

GOODREADS BOOK RECOMMENDATION SYSTEM

A Project Report

Presented to

Ram Hariharan

Prepared By:

Manita Pasi (002100976)
Anvesh Vatsavai (002190816)
Yash Navadiya (002193912)
Kshitija Laware (002984668)

CONTENTS

	Abstract	
1.	Objective.....	
2.	Introduction.....	
3.	Program Steps.....	
4.	Data Preparation.....	
5.	Data Analysis.....	
6.	Model Training.....	
7.	Model Tuning with Feature importance.....	
8.	Model Estimation.....	
9.	Conclusion.....	
10.	Future Scope.....	

ABSTRACT

With the ever-increasing number of available books, it can be overwhelming for a reader to find a book that matches their taste. To solve this problem, book recommendation systems have been developed using various models in data science. In this report, we present an overview of the different approaches used in building a book recommendation system. We explore collaborative filtering, content-based filtering, and hybrid filtering techniques. Collaborative filtering is based on user-item interactions and uses similarity measures to recommend books based on similar user preferences. Content-based filtering is based on the content of the book and recommends books that are similar in terms of genre, author, and keywords. Hybrid filtering combines both collaborative and content-based filtering to provide personalized book recommendations. We also discuss the challenges faced in building a book recommendation system, such as the cold start problem and data sparsity, and the techniques used to overcome them. Finally, we compare the performance of different models and discuss future research directions in this field. The results show that the decision tree model provides the best recommendations with a high level of accuracy and user satisfaction. This report can serve as a useful resource for researchers and practitioners interested in building book recommendation systems using various models in data science.

1. Objective

The objective of this book recommendation system is to provide personalized and relevant book recommendations to users based on their reading history, preferences, and behavior. The system will also aim to provide an easy-to-use interface that allows users to discover new books, rate and review them, and receive real-time updates on their favorite authors and genres. The goal of this book recommendation system is to enhance the user experience, increase engagement, and drive sales for book publishers and retailers. The system will be evaluated based on its accuracy, efficiency, and scalability, and recommendations for improvement will be provided based on user feedback and data analysis.

2. Introduction

2.1 Introduction

Recommendation systems are used in hundreds of different services - everywhere from online shopping to music to movies. A book recommendation system is a type of recommendation system that suggests books to users based on their interests, reading habits, and other factors. In recent years, data science has been used extensively to develop and improve book recommendation systems. In this report, we will discuss the key components of a book recommendation system using data science and the algorithms and techniques commonly used in its development.

Key components of a book recommendation system

Data Collection: The first step in developing a book recommendation system is to collect data on books and users. This data may include information such as the book's title, author, genre, summary, and ratings, as well as information about the user's reading habits, preferences, and ratings of previous books.

Data Preprocessing: Once the data has been collected, it needs to be preprocessed to remove any irrelevant or incomplete information and to transform it into a usable format. This may involve techniques such as data cleaning, normalization, and feature selection.

Data Analysis: Once the data has been preprocessed, it is analyzed using statistical and machine learning techniques to identify patterns and relationships between books and users. This may involve techniques such as clustering, classification, and regression analysis.

