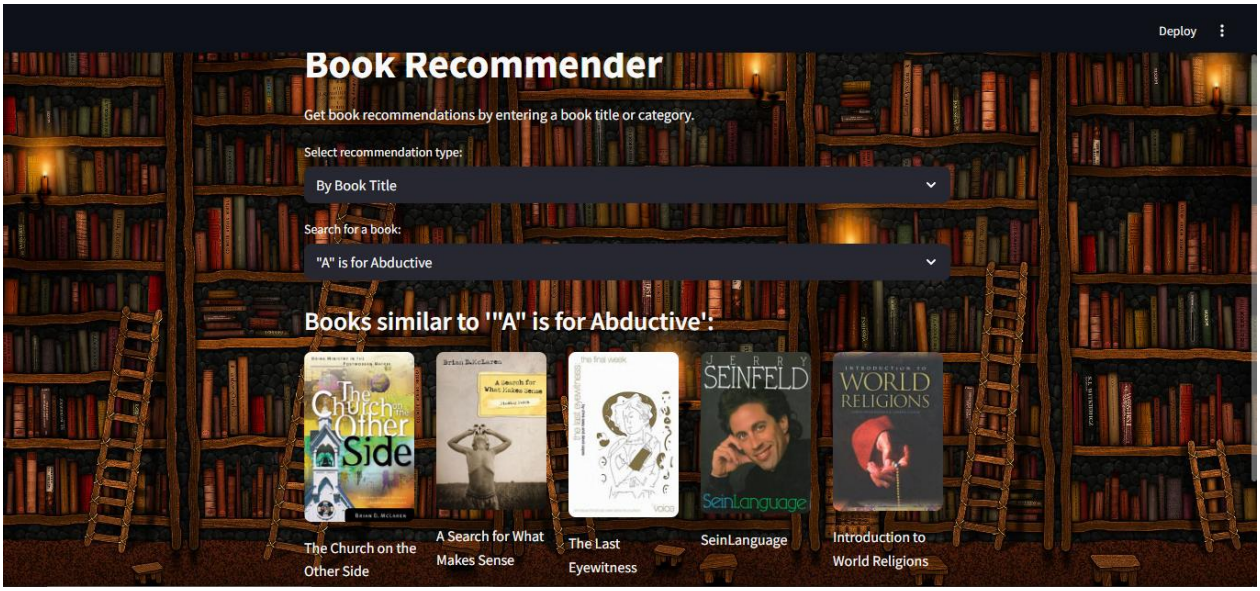
Book Recommendation system (Category or Title)



Team: 23

- احمد محمد احمد فؤاد المنيلاوي
- احمد سعيد محمد امام زايد
- احمد رضا كامل عبد الغني
- احمد محمد علي محمد
- محمد حازم السيد

إشراف: أ.د/ مدحت حسين احمد

Faculty of Engineering

Helwan National University

2025

Introduction:

In today's digital age, the overwhelming volume of available literature can make it challenging for readers to find books that match their interests efficiently. A Book Recommendation System aims to simplify this process by providing personalized suggestions, helping users discover new books aligned with their preferences. This report presents the development of a Book Recommendation System implemented in Python, integrated with a user-friendly web interface.

The system leverages powerful Python libraries for data processing and machine learning to analyze book metadata such as titles and categories. Users interact with the system via a web-based interface where they can input a book title or select a category. Based on the input, the system generates tailored recommendations by finding similar titles or books within the chosen genre, enhancing user experience and engagement.

This approach not only facilitates quicker book discovery but also supports informed reading choices by connecting users to books that match their tastes. The integration of Python's backend capabilities with a responsive webpage interface demonstrates a practical and scalable solution suitable for modern digital libraries and online bookstores.

