A Project Report On

# Bookify: The Ultimate Book Recommendation Application with Data-Driven Intelligence

*Submitted by*

Divya Pondkule (PL)

(220940325029)

Dhiraj Mankar

(220940325026)

Rhutuja Wagh

(220940325054)

Twinkal Patil

(220940325084)

Deepam Gharsele

(220940325025)

*In partial fulfilment of*

*the requirements for the award of the degree of*

**Post Graduate Diploma**

**In**

**Big Data Analytics**

**(PG-DBDA)**

**Year 2023**

# Abstract

The rapid growth of online bookstores has led to an increase in the number of books available for purchase, making it difficult for users to discover books that match their interests.

To address this challenge, we propose a Bookify: The Ultimate Book Recommendation Application with Data-Driven Intelligence that leverages user behaviour and book metadata to provide tailored book recommendations.

# Table of Contents

# 1. Introduction

The growing availability of books online has made it challenging for readers to find books that match their interests. Bookstores and online platforms are inundated with numerous books, making it difficult for readers to navigate through them and choose books that fit their preferences. To help readers discover books they will enjoy, book recommendation systems have been developed to provide popularity recommendations based on user preferences and book metadata.

In this report, we propose a book recommendation system that utilizes user preferences and book metadata to provide personalized recommendations. Our system collects data on user reading history, ratings, and preferences to build user profiles. We use a collaborative-based filtering algorithm to analyse user profiles and recommend books that have similar content and features to those previously enjoyed by the user. Additionally, our system incorporates book metadata such as genre, author, and publisher to recommend books that match the user's interests.

# 2. Objectives

The objective of the book recommendation system project report is to design and implement an intelligent algorithm that can suggest books to users based on their reading preferences. The system should be able to analyse the user's reading history, preferences, and feedback to recommend books that are most likely to interest them.

The project aims to develop a system that can accurately predict the user's reading choices and provide personalized recommendations. The system should be user-friendly, scalable, and efficient in handling large amounts of data.

The project report will include an analysis of various recommendation algorithms, their strengths, and limitations. It will also include a detailed description of the system's architecture, data collection, and evaluation metrics.

The project aims to create a book recommendation system that provides an enhanced user experience and encourages users to explore new genres and authors. The report will also discuss potential future enhancements and developments of the system.

# 3. Research Motivation

The availability of online books has increased exponentially in recent years, making it difficult for readers to navigate through the vast selection of books available and find books that match their interests. Consequently, there is a need for book recommendation systems that can provide personalized recommendations to readers. These systems can help readers discover books that they will enjoy.

The project's motivation is to provide readers with book recommendations that match their interests and preferences. We are recommending books using

popularity based filtering (showing top 50 books in entire collection and books popular yearly) and using collaborative filtering approach in which we have implemented k-NN model.

# 4. Problem Statement

With the growing availability of books online, it has become increasingly difficult for readers to navigate through the vast selection of books and find books that match their interests. While traditional bookstores rely on staff recommendations, online bookstores have utilized recommendation systems to help users find books they will enjoy. However, many of these systems are limited in their ability to provide personalized recommendations based on user preferences and book metadata.

Therefore, the problem this project aims to address is to develop a book recommendation system that can provide tailored book recommendations to users based on book metadata. The system will utilize machine learning techniques to pre-process book data and extract relevant features from book descriptions and metadata.

# 5. Project Development

The book recommendation system project will involve several stages of development. First, the project team will collect and pre-process book data, including book metadata and descriptions. Next, they will develop machine learning models to analyse the data and extract relevant features from the book descriptions. The team will then develop a recommendation engine that utilizes collaborative filtering techniques to identify similar users and recommend books that those users have enjoyed. The system will also incorporate popularity-based filtering algorithms to match the frequency of usage and popular items of the books to the users. The final product will be a book recommendation system that provides book recommendations based on user preferences and book metadata.

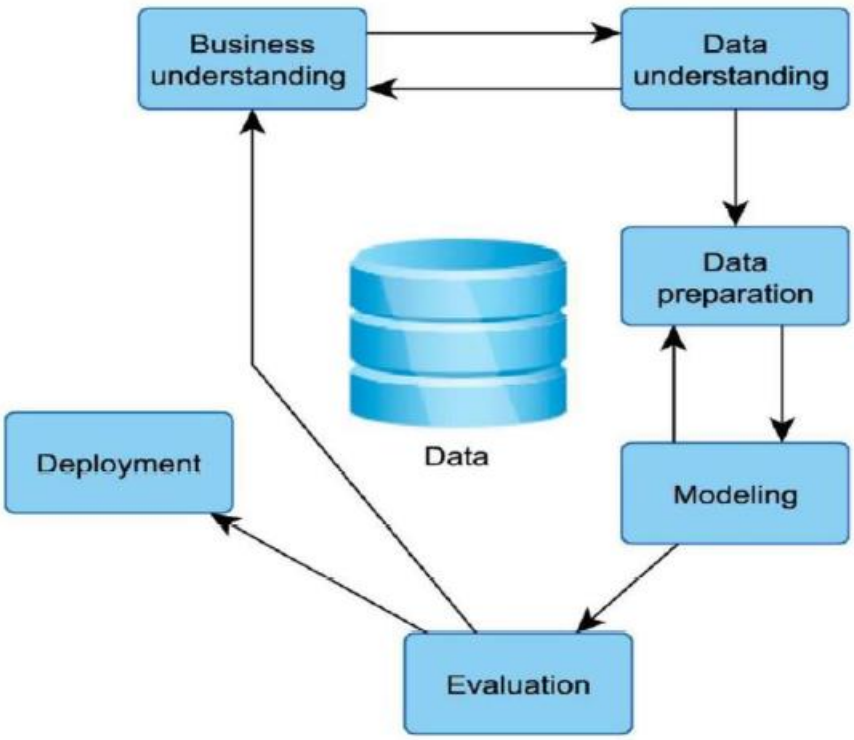



Figure 1: Project Development

# 6. Methodology and Implementation

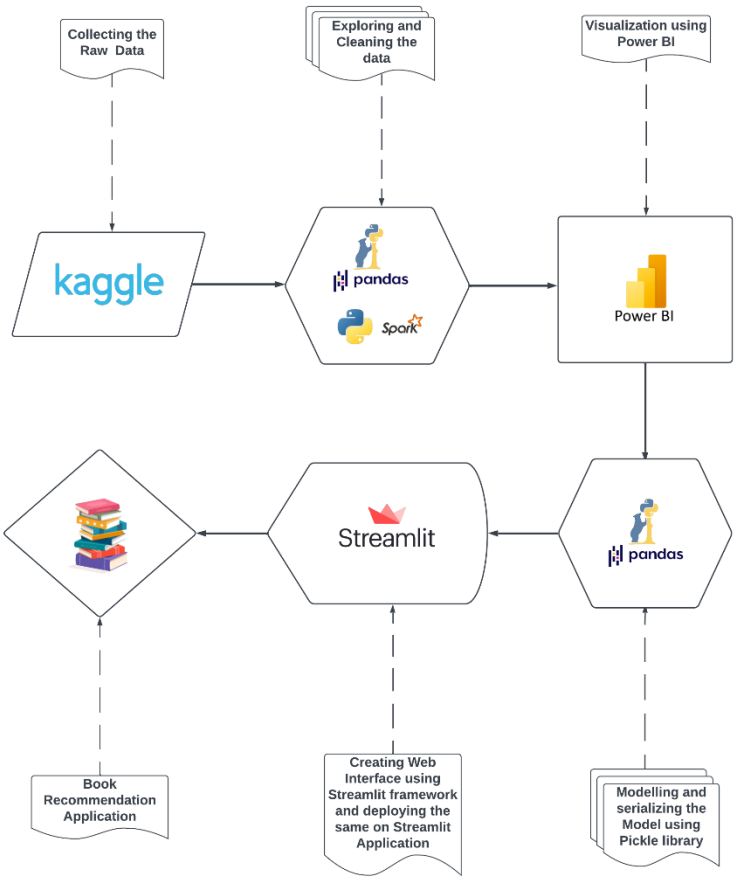



Figure 2: Project Architecture

## Technologies Used :

1. Pyspark
2. Python
3. Machine Learning
4. Streamlit
5. Power Bi

## A. Data Source:

The data source that we will be using in the is analysis is a dataset from Kaggle which contains the Book-Crossing dataset comprises 3 files.

- Users
  Contains the users and having rows approximately more than 0.2 million. Note that user IDs (User-ID) have been anonymized and map to integers. Demographic data is provided (Location, Age) if available. Otherwise, these fields contain NULL-values.