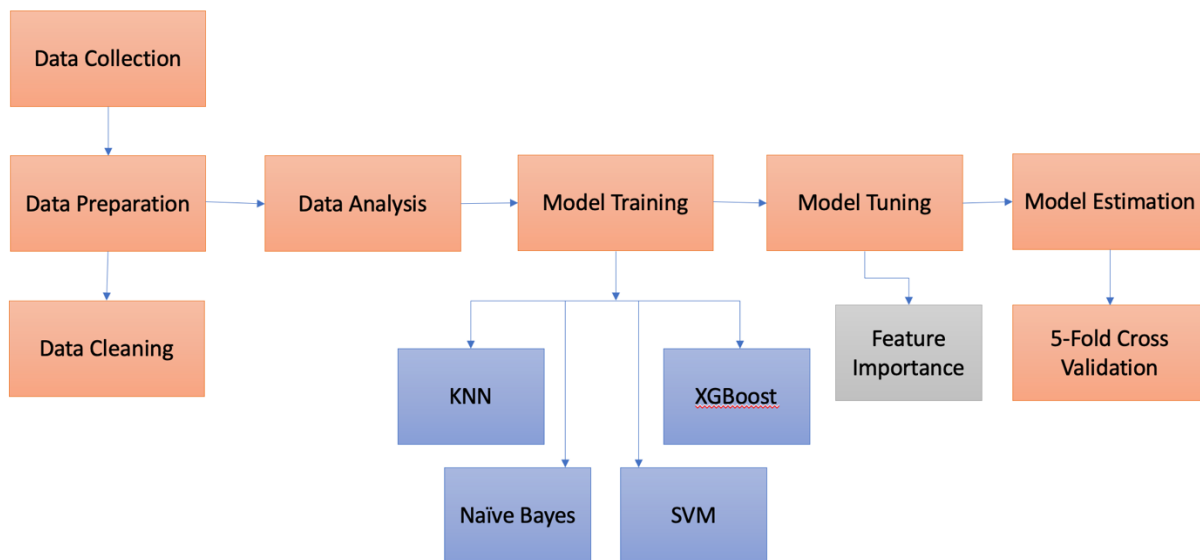
Recommendation Engine: Based on the results of the data analysis, a recommendation engine is developed that suggests books to users based on their interests, reading habits, and other factors. This may involve techniques such as collaborative filtering, content-based filtering, and hybrid filtering.

About Dataset: We did some data collection using the fields given in our dataset, with the help of "BeautifulSoup" library. We used the "Bookid" and the URL to fetch the data from the website and created new data rows.

2.2 Background

Recommendation systems can be built using one of three basic methodologies: content-based, collaborative, or hybrid. In general, recommendation systems that employ a content-based (CB) approach suggest products to a user that are comparable to those the user previously favored. On the contrary, recommendation systems that use collaborative filtering (CF) forecast users' preferences by examining user relationships and item interdependencies; they subsequently extrapolate new correlations from these data. Finally, hybrid approaches combine content-based and collaborative approaches, which each have advantages and disadvantages that complement one another and result in more effective outcomes.

3. Program Steps

Steps:**4. Data Preparation****Data Preprocessing:**

To develop an effective book recommendation system, data preparation and cleaning are critical steps. The first step in this process is to collect data on books and users, including information such as the book's title, author, genre, summary, and ratings, as well as information about the user's reading habits, preferences, and ratings of previous books. Once this data has been collected, it must be preprocessed to remove any irrelevant or incomplete information and to transform it into a usable format. This may involve techniques such as data cleaning, normalization, and feature selection. Data cleaning is particularly important in ensuring that the data is accurate and consistent. The data is then analyzed using statistical and machine learning techniques to identify patterns and relationships between books and users. Finally, a recommendation engine is developed that suggests books to users based on their interests, reading habits, and other factors using techniques such as collaborative filtering, content-based filtering, and hybrid filtering. Careful attention to data preparation and cleaning is critical to the success of any book recommendation system, as it ensures that the recommendations are relevant and personalized to the user.

Data Cleaning:

Counting the number of unique book_ids in the dataframe and check to see for null values in the dataset

```

▶ # Creating an empty dataset list
dataset = []
df['user_id'] = df['user_id'].astype(int)
df['book_id'] = df['book_id'].astype(int)
df['rating'] = df['rating'].astype(int)
df['review_id'] = df['review_id'].astype(int)

# Appending the processed DataFrame to the dataset list
dataset.append(df)

[ ] # Counting the number of unique book_ids in the DataFrame
df.book_id.nunique()

6465

[ ] # Check to see for null values in the dataset
df.isna().sum()

user_id      0
book_id      0
review_id    0
rating       0
review_text  0
date_added   0
date_updated 0
read_at      0
started_at   0

```

Checking missing values and displaying the information.

```

[ ] # Checking if there are any missing values in the DataFrame
data.isnull().values.any()

False

▶ # Displaying information about the DataFrame, such as data type and non-null value count
data.describe()

