Neelesh Talasila
CS210
Section 7
GitHub link: https://github.com/NeeleshTalasila/BookRec/tree/main

Book Recommendation Project:

In today's day and age many people find it difficult to read. According to statistca.com 46% of Americans said they didn't read this year. A contributing factor of that is that people don't know how to find the books they want because typically you don't see them advertised as much as say a movie or tv show. In order to address this issue I created a Book Recommendation System that will identify the user's preferences from the dataset, and create the most optimal recommendation of books from a dataset of books.

By personalizing recommendations using individual user preferences and past behaviors, the system will promote ideal book discovery. Aspects include leveraging structured datasets, machine learning models, and database queries to generate three types of recommendations: content-based, rating-based, and hybrid. The project ties closely to data management themes discussed in the course. It demonstrates the end-to-end process of data collection, transformation, storage, and analysis. The system integrates diverse components of data science, from SQL-based database management to machine learning algorithms. This provides a very good solution to this widespread problem in modern day society.

The most ideal recommendation approach I came up with was a hybrid between content-based filtering and rating-based filtering. This gave the most accurate and personalized recommendations.Unlike conventional systems that rely on a single method, the hybrid approach capitalizes on the strengths of each technique. Content-based filtering ensures recommendations are relevant to a user's preferences, while rating-based filtering introduces serendipitous discoveries by analyzing ratings from similar users.
This project is particularly important because it addresses a key limitation in many existing systems: the lack of user-centric recommendations that factor in nuanced preferences like preferred decades of publication or inclination toward local versus foreign books. The use of synthetic datasets allows for controlled experimentation and simulation of diverse scenarios. By integrating these features, the system provides a robust framework that could be adapted to real-world applications in e-commerce, streaming services, and more.

Existing research highlights the challenges of implementing effective recommendation systems, particularly in balancing personalization with exploration. This project builds upon prior works by introducing additional layers of user-specific preferences, such as genre prioritization and author preferences, ensuring a richer and more customized recommendation experience. The project utilizes synthetic datasets generated programmatically using Python libraries such as Faker. The book dataset includes attributes like title, author, genre, year of publication, country, and user ratings. To ensure realism, 50% of the books originate from the United States, with the remainder distributed across countries like India, the UK, and Japan. User profiles capture details such as age, gender, preferred genres, preferred authors, and more.
The following code snippet illustrates how metadata for books was constructed for content-based filtering:

Neelesh Talasila
CS210
Section 7
GitHub link: https://github.com/NeeleshTalasila/BookRec/tree/main

```
# Combine metadata for content-based filtering
books_df['metadata'] = (
    books_df['Genre'] + " " +
    books_df['Author'] + " " +
    books_df['Country'] + " " +
    books_df['Year_of_Publication'].astype(str)
)
```

This metadata serves as the foundation for text-based similarity computations, enabling the recommendation system to identify books similar to a user's preferences.

The system employs three main recommendation techniques:

Content-Based Filtering: This method uses TF-IDF vectorization to convert metadata into numerical representations, followed by cosine similarity to measure the closeness of books. For instance:

```
# TF-IDF Vectorizer for content-based filtering (helps create my sythetic data)
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(books_df['metadata'])
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

Books with the highest similarity scores are recommended to users based on their metadata and preferences.

Another technique was Rating-Based Filtering:

Using the Surprise library's Singular Value Decomposition (SVD), the system predicts user ratings for unrated books based on collaborative filtering principles. The following snippet demonstrates this process:

```
# Rating-based recommendation function
reader = Reader(rating_scale=(1, 10))
data = Dataset.load_from_df(ratings_df[['User_ID', 'ISBN', 'Rating']], reader)
trainset, testset = train_test_split(data, test_size=0.2)
model = SVD()
model.fit(trainset)
```

The final technique was the Hybrid Approach:

By combining content-based and rating-based methods, this approach merges results to generate a balanced set of recommendations. The hybrid recommendations are prioritized based on relevance and predicted ratings:

Neelesh Talasila
CS210
Section 7
GitHub link: https://github.com/NeeleshTalasila/BookRec/tree/main

```python
# Hybrid recommendation
def recommend_books_hybrid(user_id, top_n=5):
    content_recommendations = recommend_books_content(user_id, top_n)
    rating_recommendations = recommend_books_rating(user_id, top_n)
    if isinstance(content_recommendations, str):
        return rating_recommendations
    hybrid_recommendations = pd.concat([content_recommendations, rating_recommendations])
    hybrid_recommendations = hybrid_recommendations.drop_duplicates(subset='Title').head(top_n)
    return hybrid_recommendations
```

Data storage and retrieval are handled using SQLite. The synthetic datasets are stored in structured tables, and SQL queries facilitate data manipulation and analysis. For example, retrieving user preferences is as simple as:

```python
# Display user details
def display_user_details(user_id):
    user_preferences = pd.read_sql_query(f"SELECT * FROM Users WHERE User_ID = {user_id}", conn).iloc[0]
    print("User Details:")
    print(f"User ID: {user_preferences['User_ID']}")
    print(f"Name: User_{user_preferences['User_ID']}")  # Simulated name
    print(f"Age: {user_preferences['Age']}")
    print(f"Gender: {user_preferences['Gender']}")
    print(f"Country: {user_preferences['Country']}")
    print(f"Prefers High Rated Books: {user_preferences['Prefers_High_Rated_Books']}")
    print(f"Preferred Genres: {user_preferences['Preferred_Genres']}")
    print(f"Preferred Authors: {user_preferences['Preferred_Authors']}")
    print(f"Preferred Decade: {user_preferences['Preferred_Decade']}")
    print(f"Likes Foreign Books: {user_preferences['Likes_Foreign_Books']}")
    print("-" * 50)
```

This integration ensures efficient data handling and aligns with best practices in database management.

Experiments involved testing the system with various user profiles to validate the effectiveness of the recommendations. For example, a user who prefers high-rated books by specific authors in the mystery genre was provided with tailored suggestions based on both content similarity and collaborative filtering predictions. The results confirmed the hypothesis that a hybrid approach outperforms standalone methods in accuracy and user satisfaction. Users received diverse recommendations that balanced familiarity with novelty, addressing both relevance and serendipity.

The system's key advantage is its flexibility and robustness, achieved through the integration of multiple recommendation strategies. However, the reliance on synthetic data, while advantageous for controlled testing, may not fully replicate real-world complexities. Additionally, content-based filtering can struggle with sparse metadata, and collaborative filtering relies on sufficient rating data for accuracy.

Initially, the project intended to use real-world datasets. However, the absence of suitable data led to the generation of synthetic datasets. This change allowed for greater control and customization, enabling the inclusion of attributes like preferred decades and foreign book preferences. Another deviation was the addition of hybrid recommendations, which enhanced the system's versatility.

Neelesh Talasila
CS210
Section 7
GitHub link: https://github.com/NeeleshTalasila/BookRec/tree/main

This project demonstrates the effective integration of data management, machine learning, and database systems to solve a real-world problem. The hybrid Book Recommendation System offers personalized and meaningful suggestions, leveraging structured data, advanced algorithms, and user-specific preferences. By addressing limitations and iterating further, the system holds the potential for real-world applications, enhancing user experiences in various domains.