- Books
  Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. The dataset contains rows approximately more than 0.2 million. Moreover, some content-based information is given (Book-Title, Book-Author, Year-Of-Publication, Publisher), obtained from Amazon Web Services. Note that in case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavours (Image-URL-S, Image-URL-M, Image-URL-L), i.e., small, medium, large. These URLs point to the Amazon web site.

- Ratings
  Contains the book rating information and have rows approximately more than 1 million. Ratings (Book-Rating) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0.

**Users Data**

```
In [36]: users.head()
```

Out[36]:

| | User-ID | Location | Age |
|---|---|---|---|
| 0 | 1 | nyc, new york, usa | NaN |
| 1 | 2 | stockton, california, usa | 18.0 |
| 2 | 3 | moscow, yukon territory, russia | NaN |
| 3 | 4 | porto, v.n.gaia, portugal | 17.0 |
| 4 | 5 | farnborough, hants, united kingdom | NaN |

Figure :3

**Books Data**

```
In [35]: books.head()
```

Out[35]:

| | ISBN | Book-Title | Book-Author | Year-Of-Publication | Publisher | Image-URL-S | |
|---|---|---|---|---|---|---|---|
| 0 | 0195153448 | Classical Mythology | Mark P. O. Morford | 2002 | Oxford University Press | http://images.amazon.com/images/P/0195153448.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/01 |
| 1 | 0002005018 | Clara Callan | Richard Bruce Wright | 2001 | HarperFlamingo Canada | http://images.amazon.com/images/P/0002005018.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/00 |
| 2 | 0060973129 | Decision in Normandy | Carlo D'Este | 1991 | HarperPerennial | http://images.amazon.com/images/P/0060973129.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/00 |
| 3 | 0374157065 | Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the Virus That Caused It | Gina Bari Kolata | 1999 | Farrar Straus Giroux | http://images.amazon.com/images/P/0374157065.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/03 |
| 4 | 0393045218 | The Mummies of Urumchi | E. J. W. Barber | 1999 | W. W. Norton & Company | http://images.amazon.com/images/P/0393045218.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/03 |

Figure :4

**Ratings Data**

```
In [37]: ratings.head()
```

Out[37]:

| | User-ID | ISBN | Book-Rating |
|---|---|---|---|
| 0 | 276725 | 034545104X | 0 |
| 1 | 276726 | 0155061224 | 5 |
| 2 | 276727 | 0446520802 | 0 |
| 3 | 276729 | 052165615X | 3 |
| 4 | 276729 | 0521795028 | 6 |

Figure :5

## B. Data Storage

We have used local machine to store the data.

## C. Data Pre-processing

Data pre-processing is an essential step in developing a book recommendation system as it involves cleaning, transforming, and preparing raw data into a format that can be used for training and testing machine learning models.

1. **Books Dataset:** In this dataset, we checked for null values, duplicates and unique values. After that we checked that some values have been placed into the wrong columns. So, we shift the values into the respective rows and placed them in the right columns. Another observation is that few field have wrong information so we have corrected with proper values. After that we replaced the null values of 'Year of Publication' by mode, 'Book Author' and 'Publisher' column by 'Not Mentioned'.
2. **Users Dataset:** Most of the users are from the age group 25-50

   It is highly unlikely to have users under the age of 4 and above 100.The peaks near 0 and 100 in the kdeplot indicates that there are some outlier values in the 'Age' column. It is highly unlikely to have users of age above 100 and below in this case. so, we replace these values with np.nan.
3. **Ratings Dataset:** The dataset ratings doesn't need any pre-processing.

We extracted the clean data in the form of csv file in-order to show the insights of data by using **pyspark**:

```
import pyspark
```

```
from pyspark.sql import SparkSession
```

```
# spark = SparkSession.builder.appName('BRS').getOrCreate()
spark = SparkSession.builder.appName("Spark SQL basic example").enableHiveSupport().getOrCreate()
```

Figure :6

```
# Books dataset
```

```
from pyspark.sql.types import StructType, StringType, IntegerType, DoubleType, LongType
```

```
schema1=StructType().add("ISBN",LongType(),True).add("BookTitle",StringType(),True).add("BookAuthor",StringType(),True).add("Year
```

```
print(schema1)
```

```
StructType([StructField('ISBN', LongType(), True), StructField('BookTitle', StringType(), True), StructField('BookAuthor', Stri
ngType(), True), StructField('YearOfPublication', IntegerType(), True), StructField('Publisher', StringType(), True), StructFie
ld('ImageURLS', StringType(), True), StructField('ImageURLM', StringType(), True), StructField('ImageURLL', StringType(), Tru
e)])
```

```
df_books=spark.read.format("csv").option("header","True").schema(schema1).load("C:/Users/pppon/BRS_FINAL/Cleaned Data/cleaned_boo
```

```
df_books.printSchema()
```

```
root
 |-- ISBN: long (nullable = true)
 |-- BookTitle: string (nullable = true)
 |-- BookAuthor: string (nullable = true)
 |-- YearOfPublication: integer (nullable = true)
 |-- Publisher: string (nullable = true)
 |-- ImageURLS: string (nullable = true)
 |-- ImageURLM: string (nullable = true)
 |-- ImageURLL: string (nullable = true)
```

Figure :7

```
# Users dataset
```

```
schema2=StructType().add("UserID",LongType(),True).add("Location",StringType(),True).add("Age",IntegerType(),True)
```

```
print(schema2)
```

```
StructType([StructField('UserID', LongType(), True), StructField('Location', StringType(), True), StructField('Age', IntegerTyp
e(), True)])
```

```
df_users=spark.read.format("csv").option("header","True").schema(schema2).load("C:/Users/pppon/BRS_FINAL/Cleaned Data/cleaned_use
```

```
df_users.printSchema()
```

```
root
 |-- UserID: long (nullable = true)
 |-- Location: string (nullable = true)
 |-- Age: integer (nullable = true)
```

```
df_users.show()
```

```
+------+--------------------+---+
|UserID|            Location|Age|
+------+--------------------+---+
|     1|  nyc, new york, usa| 24|
|     2|stockton, califor...| 18|
|     3|moscow, yukon ter...| 24|
|     4|porto, v.n.gaia, ...| 17|
|     5|farnborough, hant...| 24|
|     6|santa monica, cal...| 61|
|     7| washington, dc, usa| 24|
|     8|timmins, ontario,...| 24|
|     9|germantown, tenne...| 24|
|    10|albacete, wiscons...| 26|
|    11|melbourne, victor...| 14|
|    12|fort bragg, calif...| 24|
|    13|barcelona, barcel...| 26|
|    14|mediapolis, iowa,...| 24|
|    15|calgary, alberta,...| 24|
|    16|albuquerque, new ...| 24|
|    17|chesapeake, virgi...| 24|
|    18|rio de janeiro, r...| 25|
|    19|           weston, ,| 14|
|    20|langhorne, pennsy...| 19|
+------+--------------------+---+
only showing top 20 rows
```

Figure :8

```
# Ratings dataset
```

```
schema3=StructType().add("UserID",LongType(),True).add("ISBN",LongType(),True).add("BookRating",IntegerType(),True)
```

```
print(schema3)
```

```
StructType([StructField('UserID', LongType(), True), StructField('ISBN', LongType(), True), StructField('BookRating', IntegerTy
pe(), True)])
```

```
df_ratings=spark.read.format("csv").option("header","True").schema(schema3).load("C:/Users/pppon/BRS_FINAL/Cleaned Data/cleaned_r
```

```
df_ratings.printSchema()
```

```
root
 |-- UserID: long (nullable = true)
 |-- ISBN: long (nullable = true)
 |-- BookRating: integer (nullable = true)
```

```
df_ratings.show()
```

```
+------+----------+----------+
|UserID|      ISBN|BookRating|
+------+----------+----------+
|276725|      null|         0|
|276726| 155061224|         5|
|276727| 446520802|         0|
|276729|      null|         3|
|276729| 521795028|         6|
|276733|2080674722|         0|
|276736|3257224281|         8|
|276737| 600570967|         6|
|276744|      null|         7|
|276745| 342310538|        10|
|276746| 425115801|         0|
|276746| 449006522|         0|
|276746| 553561618|         0|
|276746|      null|         0|
|276746| 786013990|         0|
|276746| 786014512|         0|
|276747|  60517794|         9|
|276747| 451192001|         0|
|276747| 609801279|         0|
|276747| 671537458|         9|
+------+----------+----------+
only showing top 20 rows
```

Figure :9

## D. Data Visualization:

After our data is cleaned, we are ready for visualizing our data to gain some valuable insights as well as present our project.

