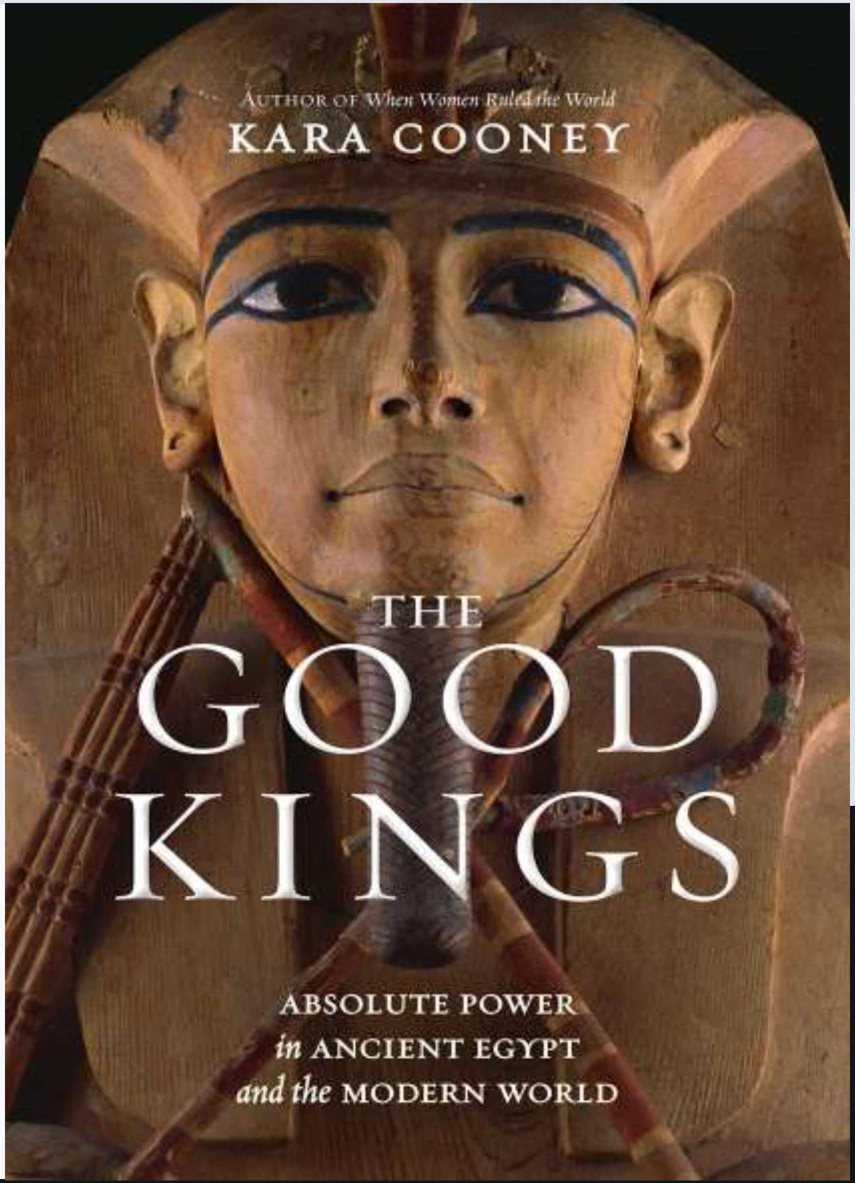
BOOK RECOMMENDER


Author of When Women Ruled the World
KARA COONEY
THE GOOD KINGS
ABSOLUTE POWER in ANCIENT EGYPT and the MODERN WORLD

INTRODUCTION

The project is primarily focused on building a **book recommendation system**, which typically falls under the domain of **collaborative filtering**. Collaborative filtering is an unsupervised learning approach used for recommendation tasks. It's neither regression nor classification, as the goal is to predict user preferences or item similarities rather than a specific numeric value or category.

The **k-nearest neighbors (KNN) algorithm** is commonly used for collaborative filtering in recommendation systems. In this case, it's used to find similar books based on user preferences or book characteristics. KNN is a type of unsupervised learning algorithm that belongs to the broader category of instance-based learning.