Data Preprocessing

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pickle
```

[1] ✓ 10.7s

LOAD DATA

```
books = pd.read_csv ('Books.csv')
```

[2] ✓ 0.1s

```python
# Clean column names and whitespace
books.columns = books.columns.str.strip()

# Remove spaces in object columns
for col in books.select_dtypes(include=["object"]).columns:
    books[col] = books[col].replace(r'\s+', ' ', regex=True)
```

[3] ✓ 0.4s

```python
# Fix column names according to your dataset
books.rename(columns={'published': 'published_year', 'average_r': 'average_rating', 'num_page': 'num_pages'}, inplace=True)
```

[4] ✓ 0.0s

```python
# Handle Missing Data
print("Missing values before handling:")
books.isnull().sum()
```

Chat (CTRL + I) / Share (CTRL + L)

[5] ✓ 0.0s

```
Missing values before handling:

isbn13               0
isbn10               0
title                0
subtitle          4429
authors             72
categories          99
thumbnail          329
description        262
published_year       6
```

```python
books.shape
```

[6] ✓ 0.0s Python

```
(6810, 12)
```

```python
columns with too many missing values or low importance
 books.drop(columns=['subtitle', 'description'])

e missing values with a placeholder
authors'] = books['authors'].fillna('Unknown')
thumbnail'] = books['thumbnail'].fillna('https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSOAp5VXoGKS16VrxT0PuDgPsktnBnT614axA&s')
categories'] = books['categories'].fillna('Unknown')

e missing values for 'average_rating' with median
average_rating'] = books['average_rating'].fillna(books['average_rating'].median())

e missing values for 'ratings_count' with 0
ratings_count'] = books['ratings_count'].fillna(0)

a missing values with median
ropna(subset=['num_pages' , 'published_year'], inplace=True)
eset_index(drop=True, inplace=True)

After Handling:")
ooks.isnull().sum())
```

[7] ✓ 0.0s Python

```
After Handling:
isbn13            0
isbn10            0
title             0
authors           0
categories        0
thumbnail         0
published_year    0
average_rating    0
num_pages         0
ratings_count     0
dtype: int64
```