### 1. Book Dataset:

**Analysis no. 1 - Author with highest no. of books**

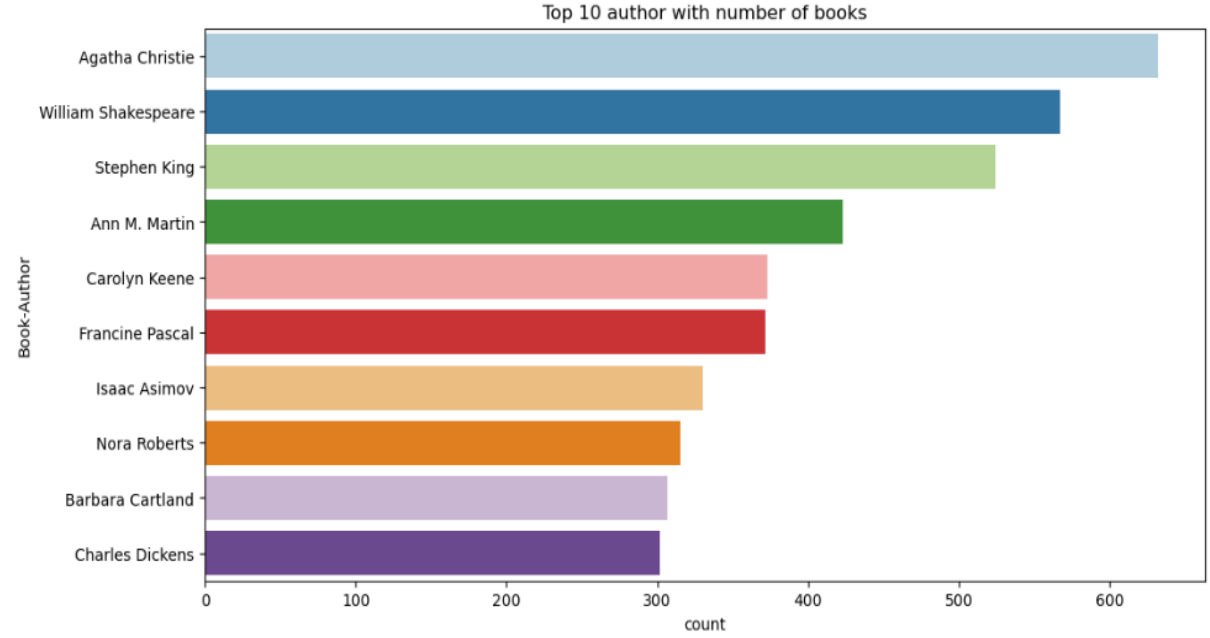



Figure :10

Agatha Christie is leading at top with more than 600 counts as she has the greatest number of books written as compared to others, followed by William Shakespeare.

## Analysis no. 2 - Top publishers

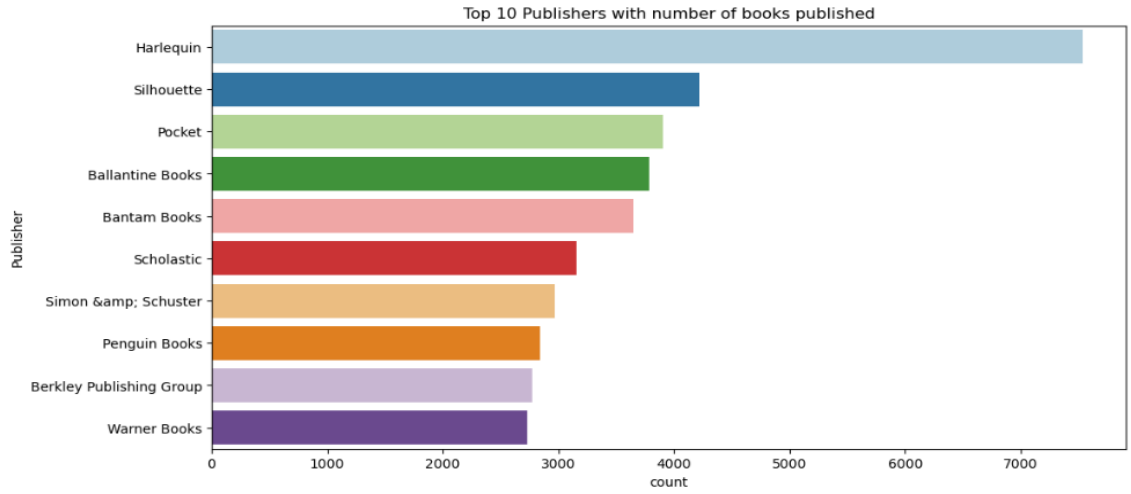



Figure :11

Harlequin has the greatest number of books published, followed by Silhouette.

## Analysis no.3 - Number of Books published on yearly basis

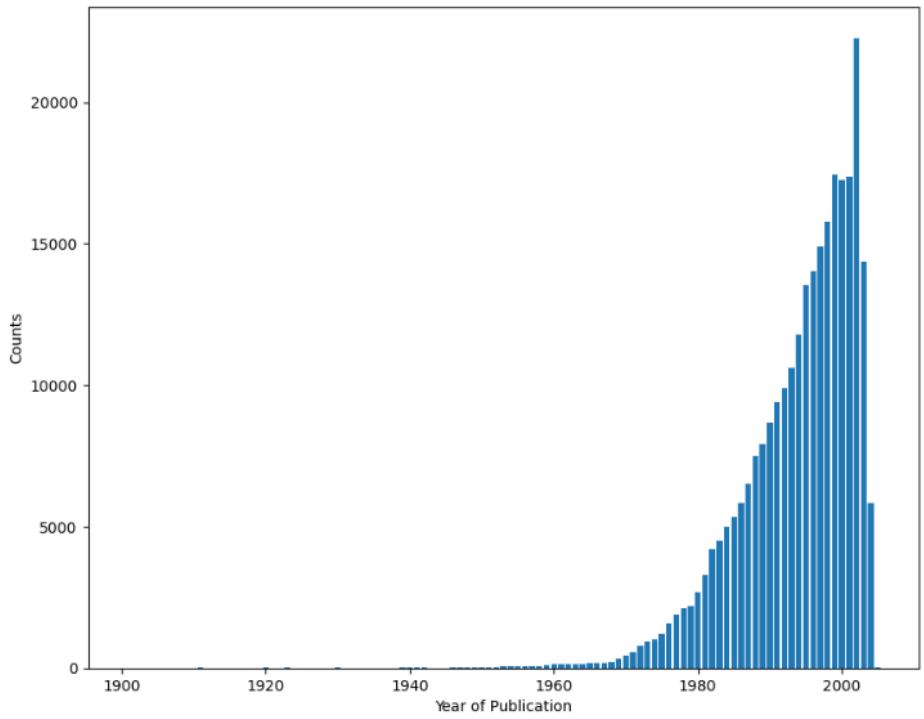



Figure :12

Most no. of books were published in the year 2002. The book industry boomed from 1980's and it was at its peak around the year 2000, it might because people started to understand the importance of books and gradually started to implement productive habits in their life.

## 2. Users Dataset:

### Analysis no.4 - Age wise distributions of users

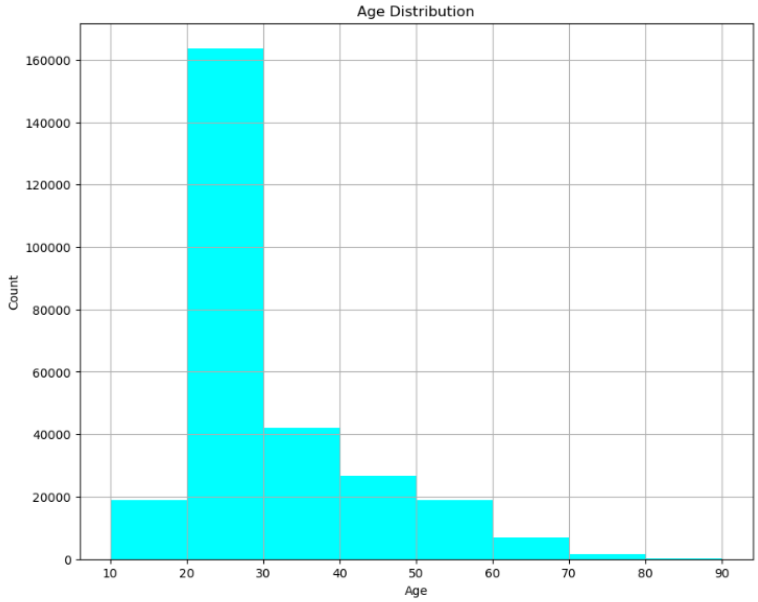



Figure :13

Most of the users are 20 - 30 years old which represents the youth.