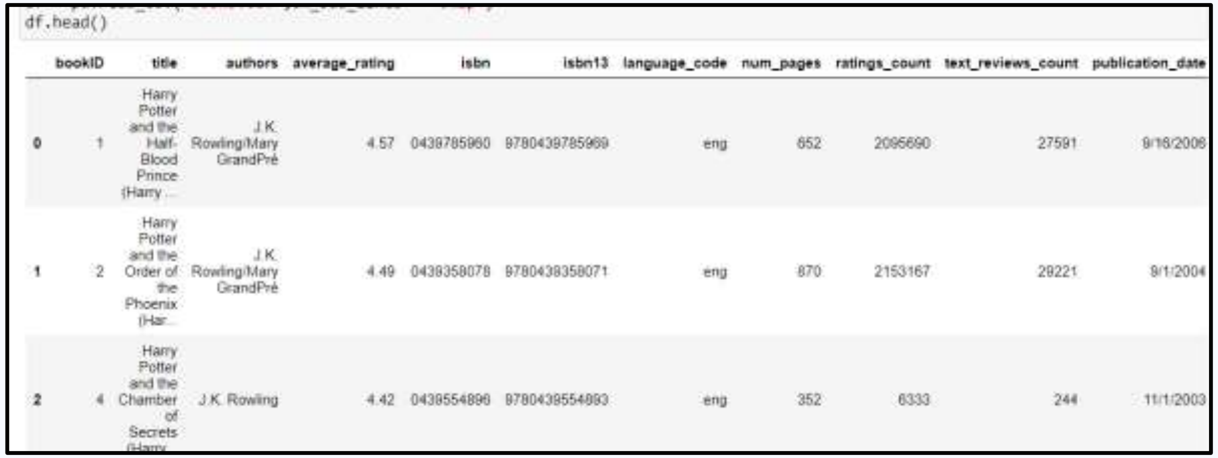
In the context of a book recommendation system, the primary learning task is generally considered unsupervised, as the goal is to find **patterns or similarities in user behavior or item features without explicit labels**.

THE DATASET

The dataset was obtained from Kaggle: https://www.kaggle.com/datasets/jealousleopard/goodreadsbooks?resource=download

- 12 columns
- 11126 unique entries

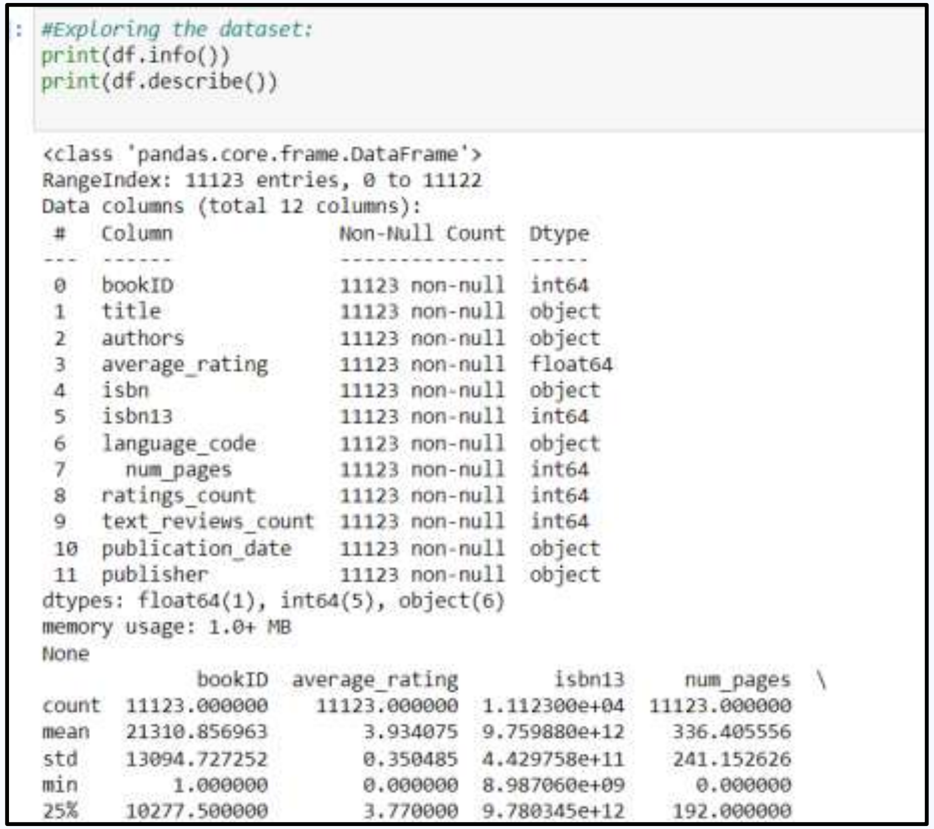
EXPLORING DATA



```
df.head()
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	publication_date
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591	9/16/2006
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221	9/1/2004
2	4	Harry Potter and the Chamber of Secrets (Harry	J.K. Rowling	4.42	0439554896	9780439554893	eng	352	6333	244	11/1/2003

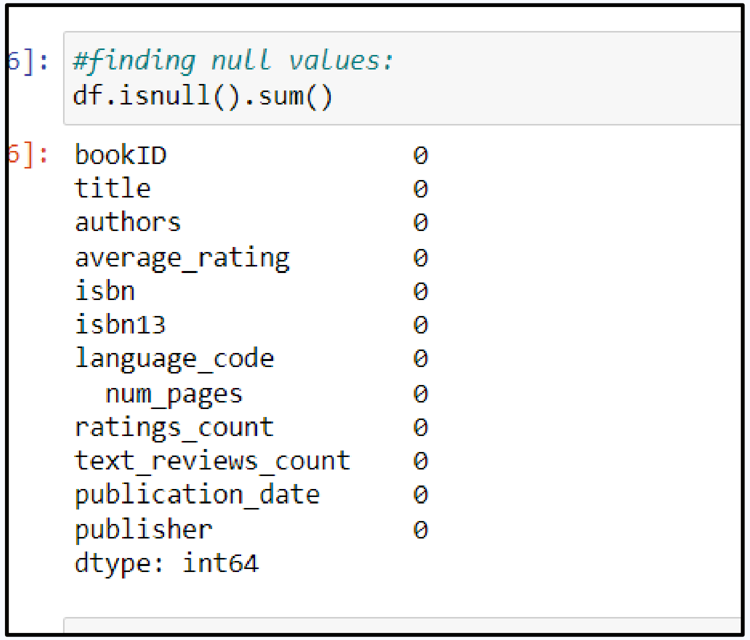
EXPLORING DATA