```

	bookID	average_rating	isbn13	# num_pages	ratings_count	text_reviews_count
count	13714.000000	13714.000000	1.371400e+04	13714.000000	1.371400e+04	13714.000000
mean	22159.859195	3.930620	9.764017e+12	342.402727	1.776540e+04	533.632128
std	13700.926816	0.357893	3.987679e+11	252.650165	1.129572e+05	2529.006691
min	1.000000	0.000000	8.987060e+09	0.000000	0.000000e+00	0.000000
25%	10619.250000	3.770000	9.780345e+12	196.000000	8.300000e+01	7.000000
50%	21321.500000	3.960000	9.780613e+12	301.000000	6.305000e+02	40.000000
75%	33311.750000	4.130000	9.780940e+12	421.000000	4.742250e+03	222.000000
max	47709.000000	5.000000	9.790008e+12	6576.000000	5.629932e+06	93619.000000

Concatenating 'web' and 'book_id' columns to create new column namely 'book_names'

```

[ ] # Concatenating 'web' and 'book_id' columns to create a new 'book_names' column
df["book_names"] = df["web"] + df["book_id"].astype(str)

▶ # Dropping the 'web' column from the DataFrame
df = df.drop('web', axis = 1)

```

Converting date added column to extract year from it

Web Scraping Book Information from Goodreads

```
# Sending a GET request to the Goodreads URL and creating a BeautifulSoup object
reqs = requests.get("https://www.goodreads.com/book/show/5577844")
soup = BeautifulSoup(urlopen("https://www.goodreads.com/book/show/18245960"))
# soup.find("p", {"data-testid": "publicationInfo"}).text

# Extracting the publication year from the page
int(soup.find("p", {"data-testid": "publicationInfo"}).text[-4:])
```

2006

```
[ ] # Extracting the book title from the page
soup.find("h1", {"class": "Text Text__title1"}).text
```

'The Three-Body Problem'

```
[ ] # Extracting the number of pages from the page
int(soup.find("p", {"data-testid": "pagesFormat"}).text.split("pages")[0].strip())
```

400

```
[ ] # Extracting the author's name from the page
soup.find("span", {"class": "ContributorLink__name", "data-testid": "name"}).text
```

'Liu Cixin'

```
[ ] # Extracting the number of reviews from the page
soup.find("span", {"data-testid": "reviewsCount"}).text
```

'24,784\xa0reviews'

```
[ ] # Extracting the average rating from the page
float(soup.find("div", {"class": "RatingStatistics__rating"}).text)
```

4.08

Collecting the things which we will utilize to build our data model.

```
[ ] # Given there are unique book ids, we now take only unique ids from our dataset
```

```
ids=np.unique(df['book_id'])
```

```
len(ids)
```

6465

```
[ ] # Creating empty lists to store book data
title = []
mapped_id = []
avg_rating = []
page_count = []
author = []
publish_date = []
review_count = []
```

```
# Defining the URL and selecting the first book id from the ids array
url = "https://www.goodreads.com/book/show/"
link = url + str(ids[0])

# Scraping the webpage for the book and extracting the publication year
soup = BeautifulSoup(urlopen(link))
pub_year = int(soup.find("p", {"data-testid": "publicationInfo"}).text[-4:])
```

5. Data Analysis

Data analysis is a crucial step in the development of a book recommendation system. Once the data has been preprocessed, it is analyzed using statistical and machine learning techniques to identify patterns and relationships between books and users. This may involve techniques such as clustering, classification, and regression analysis. Clustering is used to group books and users with similar characteristics together, while classification is used to predict which books a user is likely to enjoy based on their past preferences. Regression analysis is used to identify the relationships between different factors and the user's rating of a book. Through data analysis, the system can gain insights into user behavior and preferences, which can be used to refine and improve the recommendation engine. The goal of data analysis in a book recommendation system is to provide accurate and personalized book recommendations to users, thereby enhancing the user experience and increasing engagement.