## 3. Ratings Dataset:

### Analysis No. 5 - Rating distribution

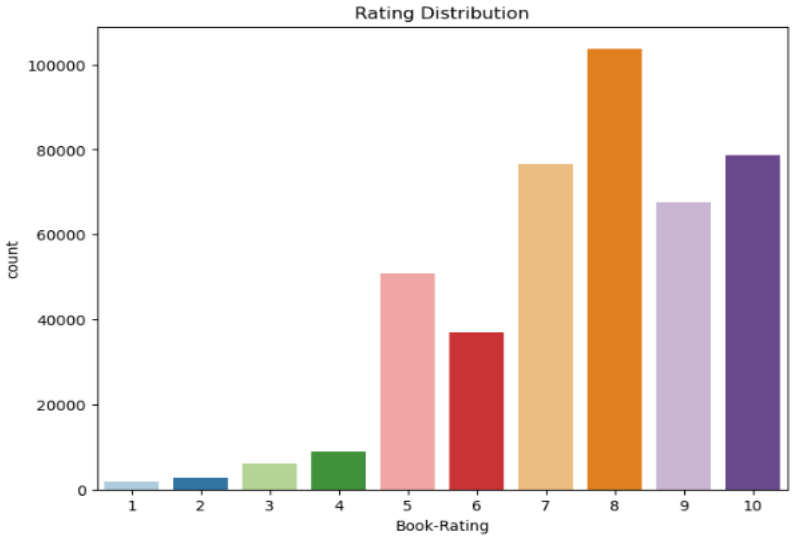



Figure :14

Above count plot indicates that higher ratings are more common amongst users and rating 8 has been rated highest number of times.

## Analysis No. 6 - Top 10 highest rated books

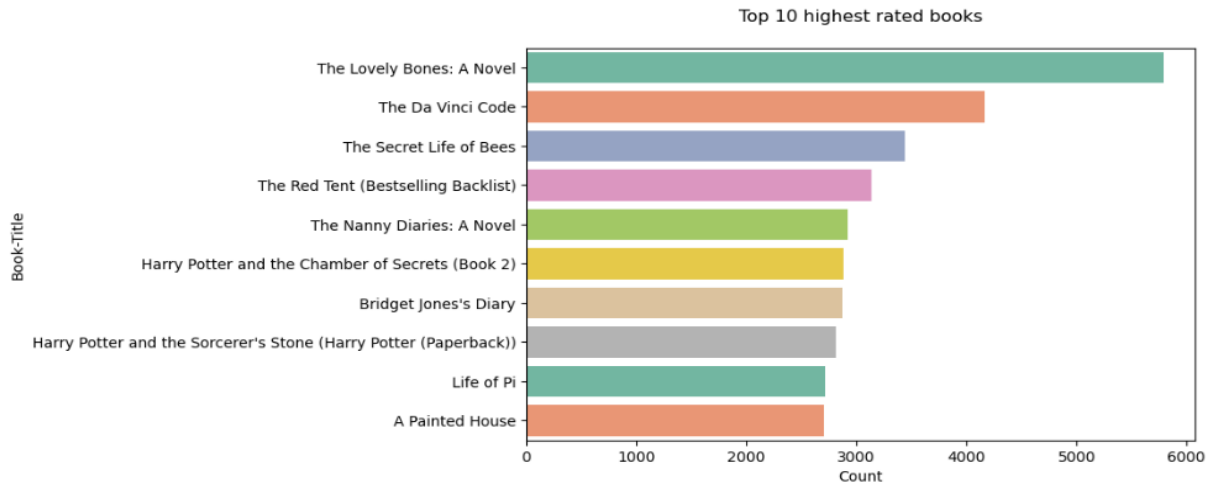



Figure :15

The book which has been rated by most number of users is 'The Lovely Bones'.

## Analysis No. 7 - Top 10 highest rated authors

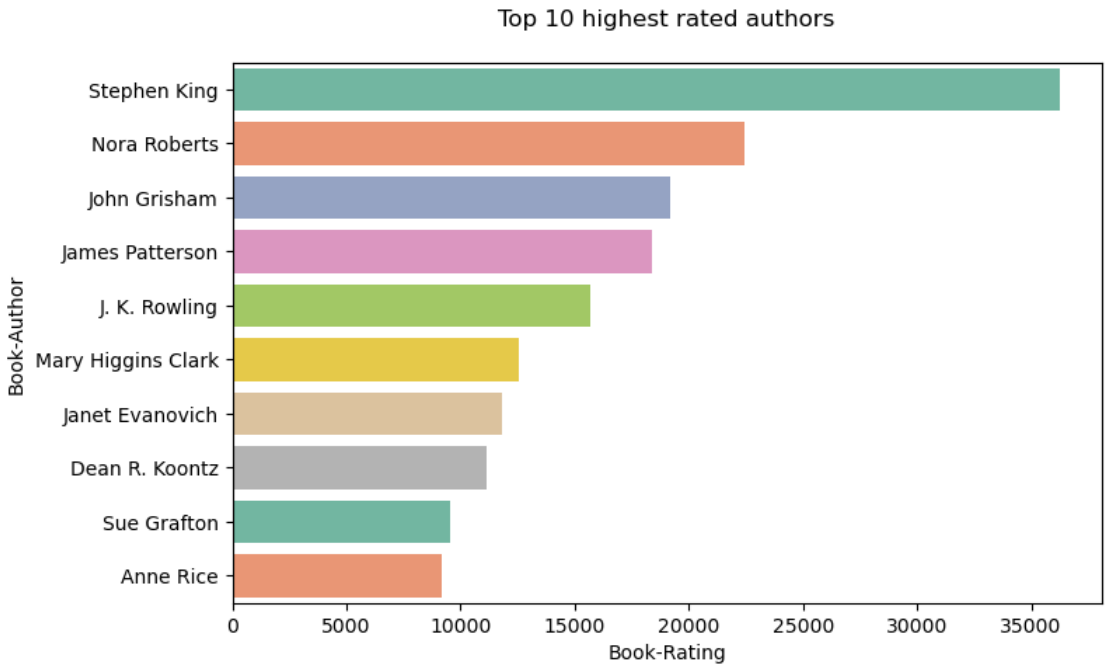



Figure :16

Top book author with respect to the number of ratings is Stephen King.

## E. Modelling/Approaches:

To build this recommendation system, we have taken three approaches as follows:

### 1) Popularity Based Recommendation System.

With its simplicity, this is the most basic recommendation system which offers generalized recommendation to every user based on their popularity. In a bookstore, if a certain book is popular among its customers & is also critically acclaimed, in the scenario that a new customer walk in & asks for the best, they would be suggested to try that book too. The same is true for movies, shows, music, etc. Whatever is more popular among the general public, is more likely to be recommended to new customers too.

This type of recommendation system makes generalized recommendation not personalized, meaning that this system will not take into account the personal preferences or choices, rather it would tell that this particular thing is liked by most of the users.

#### 1. Using average rating - Top 50 books in whole collection

The average rate of books is displayed below:

```
avg_rating_df = ratings_with_name.groupby('Book-Title').mean()['Book-Rating'].reset_index()
avg_rating_df.rename(columns={'Book-Rating':'avg_ratings'},inplace=True)
avg_rating_df
```

| | Book-Title | avg_ratings |
|---|---|---|
| 0 | A Light in the Storm: The Civil War Diary of Amelia Martin, Fenwick Island, Delaware, 1861 (Dear America) | 2.250000 |
| 1 | Always Have Popsicles | 0.000000 |
| 2 | Apple Magic (The Collector's series) | 0.000000 |
| 3 | Ask Lily (Young Women of Faith: Lily Series, Book 5) | 8.000000 |
| 4 | Beyond IBM: Leadership Marketing and Finance for the 1990s | 0.000000 |
| ... | ... | ... |
| 241066 | Ã?Â?lpiraten. | 0.000000 |
| 241067 | Ã?Â?rger mit Produkt X. Roman. | 5.250000 |
| 241068 | Ã?Â?sterlich leben. | 7.000000 |
| 241069 | Ã?Â?stlich der Berge. | 2.666667 |
| 241070 | Ã?Â?thique en toc | 4.000000 |