```python
#Exploring the dataset:
print(df.info())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11123 entries, 0 to 11122
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   bookID              11123 non-null  int64  
 1   title               11123 non-null  object 
 2   authors             11123 non-null  object 
 3   average_rating      11123 non-null  float64
 4   isbn                11123 non-null  object 
 5   isbn13              11123 non-null  int64  
 6   language_code       11123 non-null  object 
 7     num_pages         11123 non-null  int64  
 8   ratings_count       11123 non-null  int64  
 9   text_reviews_count  11123 non-null  int64  
 10  publication_date    11123 non-null  object 
 11  publisher           11123 non-null  object 
dtypes: float64(1), int64(5), object(6)
memory usage: 1.0+ MB
None
             bookID  average_rating        isbn13     num_pages  \
count  11123.000000    11123.000000  1.112300e+04  11123.000000   
mean   21310.856963        3.934075  9.759880e+12    336.405556   
std    13094.727252        0.350485  4.429758e+11    241.152626   
min        1.000000        0.000000  8.987060e+09      0.000000   
25%    10277.500000        3.770000  9.780345e+12    192.000000   
```

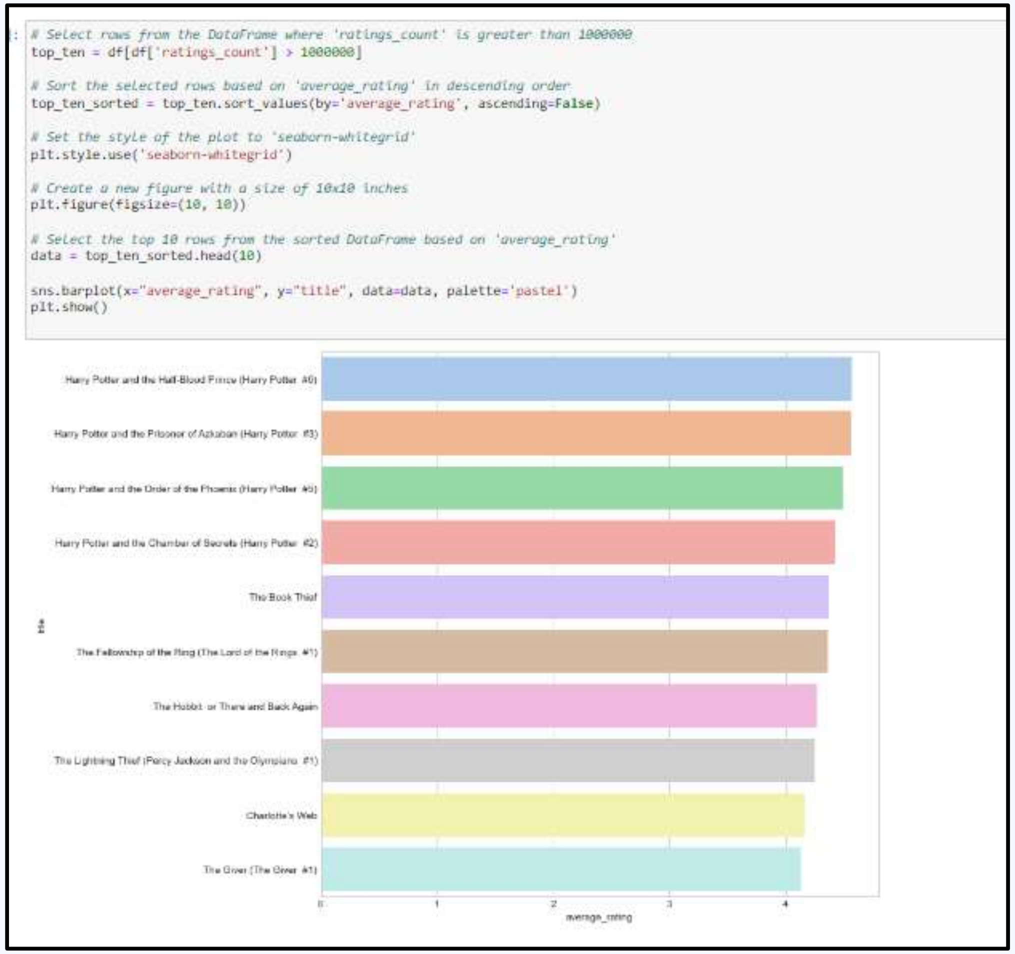
NULL VALUES