```python
books.shape
```

[8] ✓ 0.0s

```
(6762, 10)
```

```python
# Select essential columns
books = books[['title', 'authors', 'categories', 'average_rating', 'ratings_count', 'num_pages', 'published_year', 'thumbnail']]
```

[9] ✓ 0.0s

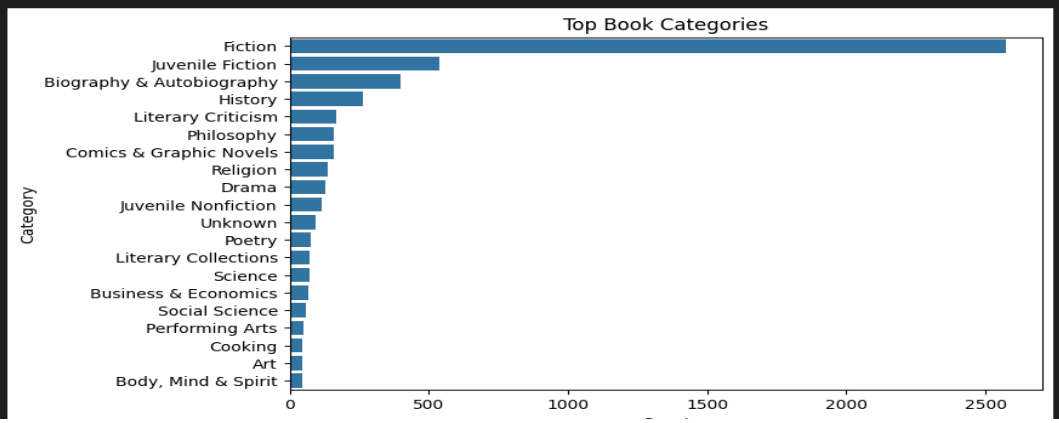
```python
# === 4. FEATURE ENGINEERING ===
books['tags'] = books['title'] + ' ' + books['authors'] + ' ' + books['categories']
books['tags'] = books['tags'].str.lower()
```

[10] ✓ 0.0s

Data Visualization

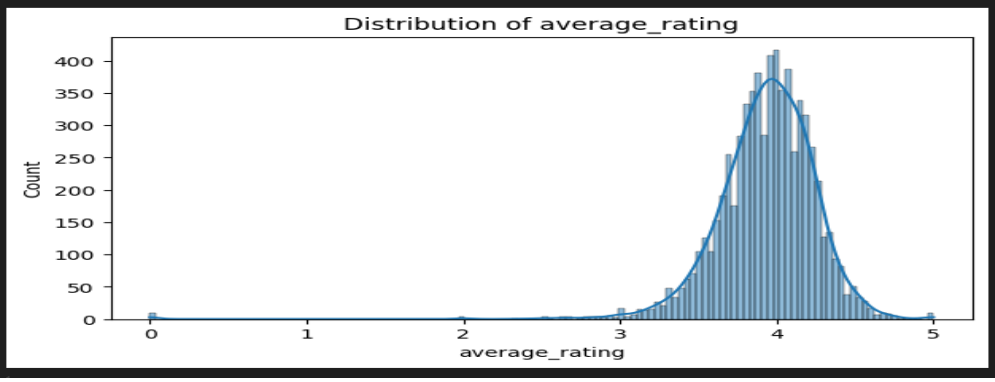
```
sns.barplot(x=top_categories.values, y=top_categories.index)
plt.title("Top Book Categories")
plt.xlabel("Count")
plt.ylabel("Category")
plt.show()
```

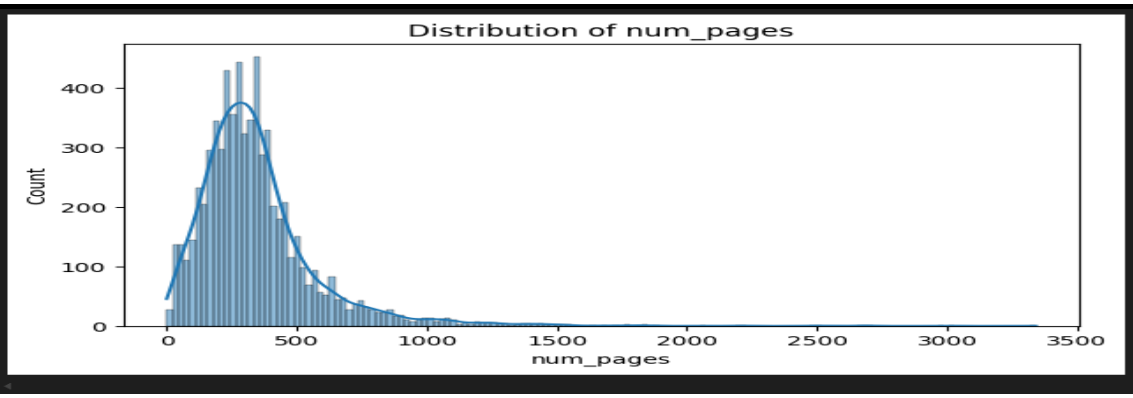
[11] ✓ 0.6s

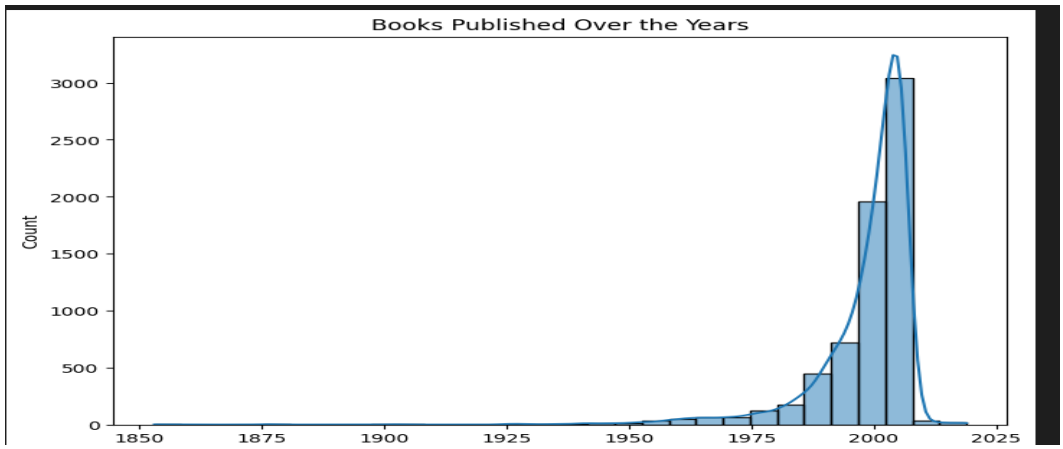


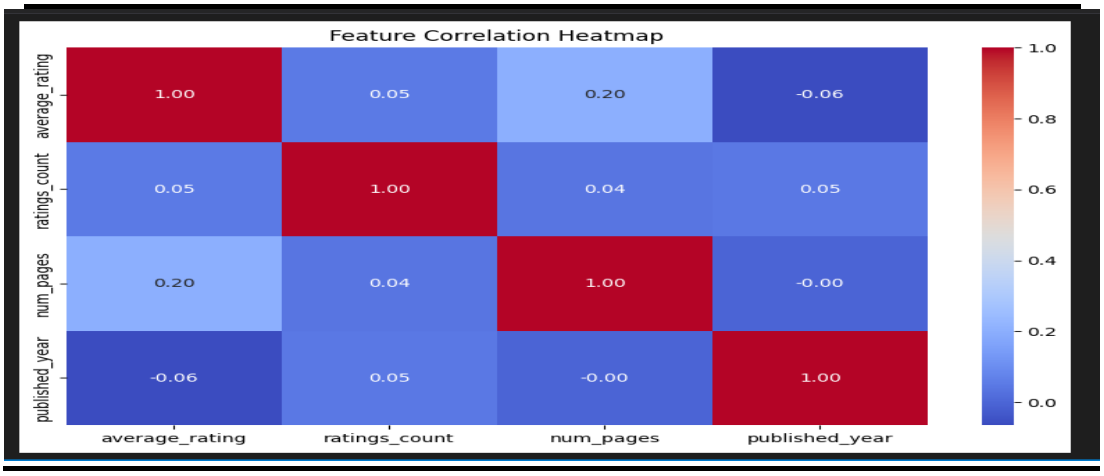