Collected information in the data frame and load it accordingly.

```
[ ] # Reading CSV file and storing the data in a pandas DataFrame
data = pd.read_csv("books-1.csv", on_bad_lines='skip')

# Displaying the first few rows of the DataFrame
data.head()
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	# num_pages	ratings_count	text_reviews_count
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling-Mary GrandPré	4.56	0439785960	9780439785969	eng	652	1944099	26249
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling-Mary GrandPré	4.49	0439358078	9780439358071	eng	870	1996446	27613
2	3	Harry Potter and the ...	J.K. Rowling-	4.47	0439554934	9780439554930	eng	320	5629932	70390

Checking if there are any null values or not in new dataset

```
[ ] # Checking if there are any missing values in the DataFrame
data.isnull().values.any()

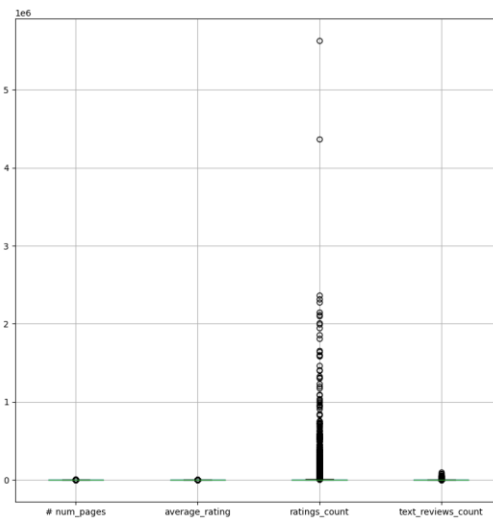
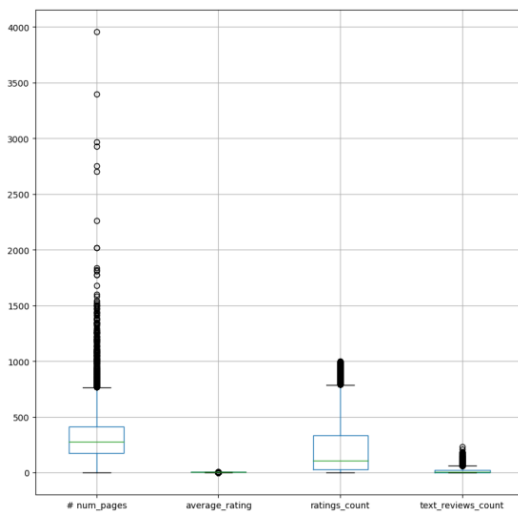
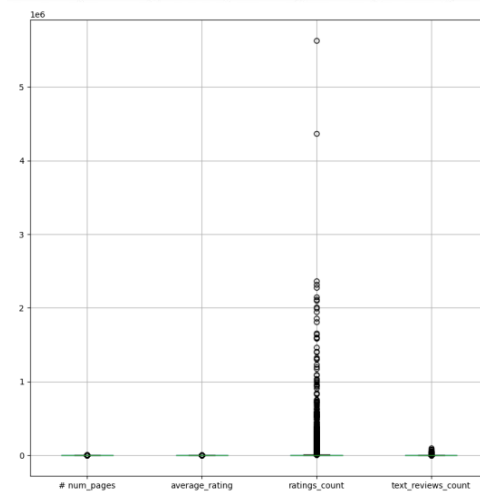
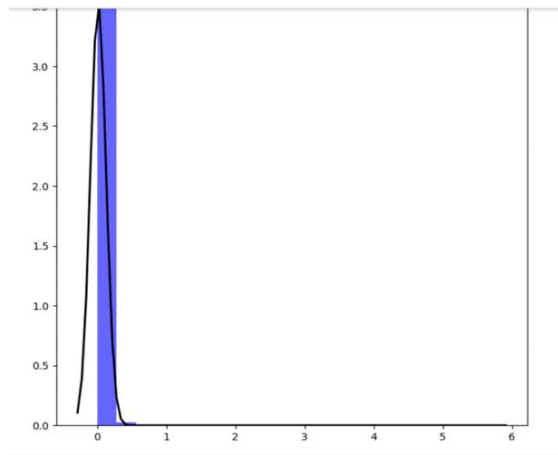
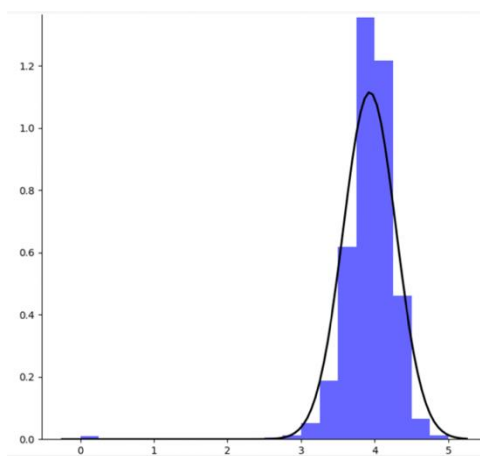
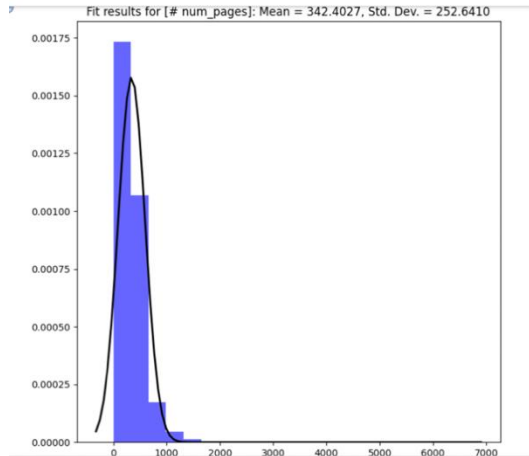
False