241071 rows × 2 columns

Figure :17

The top 50 recommended books are:

```
popular_df = popular_df[popular_df['num_ratings']>=250 ].sort_values('avg_ratings',ascending=False).head(50)
```

```
popular_df=popular_df.merge(books,on='Book-Title').drop_duplicates('Book-Title')[['Book-Title','Book-Author','Image-URL-M','num_r
popular_df.reset_index(inplace=True)
popular_df
```

| | index | Book-Title | Book-Author | Image-URL-M | num_ratings | avg_ratings |
|---|---|---|---|---|---|---|
| 0 | 0 | Harry Potter and the Prisoner of Azkaban (Book 3) | J. K. Rowling | http://images.amazon.com/images/P/0439136350.01.MZZZZZZZ.jpg | 428 | 5.852804 |
| 1 | 3 | Harry Potter and the Goblet of Fire (Book 4) | J. K. Rowling | http://images.amazon.com/images/P/0439139597.01.MZZZZZZZ.jpg | 387 | 5.824289 |
| 2 | 5 | Harry Potter and the Sorcerer's Stone (Book 1) | J. K. Rowling | http://images.amazon.com/images/P/0590353403.01.MZZZZZZZ.jpg | 278 | 5.737410 |
| 3 | 9 | Harry Potter and the Order of the Phoenix (Book 5) | J. K. Rowling | http://images.amazon.com/images/P/043935806X.01.MZZZZZZZ.jpg | 347 | 5.501441 |
| 4 | 13 | Harry Potter and the Chamber of Secrets (Book 2) | J. K. Rowling | http://images.amazon.com/images/P/0439064872.01.MZZZZZZZ.jpg | 556 | 5.183453 |
| 5 | 16 | The Hobbit : The Enchanting Prelude to The Lord of the Rings | J.R.R. TOLKIEN | http://images.amazon.com/images/P/0345339681.01.MZZZZZZZ.jpg | 281 | 5.007117 |
| 6 | 17 | The Fellowship of the Ring (The Lord of the Rings, Part 1) | J.R.R. TOLKIEN | http://images.amazon.com/images/P/0345339703.01.MZZZZZZZ.jpg | 368 | 4.948370 |
| 7 | 26 | Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback)) | J. K. Rowling | http://images.amazon.com/images/P/059035342X.01.MZZZZZZZ.jpg | 575 | 4.895652 |
| 8 | 28 | The Two Towers (The Lord of the Rings, Part 2) | J.R.R. TOLKIEN | http://images.amazon.com/images/P/0345339711.01.MZZZZZZZ.jpg | 260 | 4.880769 |
| 9 | 39 | To Kill a Mockingbird | Harper Lee | http://images.amazon.com/images/P/0446310786.01.MZZZZZZZ.jpg | 510 | 4.700000 |
| 10 | 47 | The Da Vinci Code | Dan Brown | http://images.amazon.com/images/P/0385504209.01.MZZZZZZZ.jpg | 898 | 4.642539 |

Figure :18

## 2. Books popular yearly :

Here, we have shown the books which are popular by yearly.

```
years = set()
indices = []
for ind, row in popular_df_y.iterrows():
    if row['Year-Of-Publication'] in years:
        indices.append(ind)
    else:
        years.add(row['Year-Of-Publication'])

popular_df_y = popular_df_y.drop(indices)
popular_df_y = popular_df_y.drop(['num_ratings','avg_ratings'], axis = 1)
popular_df_y = popular_df_y.sort_values('Year-Of-Publication')
```

```
popular_df_y
```

| | ISBN | Book-Title | Book-Author | Year-Of-Publication | Publisher | Image-URL-S | |
|---|---|---|---|---|---|---|---|
| 166739 | 0781228956 | Complete Works 10 Volumes [2,6,7,8,9] (Notable American Authors) | Benjamin Franklin | 1806 | Reprint Services Corp | http://images.amazon.com/images/P/0781228956.01.THUMBZZZ.jpg | http://images.amazon.com/images/I |
| 166743 | 0781268001 | Hugh Wynne, Free Quaker (2 Volumes (BCL1-PS American Literature) | Silas Weir Mitchell | 1897 | Reprint Services Corp | http://images.amazon.com/images/P/0781268001.01.THUMBZZZ.jpg | http://images.amazon.com/images/I |
| 217769 | 1551103982 | The Cycling Adventures of Coconut Head: A North American Odyssey | Ted Schredd | 1900 | Graphic Arts Center Pub Co | http://images.amazon.com/images/P/1551103982.01.THUMBZZZ.jpg | http://images.amazon.com/images/I |
| 140778 | 0671825356 | W D HSE PLANTS | Jd Hersey | 1901 | Simon & Schuster | http://images.amazon.com/images/P/0671825356.01.THUMBZZZ.jpg | http://images.amazon.com/images/I |
| 190035 | 0841499306 | Charlotte Bronte, George Eliot and Jane Austin: | Henry H. Bonnell | 1902 | Folcroft Library Editions | http://images.amazon.com/images/P/0841499306.01.THUMBZZZ.jpg | http://images.amazon.com/images/I |

Figure :19

## 2) Correlation Based Recommendation System

Correlation coefficients are used to measure how strong a relationship is between two variables. There are several types of correlation coefficient, but the most popular is Pearson's. A Pearson correlation is a number between -1 and +1 that indicates to which extent 2 variables are linearly related. So in this case, it is the rating for two books.

First, the average rating & the number of ratings each book received were found and then sorted based on the rating count in descending order.

```
average_rating = pd.DataFrame(ratings.groupby('ISBN')['Book-Rating'].mean())
average_rating['ratingCount'] = pd.DataFrame(ratings.groupby('ISBN')['Book-Rating'].count())
average_rating.sort_values('ratingCount', ascending=False).head()
```

| ISBN | Book-Rating | ratingCount |
|---|---|---|
| 0971880107 | 1.019584 | 2502 |
| 0316666343 | 4.468726 | 1295 |
| 0385504209 | 4.652322 | 883 |
| 0060928336 | 3.448087 | 732 |
| 0312195516 | 4.334716 | 723 |

Figure :20

**Observations:** The book with the most rating counts isn't necessarily a highly rated book. As seen from the table above, the book with the most rating counts of '2502' only had a rating of '1.019584'. As a result, if recommendations were made solely based on rating counts, it is evident that mistakes would be made.

The 'ratings' table is then converted into a 2D matrix. The matrix is sparse since not every user rated every book.

```
ratings_pivot = ratings_cor.pivot(index='User-ID', columns='ISBN')['Book-Rating']
userID = ratings_pivot.index
ISBN = ratings_pivot.columns
print(ratings_pivot.shape)
ratings_pivot.head()
```

```
(905, 207699)
```

| ISBN | 0330299891 | 0375404120 | 0586045007 | 9022906116 | 9032803328 | 9044922564 | 9044922572 | 9044922718 | 9044923161 | 904492401X | ... | UNGRANDHOMMED |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| User-ID | | | | | | | | | | | | |
| 254 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN |
| 2276 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN |
| 2766 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN |
| 2977 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN |
| 3363 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN |

5 rows × 207699 columns

Figure :21

To test the system, the book ' **The Lovely Bones: A Novel**' by American writer Alice Sebold.

The following are the books correlated with the above-mentioned book:

```
# Let's find out which books are correlated with the 2nd most rated book "The Lovely Bones: A Novel", '0316666343'.

bones_ratings = ratings_pivot['0316666343']
similar_to_bones = ratings_pivot.corrwith(bones_ratings)
corr_bones = pd.DataFrame(similar_to_bones, columns=['pearsonR'])
corr_bones.dropna(inplace=True)
corr_summary = corr_bones.join(average_rating['ratingCount'])
corr_summary[corr_summary['ratingCount']>=300].sort_values('pearsonR', ascending=False).head(10)
```

| ISBN | pearsonR | ratingCount |
|---|---|---|
| 0316666343 | 1.000000 | 1295 |
| 0312291639 | 0.471872 | 354 |
| 0316601950 | 0.434248 | 568 |
| 0446610038 | 0.429712 | 391 |
| 0446672211 | 0.421478 | 585 |
| 0385265700 | 0.351635 | 319 |
| 0345342968 | 0.316922 | 321 |
| 0060930535 | 0.309860 | 494 |
| 0375707972 | 0.308145 | 354 |
| 0684872153 | 0.272480 | 326 |

Figure :22

The recommended books are:

| | ISBN | Book-Title | Book-Author | Year-Of-Publication | Publisher | Image-URL-S | |
|---|---|---|---|---|---|---|---|
| 0 | 0312291639 | The Nanny Diaries: A Novel | Emma McLaughlin | 2003 | St. Martin's Griffin | http://images.amazon.com/images/P/0312291639.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/0 |
| 1 | 0316601950 | The Pilot's Wife : A Novel | Anita Shreve | 1999 | Back Bay Books | http://images.amazon.com/images/P/0316601950.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/0 |
| 2 | 0446610038 | 1st to Die: A Novel | James Patterson | 2002 | Warner Vision | http://images.amazon.com/images/P/0446610038.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/0 |
| 3 | 0446672211 | Where the Heart Is (Oprah's Book Club (Paperback)) | Billie Letts | 1998 | Warner Books | http://images.amazon.com/images/P/0446672211.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/0 |
| 4 | 0385265700 | The Book of Ruth (Oprah's Book Club (Paperback)) | Jane Hamilton | 1990 | Anchor | http://images.amazon.com/images/P/0385265700.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/0 |
| 5 | 0345342968 | Fahrenheit 451 | RAY BRADBURY | 1987 | Del Rey | http://images.amazon.com/images/P/0345342968.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/0 |
| 6 | 0060930535 | The Poisonwood Bible: A Novel | Barbara Kingsolver | 1999 | Perennial | http://images.amazon.com/images/P/0060930535.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/0 |
| 7 | 0375707972 | The Reader | Bernhard Schlink | 1999 | Vintage Books USA | http://images.amazon.com/images/P/0375707972.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/0 |
| 8 | 0684872153 | Angela's Ashes (MMP) : A Memoir | Frank McCourt | 1999 | Scribner | http://images.amazon.com/images/P/0684872153.01.THUMBZZZ.jpg | http://images.amazon.com/images/P/0 |

Figure :23

From this we can conclude that the all the books mentioned in Figure :19 does not show the similarity from the book ' **The Lovely Bones: A Novel'.** It's not showing the proper resulting recommended books are correlated with the book which we have tested, we can imply that our correlation-based book recommendation system is not working.