```
# Numeric distributions
for col in ['average_rating', 'num_pages']:
    plt.figure(figsize=(7, 4))
    sns.histplot(books[col], kde=True)
    plt.title(f"Distribution of {col}")
    plt.show()
```

[12] ✓ 1.2s









Recommendation

Recommendation by Book Title using similarity matrix

```python
def recommend(book_title):
    try:
        # Find the index of the book in the DataFrame
        index = books[books['title'].str.lower() == book_title.lower()].index[0]
    except IndexError:
        print(f"Book titled '{book_title}' not found.")
        return pd.DataFrame()

    # Get similarity scores for the book with all others
    distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])

    # Collect top 5 recommendations excluding the book itself
    recommended_indices = [i[0] for i in distances[1:6]]
    recommended_books = books.iloc[recommended_indices]

    return recommended_books
```

✓ 0.0s

Recommendation by Category

```python
def recommend_by_category(category_name, top_n=5):
    # Filter books containing the category (case-insensitive)
    category_books = books[books['categories'].str.contains(category_name, case=False, na=False)].copy()

    if category_books.empty:
        print(f"No books found in category '{category_name}'.")
        return pd.DataFrame()

    # Create a score combining average rating and number of ratings (popularity * quality)
    category_books['score'] = category_books['average_rating'] * category_books['ratings_count']

    # Sort by score descending and return top_n
    top_books = category_books.sort_values(by='score', ascending=False).head(top_n)

    return top_books
```

[17] ✓ 0.0s

```python
        display(recommended[['title', 'authors', 'categories', 'average_rating']])
    else:
        print("No recommendations found.")