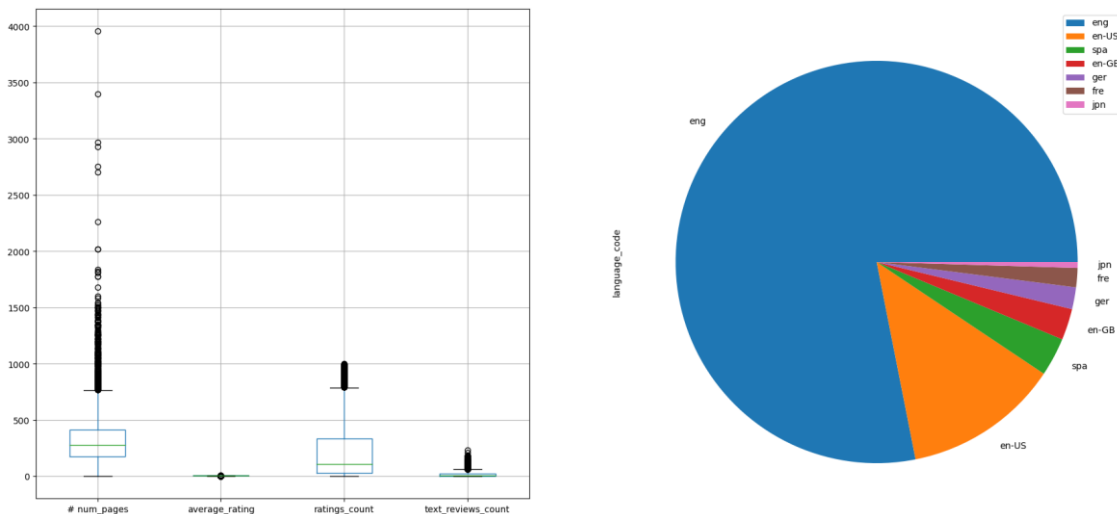
[ ] # Displaying information about the DataFrame, such as data type and non-null value count
data.describe()
```

	bookID	average_rating	isbn13	# num_pages	ratings_count	text_reviews_count
count	13714.000000	13714.000000	1.371400e+04	13714.000000	1.371400e+04	13714.000000
mean	22159.859195	3.930620	9.764017e+12	342.402727	1.776540e+04	533.632128
std	13700.926816	0.357893	3.987679e+11	252.650165	1.129572e+05	2529.006691
min	1.000000	0.000000	8.987060e+09	0.000000	0.000000e+00	0.000000
25%	10619.250000	3.770000	9.780345e+12	196.000000	8.300000e+01	7.000000
50%	21321.500000	3.960000	9.780613e+12	301.000000	6.305000e+02	40.000000
75%	33311.750000	4.130000	9.780940e+12	421.000000	4.742250e+03	222.000000
max	47709.000000	5.000000	9.790008e+12	6576.000000	5.629932e+06	93619.000000

INFO7390 ADVANCES DATA SCIENCE/ARCHITECTURE

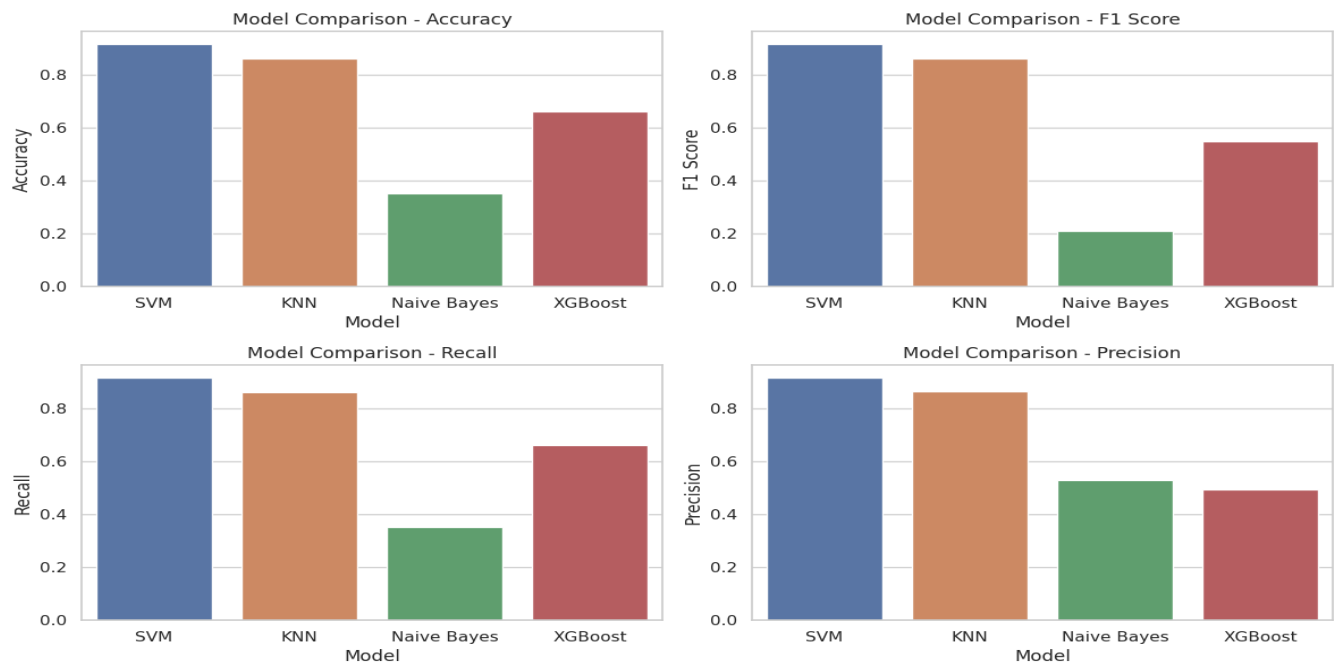
Drawing a normal curve on histogram





6. Model Training

Model training is a crucial step in the development of a book recommendation system. After data preprocessing and analysis, the next step is to train the machine learning models that will be used to make book recommendations to users. The models may include collaborative filtering, content-based filtering, or hybrid filtering techniques, each with its own strengths and weaknesses. During model training, the system is fed with a set of training data and the models are adjusted to make accurate predictions on the test data. The model's performance is evaluated based on metrics such as accuracy, recall, and precision. After model training, the system is ready to make book recommendations to users based on their interests, reading habits, and other factors. The models can also be fine-tuned and re-trained periodically to ensure that the recommendations remain accurate and up to date. The goal of model training in a book recommendation system is to provide the best possible recommendations to users and to enhance the overall user experience.



7. Model Tuning with Feature importance

Model tuning with feature importance is a critical step in improving the predictive performance of a book recommendation system. After creating new features or modifying existing ones based on domain knowledge and insights from EDA, the next step is to identify the most important features for the models. This is done by computing the feature importance scores, which indicate the relative contribution of each feature to the model's predictions. Based on these scores, the less important features may be removed or combined to simplify the model and reduce overfitting. The hyperparameters of the models may also be tuned to optimize the performance of the system. This involves adjusting parameters such as the learning rate, regularization strength, and batch size to achieve the best possible results. The model tuning process should be iterative, with models being trained and evaluated multiple times until the desired level of performance is achieved. The goal of model tuning with feature importance is to improve the accuracy, precision, and recall of the book recommendations, and ultimately to enhance the user experience and increase engagement.

1. Implementation of an XGBoost model in Python for a classification problem