## 3) Collaborative Filtering Based Recommendation System:

In Collaborative Filtering, we tend to find similar users and recommend what similar users like. In this type of recommendation system, we don't use the features of the item to recommend it, rather we classify the users into the clusters of similar types, and recommend each user according to the preference of its cluster.

### 1. Cosine Similarity Based Approach:

Cosine similarity is a metric used to measure how similar two items are. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The output value ranges from 0–1. 0 means no similarity, whereas 1 means that both the items are 100% similar.

```
def recommend(book_name):
    # index fetch
    index = np.where(pt.index==book_name)[0][0]

    # enumerate displays index along with similarity
    similar_items = sorted(list(enumerate(similarity_scores[index])),key = lambda x:x[1],reverse=True)[1:6]

    for i in similar_items:
        print(pt.index[i[0]])
#     return suggestions
```

```
recommend("Harry Potter and the Sorcerer's Stone (Book 1)")
```

```
Harry Potter and the Chamber of Secrets (Book 2)
Harry Potter and the Prisoner of Azkaban (Book 3)
Harry Potter and the Goblet of Fire (Book 4)
Harry Potter and the Order of the Phoenix (Book 5)
The Mists of Avalon
```

Figure :24

Here, in this code, we have recommended the book ‘Harry Potter and the Sorcerer’s Stone (Book 1)’ and according to this book, we are getting the similar outputs which shows the cosine similarity among the books.

## 2. Matrix Factorization:

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices.

Singular value decomposition (SVD) is used here. The users' rating table is converted into a utility matrix & the missing values are filled with zeros,

```
user_rating_pivot1 = user_rating.pivot(index = 'User-ID', columns = 'Book-Title', values = 'Book-Rating').fillna(0)
user_rating_pivot1.head()
```

| Book-Title | 10 Lb. Penalty | 16 Lighthouse Road | 1984 | 1st to Die: A Novel | 2010: Odyssey Two | 204 Rosewood Lane | 2061: Odyssey Three | 24 Hours | 2nd Chance | 3rd Degree | ... | YOU BELONG TO ME | Year of Wonders | You Belong To Me | You Shall Know Our Velocity | Young Wives | Zen and t Art Motorcy Maintenan An Inqu into Valu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| User-ID | | | | | | | | | | | | | | | | | |
| 8 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | ... | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.0 |
| 9 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | ... | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.0 |
| 14 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | ... | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.0 |
| 16 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | ... | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.0 |
| 17 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | ... | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.0 |

Figure :25

This utility matrix is transposed in order for the 'book title' & 'userIDs' to become rows & columns respectively. After using TruncatedSVD to decompose it, it is fitted into the model for further dimensionality reduction.

Pearson's R correlation coefficient is calculated for every book pair in the final matrix. :-

```
user_rating_pivot1.shape
```

```
(47994, 2444)
```

```
X = user_rating_pivot1.values.T
X.shape
```

```
(2444, 47994)
```

```
import sklearn
from sklearn.decomposition import TruncatedSVD

SVD = TruncatedSVD(n_components=12, random_state=17)
matrix = SVD.fit_transform(X)
matrix.shape
```

```
(2444, 12)
```

```
import warnings
warnings.filterwarnings("ignore",category =RuntimeWarning)
corr = np.corrcoef(matrix)
corr.shape
```

```
(2444, 2444)
```

Figure :26

```
book_title = user_rating_pivot1.columns
book_list = list(book_title)
coffey_hands = book_list.index("The Green Mile: Coffey's Hands (Green Mile Series)")
print(coffey_hands)
```

```
1908
```

```
corr_coffey_hands  = corr[coffey_hands]
```

```
book_title[corr_coffey_hands>0.94]
```

```
Index(['Hearts In Atlantis : New Fiction',
       'The Green Mile: Coffey's Hands (Green Mile Series)',
       'The Green Mile: Night Journey (Green Mile Series)', 'The Shining',
       'The Two Dead Girls (Green Mile Series)'],
      dtype='object', name='Book-Title')
```

Figure :27

Clearly, as seen from the Figure :16 codes, the recommendation system using matrix factorization also recommended the same books as the system using cosine similarity-based approach.

### 3. K-Nearest Neighbours (k-NN) :

K-Nearest Neighbours (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. KNN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.

Using this algorithm, clusters of similar users based on common book ratings can be found and predictions can be made using the average rating of the top-k nearest neighbours.

The table is converted into a 2D matrix & the missing values are filled with zeroes since the distances between rating vectors will be calculated. The values of the matrix data frame are then transformed into a scipy sparse matrix for more efficiency calculations

The algorithm used to compute the nearest neighbours is 'brute' & the metric is 'cosine' so that the algorithm will calculate the cosine similarity between rating vectors.

```
from scipy.sparse import csr_matrix


user_rating_pivot2 = user_rating.pivot(index = 'Book-Title', columns = 'User-ID', values = 'Book-Rating
user_rating_matrix = csr_matrix(user_rating_pivot2.values)

from sklearn.neighbors import NearestNeighbors

model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model_knn.fit(user_rating_matrix)
```

```
NearestNeighbors(algorithm='brute', metric='cosine')
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
# Test our model and make some recommendations:
```

```
user_rating_pivot2.shape
```

```
(2444, 47994)
```

Figure :28

```
np.where(user_rating_pivot2.index=="The Green Mile: Coffey's Hands (Green Mile Series)")[0][0]
```

```
1908
```

```
def book_recommend(bk_name):
    # index fetch
    book_id = np.where(user_rating_pivot2.index==bk_name)[0][0]
    distance,suggestion =  model_knn.kneighbors(user_rating_pivot2.iloc[book_id,:].values.reshape(1,-1)
    for i in range(len(suggestion)):
        books =user_rating_pivot2.index[suggestion[i]]
        count=0
        for j in books:
            count+=1
            if count==1:
                continue
            else:
                print(j)
```

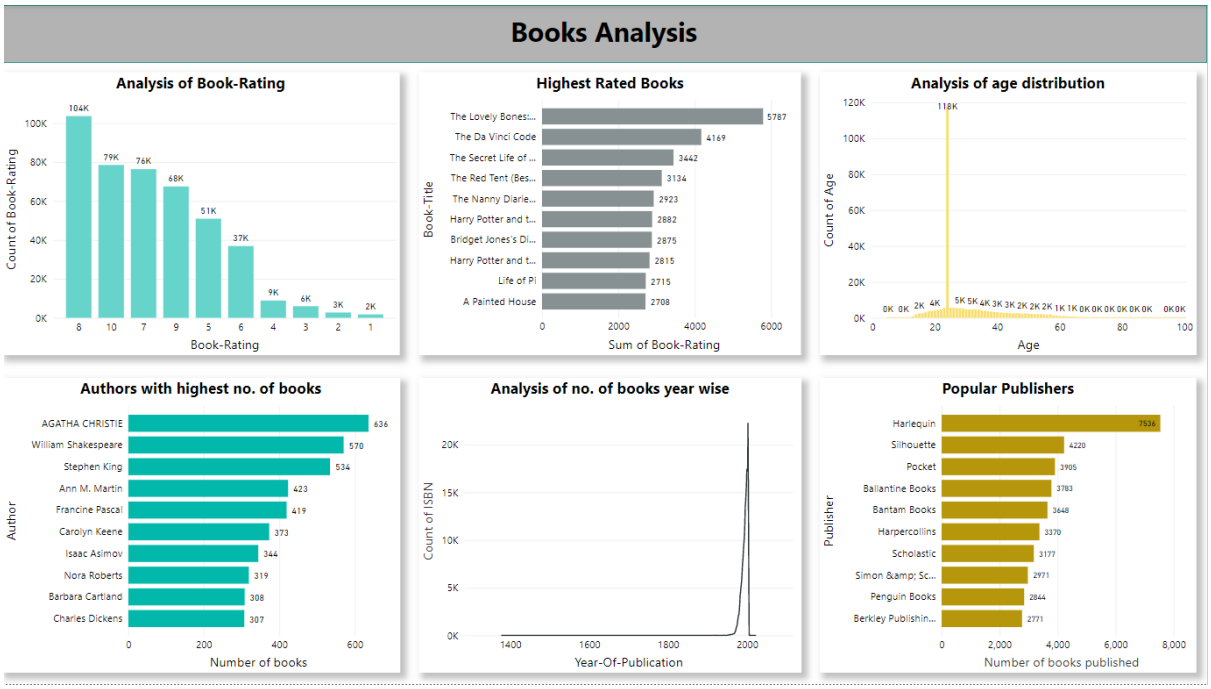
```
book_recommend("The Green Mile: Coffey's Hands (Green Mile Series)")
```

```
The Green Mile: Night Journey (Green Mile Series)
The Green Mile: The Mouse on the Mile (Green Mile Series)
The Green Mile: The Bad Death of Eduard Delacroix (Green Mile Series)
The Two Dead Girls (Green Mile Series)
The Green Mile: Coffey on the Mile (Green Mile Series)
```

Figure :29

Clearly, as seen from the Figure :18 codes, the recommendation system using k-Nearest Neighbour (k-NN) algorithm also recommended the same books as the system using cosine similarity-based approach and matrix factorization method.