```
6]:  #finding null values:
     df.isnull().sum()
```

```
6]:  bookID                0
     title                 0
     authors               0
     average_rating        0
     isbn                  0
     isbn13                0
     language_code         0
       num_pages           0
     ratings_count         0
     text_reviews_count    0
     publication_date      0
     publisher             0
     dtype: int64
```

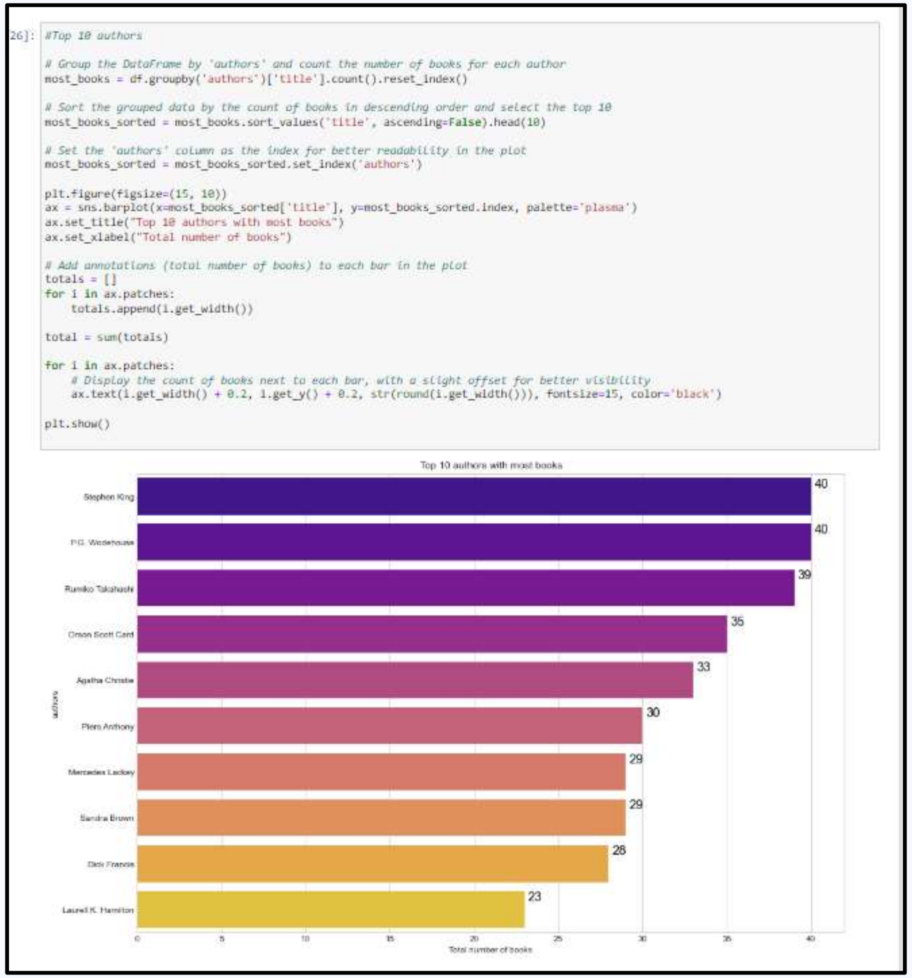
DATA VISUALIZATION