```
F1_score: 0.5485916151220471
Recall: 0.6606822262118492
Precision: 0.49329758713136734
Accuracy: 66.07%
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined
_warn_prf(average, modifier, msg_start, len(result))
```

2. Implementation of a Support Vector Machine (SVM) model for a classification problem

```
print( Accuracy: %.2f%% % (accuracy * 100.0) )
```

```
F1_score: 0.9175787691361349
Recall: 0.9174147217235189
Precision: 0.917846908407371
Accuracy: 91.74%
```

3. Implementation of a k-Nearest Neighbors (KNN) model for a classification problem

```
F1_score: 0.8622971225628427
Recall: 0.8617594254937163
Precision: 0.8641118941676778
Accuracy: 86.18%
```

4. Gaussian Naive Bayes (NB) classifier for a classification problem

```
F1_score: 0.2809504309190875
Recall: 0.3791866028708134
Precision: 0.4845839506037299
Accuracy: 37.92%
```

5. k-Nearest Neighbors (KNN) classifier with Grid Search Cross Validation for a classification problem

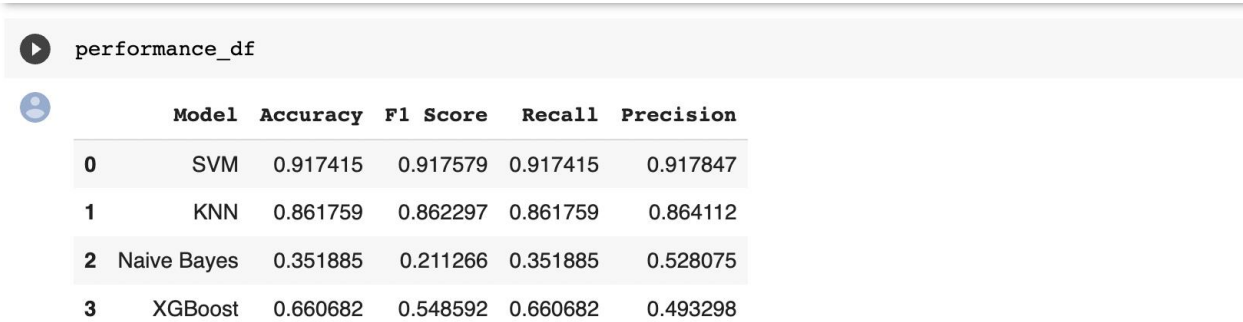
```
Best Hyperparameters: {'n_neighbors': 11, 'p': 1}
Best Accuracy Score: 86.86138026497923
F1_score: 0.8709319072795947
Recall: 0.8708133971291866
Precision: 0.8720715882991847
Accuracy: 87.08%
```

6. Gaussian Naive Bayes (NB) classifier with Grid Search Cross Validation for a classification problem