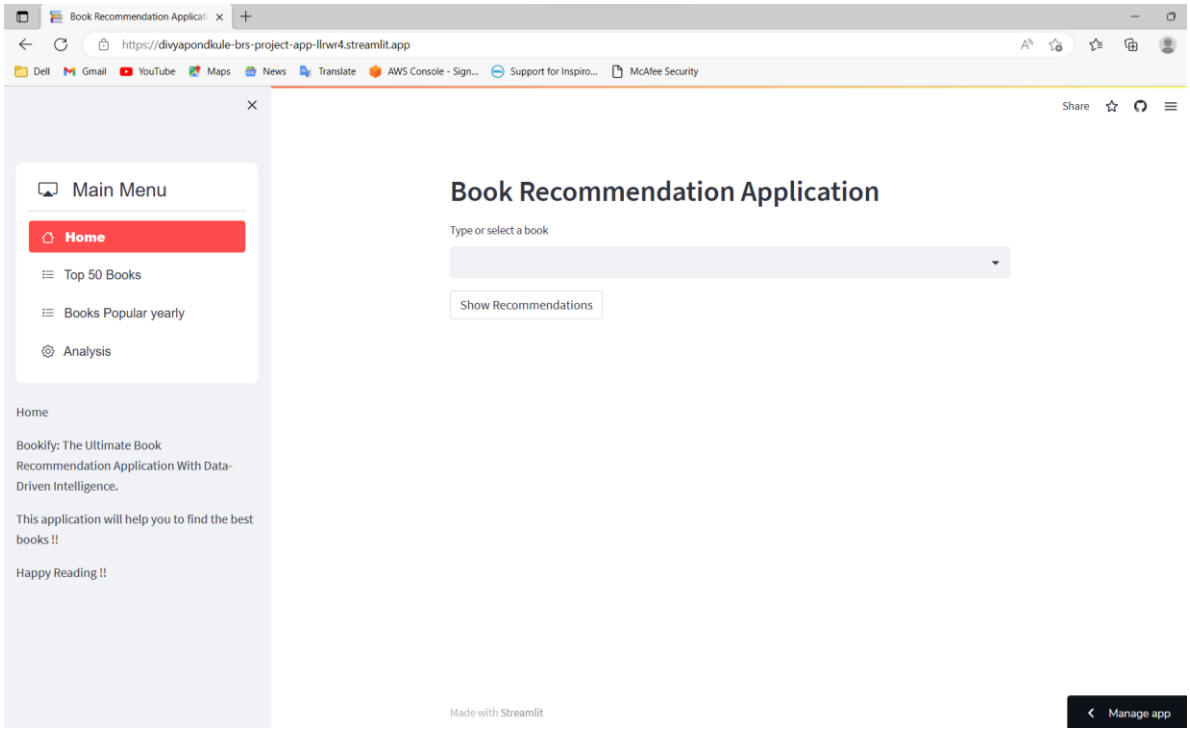
## F. Dashboard: We have created a dashboard using Power BI

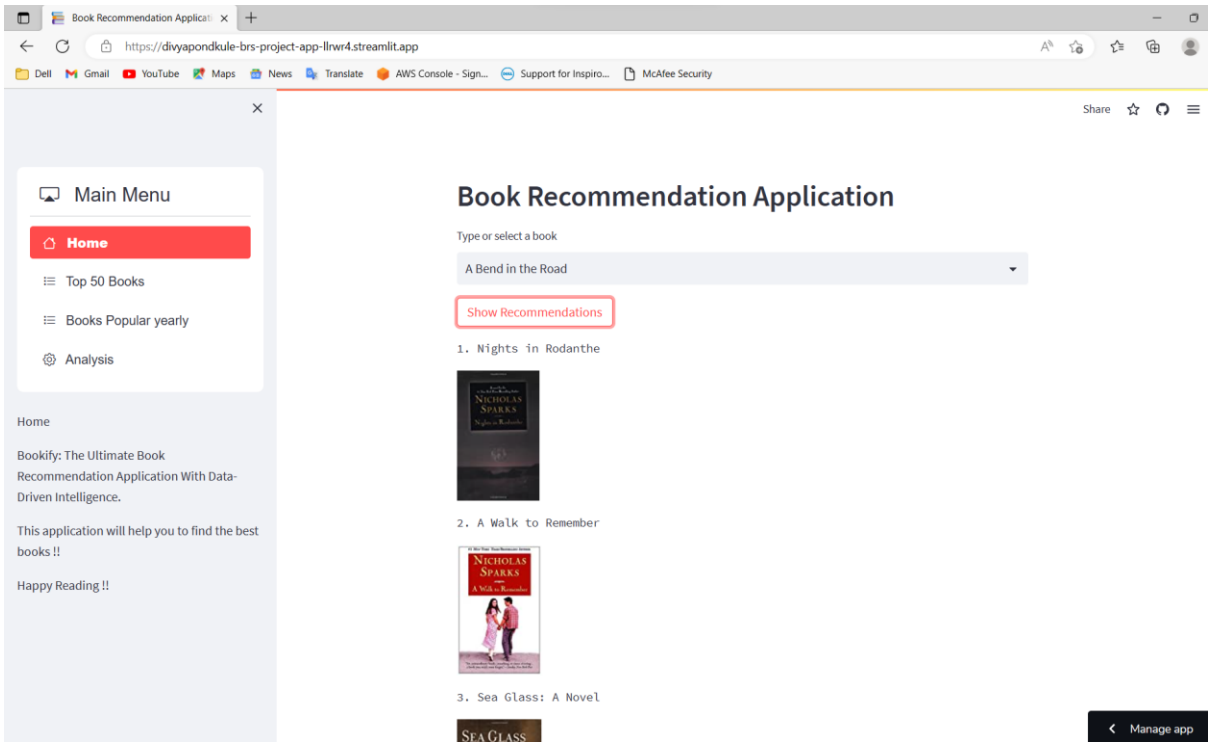


## G. Deployment:

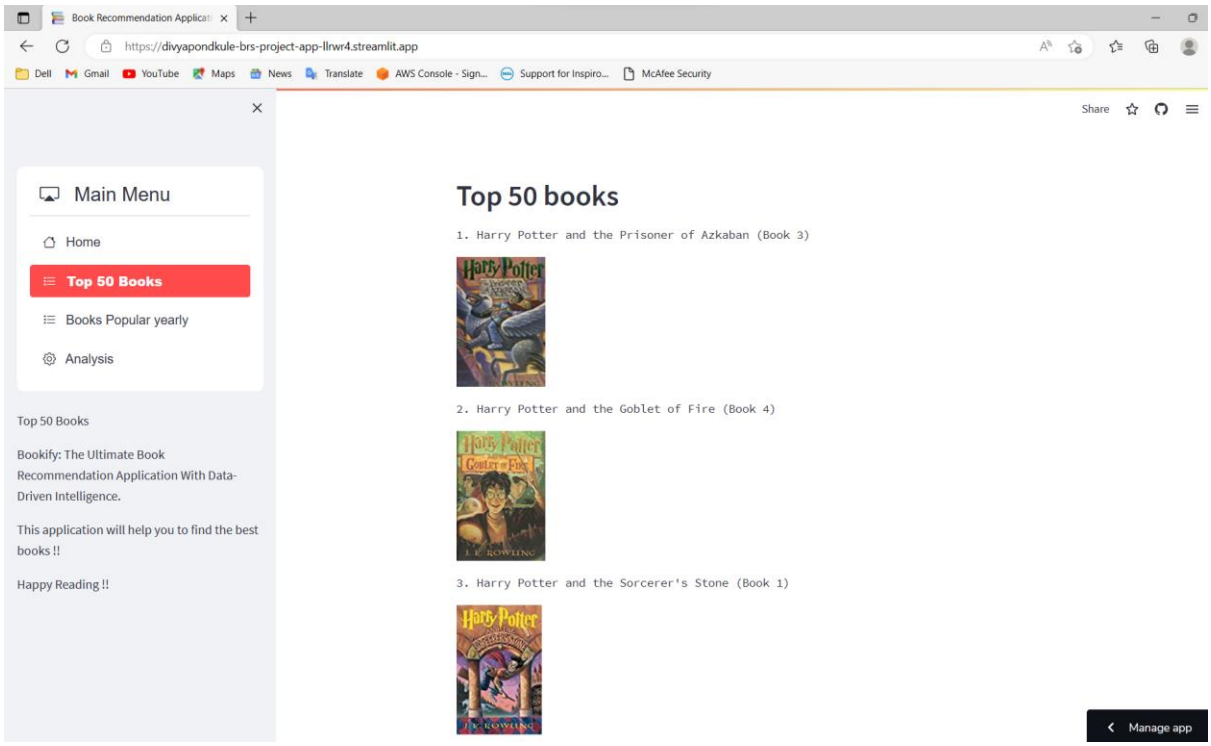
We have used streamlit framework of python to create the frontend and streamlit app to deploy the project.