```
# Select rows from the DataFrame where 'ratings_count' is greater than 1000000
top_ten = df[df['ratings_count'] > 1000000]

# Sort the selected rows based on 'average_rating' in descending order
top_ten_sorted = top_ten.sort_values(by='average_rating', ascending=False)

# Set the style of the plot to 'seaborn-whitegrid'
plt.style.use('seaborn-whitegrid')

# Create a new figure with a size of 10x10 inches
plt.figure(figsize=(10, 10))

# Select the top 10 rows from the sorted DataFrame based on 'average_rating'
data = top_ten_sorted.head(10)

sns.barplot(x="average_rating", y="title", data=data, palette='pastel')
plt.show()
```

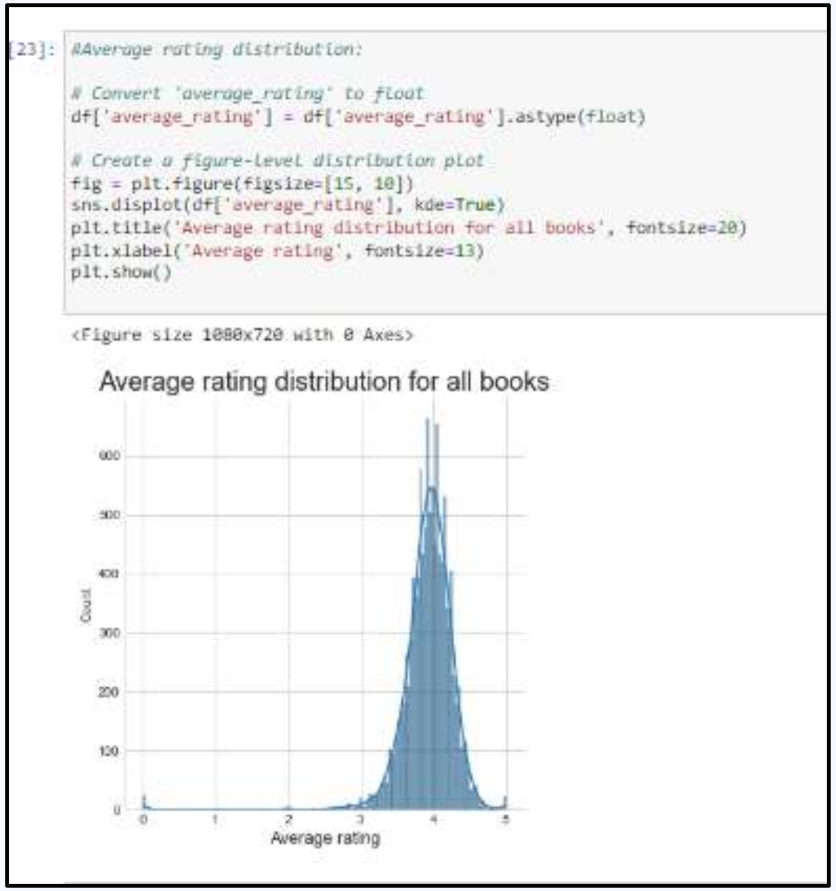
Harry Potter and the Half-Blood Prince (Harry Potter #6)
Harry Potter and the Prisoner of Azkaban (Harry Potter #3)
Harry Potter and the Order of the Phoenix (Harry Potter #5)
Harry Potter and the Chamber of Secrets (Harry Potter #2)
The Book Thief
The Fellowship of the Ring (The Lord of the Rings #1)
The Hobbit or There and Back Again
The Lightning Thief (Percy Jackson and the Olympians #1)
Charlotte's Web
The Giver (The Giver #1)
title
0
1
2
3
4
average_rating

DATA VISUALIZATION