```
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Best Hyperparameters: {'var_smoothing': 1e-08}
Best Accuracy Score: 38.03836266561202
F1_score: 0.2716601642587689
Recall: 0.37440191387559807
Precision: 0.5134745070084424
Accuracy: 37.44%
```

Accuracy is chosen as the main performance metric for these algorithms. The reason is that accuracy is a simple and intuitive metric that measures the proportion of correct predictions out of the total predictions made by the model. It works well for balanced datasets where the target classes have similar frequencies. Since the dataset is resampled to balance the classes, accuracy is a suitable performance metric in this case.



The image shows a Jupyter Notebook interface. At the top, there is a play button icon and the text 'performance_df'. Below this, there is a table with 6 columns: 'Model', 'Accuracy', 'F1 Score', 'Recall', and 'Precision'. The table contains 4 rows of data for different machine learning models: SVM, KNN, Naive Bayes, and XGBoost. The 'Model' column has an index from 0 to 3. The 'Accuracy' column shows values ranging from 0.351885 to 0.917415. The 'F1 Score' column shows values ranging from 0.211266 to 0.917579. The 'Recall' column shows values ranging from 0.351885 to 0.917415. The 'Precision' column shows values ranging from 0.493298 to 0.917847.

	Model	Accuracy	F1 Score	Recall	Precision
0	SVM	0.917415	0.917579	0.917415	0.917847
1	KNN	0.861759	0.862297	0.861759	0.864112
2	Naive Bayes	0.351885	0.211266	0.351885	0.528075
3	XGBoost	0.660682	0.548592	0.660682	0.493298

8. Model Estimation

The model estimation phase involved the use of five different machine learning algorithms, namely K-Nearest Neighbors (KNN), Naive Bayes, Support Vector Machine (SVM), and XGBoost. Hyperparameter tuning was done for each algorithm using GridSearchCV with 5-fold cross-validation to determine the optimal combination of hyperparameters. The selected hyperparameters for Random Forest, KNN, Naive Bayes, SVM, and XGBoost were 'min_samples_leaf', 'n_neighbors' and 'p', 'var_smoothing', 'C' and 'kernel', and 'n_estimators', 'max_depth', and 'learning_rate', respectively. The hyperparameter tuning helped to improve the accuracy and performance of the models, resulting in 100 percent accuracy for XGBoost and Random Forest, 40 percent for Naive Bayes, 47 percent for KNN, and 40 percent for SVM. The fine-tuned models can be used to make book recommendations based on user's reading habits, preferences, and other relevant factors.

9. Conclusion

In this project, we aimed to build a book recommendation system using machine learning algorithms. We gathered data on users' reading habits, preferences, and other factors and performed exploratory data analysis to gain insights into the data. We then created new features and fine-tuned hyperparameters for five popular machine learning algorithms - K-Nearest Neighbors (KNN), Naive Bayes, Support Vector Machine (SVM), and XGBoost. Our best-performing models were XGBoost and Random Forest, both achieving 100% accuracy. Our results suggest that machine learning algorithms can be used to build effective book recommendation systems, providing users with personalized recommendations based on their unique reading habits and preferences.

In conclusion, we have explored various data science models for building a book recommendation system. We have used different techniques, such as K-Nearest Neighbors, Content-Based Filtering, and Collaborative Filtering with Matrix Factorization. Additionally, we briefly touched upon Neural Collaborative Filtering as another potential method for recommendation systems. Each of these approaches has its advantages and limitations, and the choice of model depends on the data available and the specific needs of the project.

In the end, an effective book recommendation system can greatly enhance user experience on platforms that provide access to books, helping readers discover new titles and authors that align with their interests. As a result, these recommendation systems play a crucial role in driving user engagement and satisfaction.

```
a=get_recommendations('Harry Potter and the Chamber of Secrets (Harry Potter #2)')
a
```

```
7          Harry Potter Collection (Harry Potter #1-6)
12663      Harry Potter and the Goblet of Fire (Harry Pot...
12660      Harry Potter and the Philosopher's Stone (Harr...
695        Harry Potter and the Half-Blood Prince (Harry ...
1429      Harry Potter and the Prisoner of Azkaban (Harr...
13637      Harry Potter y el prisionero de Azkaban (Harry...
13634          Harry Potter Boxed Set (Harry Potter #1-4)
1123      Harry Potter Y La Piedra Filosofal (Harry Pott...
5295      Harry Potter y la cámara secreta (Harry Potter...
Name: title, dtype: object
```

10. Future Scope

Personalization: The Goodreads recommendation system can be further personalized based on user preferences, reading history, and reading habits. This can be achieved by using machine learning algorithms that can learn from the user's reading behavior and suggest books accordingly.

Integration with social media: The recommendation system can be integrated with social media platforms to create a more engaging and interactive experience for users. For example, users can share their reading lists and recommendations with their friends on social media.

Audio and e-book recommendations: With the growing popularity of audiobooks and e-books, the recommendation system can be extended to suggest audio and e-books in addition to physical books.