**1. Home page:** Here, we recommend books to the users.

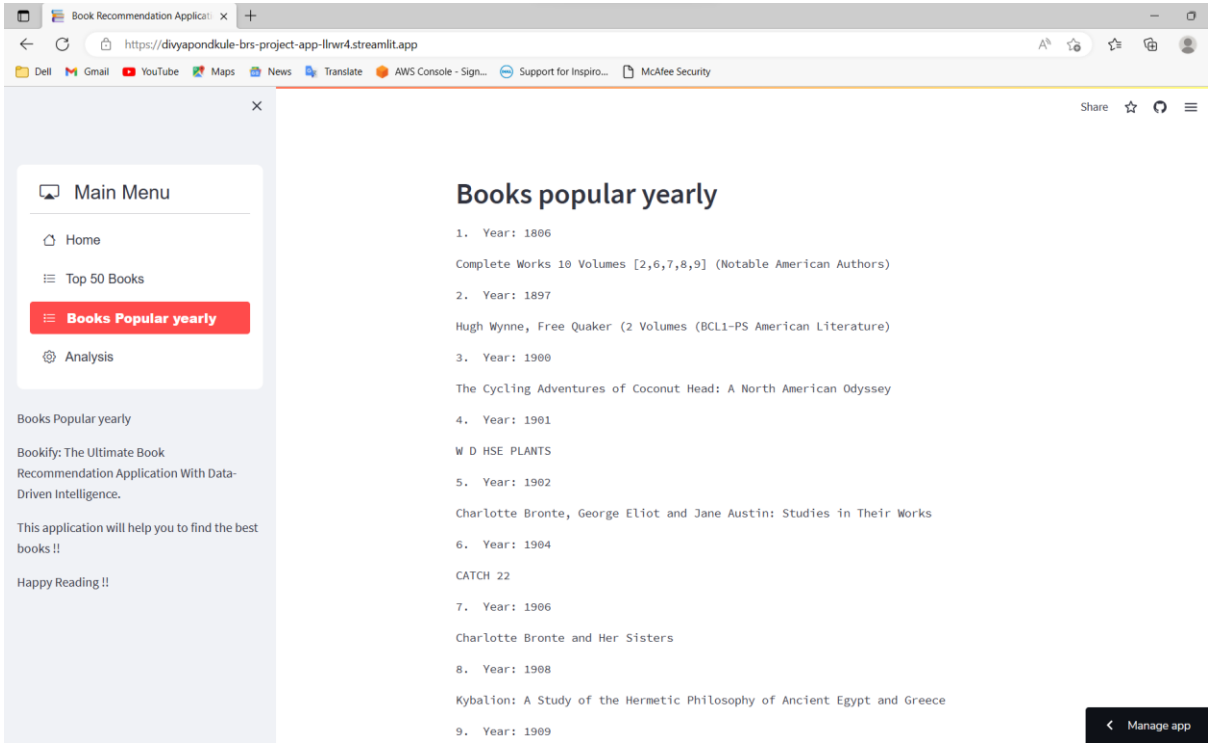


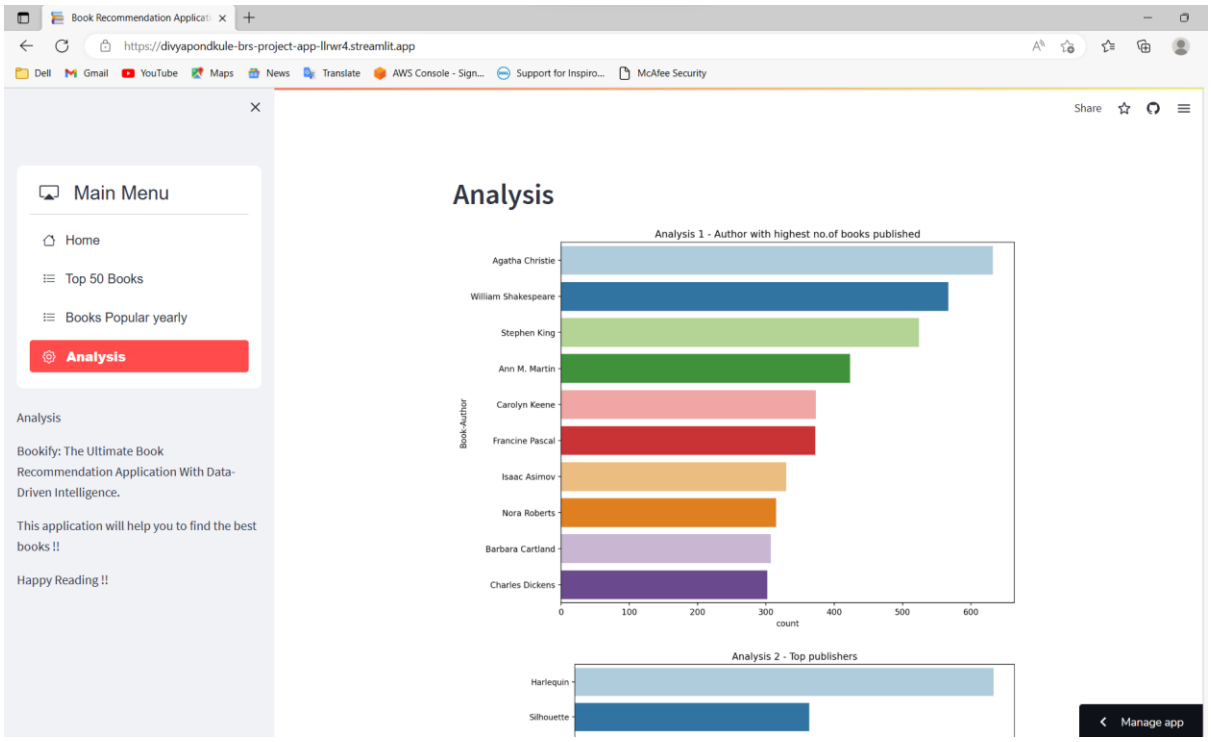
**2. Top 50 Books page:** It display the most popular 50 books in entire collection




**3. Books popular yearly page:** It displays the popular books year-wise.



**4. Analysis page:** It displays various graphs for analysis purpose.



# 7. Future Scope

- Integrate with different platforms like book selling websites etc.
- To import and process live data.
- Provide various options:

  a) To preview the books

  b) To get the purchase link

  c) To see reviews

  d) To provide references like kindle version or audio books etc

# 8. Conclusion

In conclusion, the Bookify: The Ultimate Book Recommendation Application with Data-Driven Intelligence project aimed to develop a collaborative-based and popularity-based recommendation system that suggests books to users based on their reading history and preferences. The project used a dataset of books, ratings, and users, which was pre-processed and transformed using data cleaning, encoding, and feature extraction techniques.

The collaborative-based recommendation system used user-based and item-based collaborative filtering techniques to generate recommendations, while the popularity-based recommendation system used book popularity and rating averages to suggest popular books.

In conclusion, the Bookify project demonstrates the importance of using collaborative-based and popularity-based recommendation systems to provide more accurate and personalized recommendations to users. The project contributes to the field of recommender systems and provides insights into the design and evaluation of book recommendation systems. The limitations and future work of the project include improving the scalability and robustness of the system and incorporating more advanced techniques such as deep learning and reinforcement learning for recommendation.