    # Example: Recommend top books in a category
    print("\nTop books in category 'Fantasy':")
    top_category_books = recommend_by_category("Fantasy", top_n=5)
    if not top_category_books.empty:
        display(top_category_books[['title', 'authors', 'categories', 'average_rating', 'ratings_count']])
    else:
        print("No books found in this category.")
```

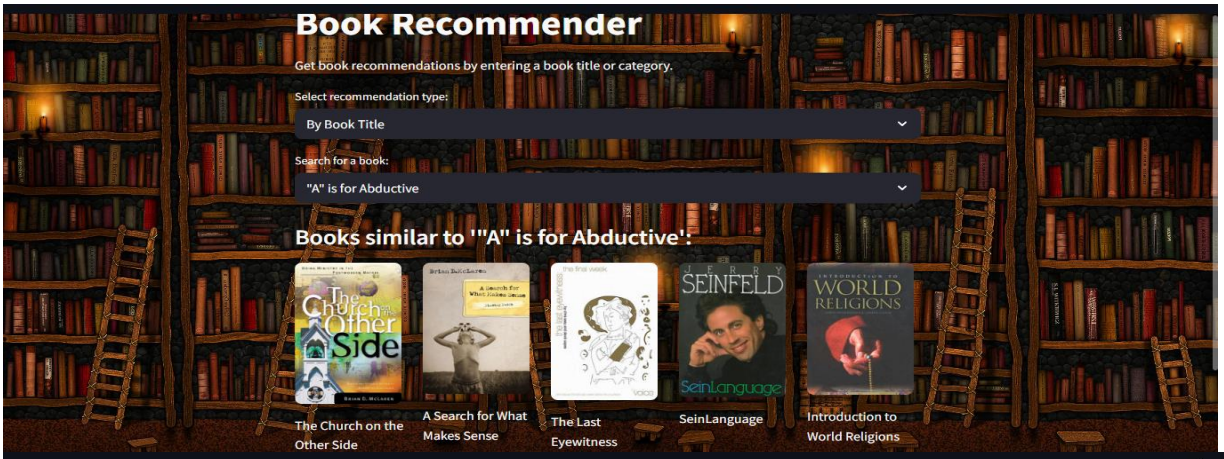
[18] ✓ 0.0s

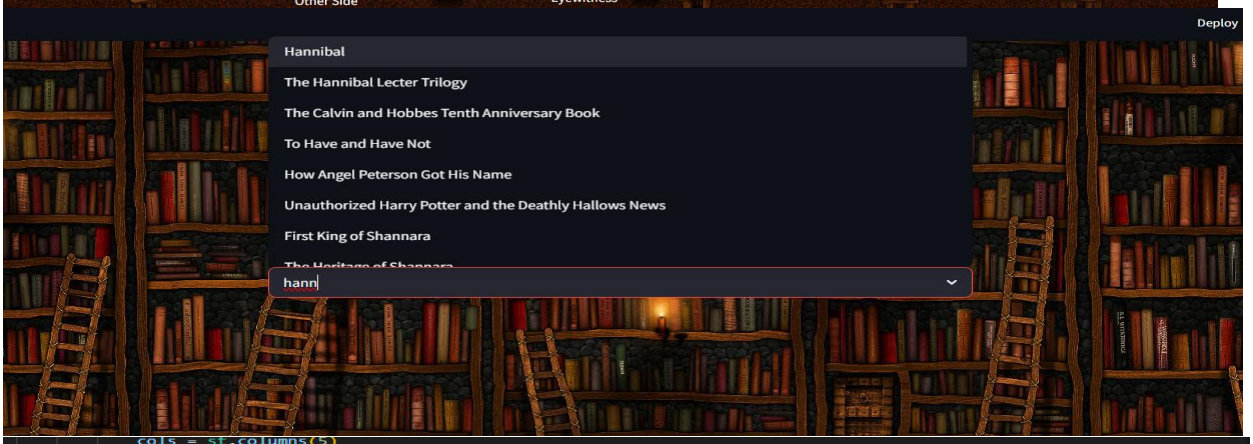
```
Recommendations based on book title:
```

	title	authors	categories	average_rating
5527	The Alienist	Machado de Assis	Mental illness	4.05
2636	The Hannibal Lecter Trilogy	Thomas Harris	Lecter, Hannibal (Fictitious character)	4.42
2995	Black Sunday	Thomas Harris	Fiction	3.60
6510	Red Dragon	Thomas Harris	Adventure stories	4.48
917	One Flew Over the Cuckoo's Nest	Ken Kesey	Mentally ill	4.19

```
Top books in category 'Fantasy':
```

	title	authors	categories	average_rating	ratings_count
1765	The Fellowship of the Ring	John Ronald Reuel Tolkien	Fantasy fiction	4.52	532629.0






```
            cols = st.columns(5)
            for idx, (_, row) in enumerate(recommended_books.iterrows):
                with cols[idx]:
                    st.image(row['thumbnail'], width=120)
                    st.write(f"{row['title']}")

else:  # By Category
    # Get unique categories
    all_categories = sorted(list(set([cat.strip() for sublist in books['categories'].str.split(',').dropna() for cat in sub

    # Create category search with auto-complete that works from first character
    category = st.selectbox(
        "Search for a category:",
        options=[""] + all_categories,
        format_func=lambda x: "Type to search..." if x == "" else x
    )

    # Show recommendations immediately when a category is selected
    if category:
        top_books, error = recommend_by_category(category, books, 5)
        if error:
            st.error(error)
        elif not top_books.empty:
            st.subheader(f"Top {5} books in '{category}' category:")
            cols = st.columns(5)
            for idx, (_, row) in enumerate(top_books.iterrows):
                with cols[idx]:
                    st.image(row['thumbnail'], width=120)
                    st.write(f"{row['title']}")

# streamlit run book_recommender.py
```

THANK YOU