```
26]: #Top 10 authors

     # Group the DataFrame by 'authors' and count the number of books for each author
     most_books = df.groupby('authors')['title'].count().reset_index()

     # Sort the grouped data by the count of books in descending order and select the top 10
     most_books_sorted = most_books.sort_values('title', ascending=False).head(10)

     # Set the 'authors' column as the index for better readability in the plot
     most_books_sorted = most_books_sorted.set_index('authors')

     plt.figure(figsize=(15, 10))
     ax = sns.barplot(x=most_books_sorted['title'], y=most_books_sorted.index, palette='plasma')
     ax.set_title("Top 10 authors with most books")
     ax.set_xlabel("Total number of books")

     # Add annotations (total number of books) to each bar in the plot
     totals = []
     for i in ax.patches:
         totals.append(i.get_width())

     total = sum(totals)

     for i in ax.patches:
         # Display the count of books next to each bar, with a slight offset for better visibility
         ax.text(i.get_width() + 0.2, i.get_y() + 0.2, str(round(i.get_width())), fontsize=15, color='black')

     plt.show()
```

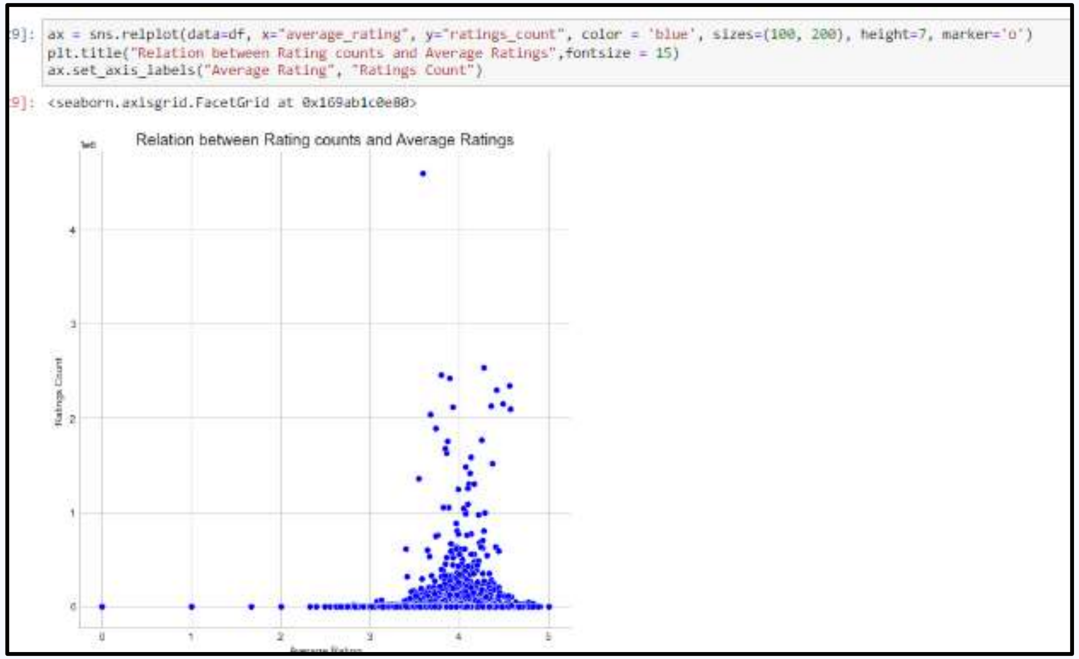
DATA VISUALIZATION



```
[23]: #Average rating distribution:

      # Convert 'average_rating' to float
      df['average_rating'] = df['average_rating'].astype(float)

      # Create a figure-level distribution plot
      fig = plt.figure(figsize=[15, 10])
      sns.displot(df['average_rating'], kde=True)
      plt.title('Average rating distribution for all books', fontsize=20)
      plt.xlabel('Average rating', fontsize=13)
      plt.show()
```

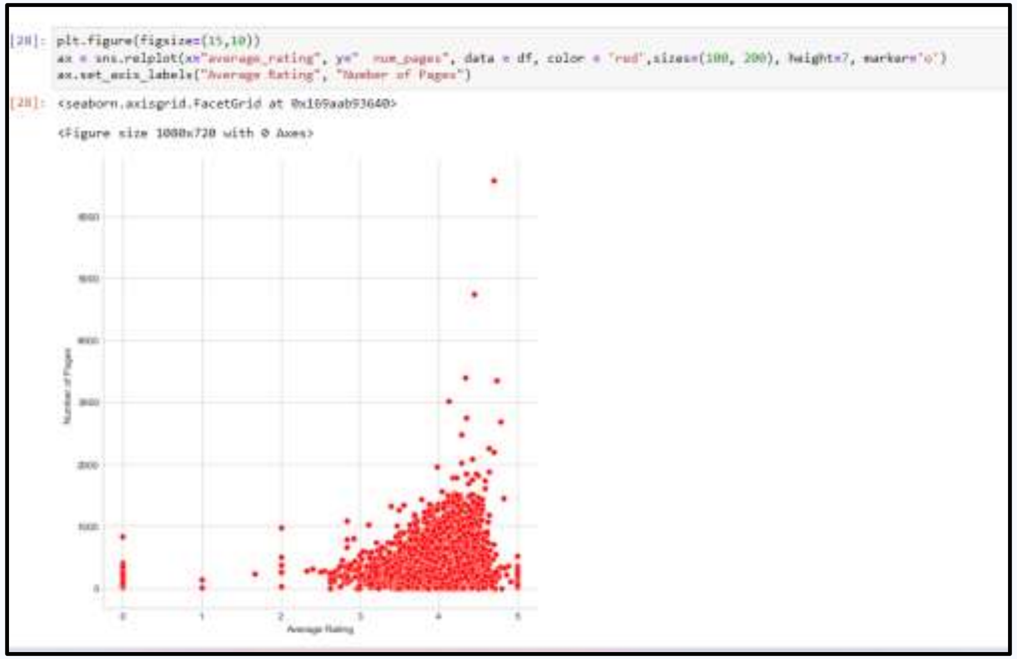
```
<Figure size 1080x720 with 0 Axes>
```

DATA VISUALIZATION



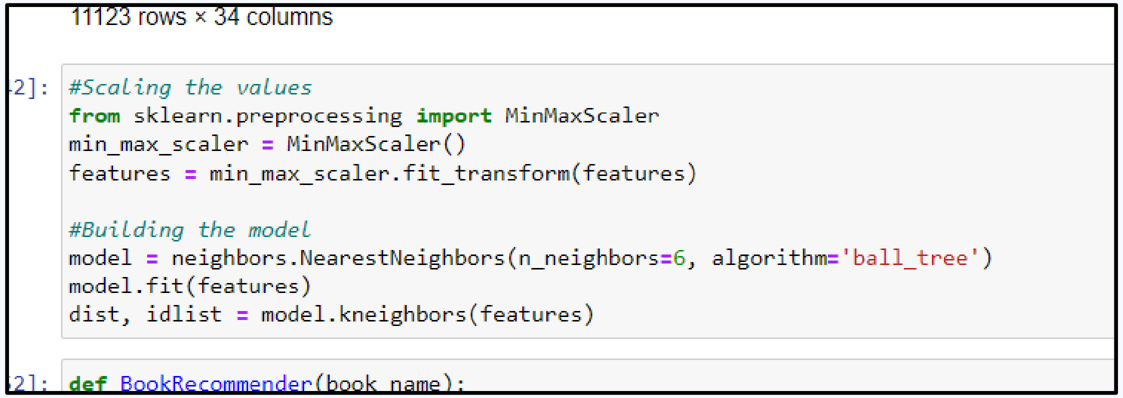
```
9]: ax = sns.relplot(data=df, x="average_rating", y="ratings_count", color = 'blue', sizes=(100, 200), height=7, marker='o')
    plt.title("Relation between Rating counts and Average Ratings",fontsize = 15)
    ax.set_axis_labels("Average Rating", "Ratings Count")

9]: <seaborn.axisgrid.FacetGrid at 0x169ab1c0e80>
```

DATA VISUALIZATION



```
[28]: plt.figure(figsize=(15,10))
      ax = sns.relplot(x="average_rating", y="  num_pages", data = df, color = 'red',sizes=(100, 200), height=7, marker='o')
      ax.set_axis_labels("Average Rating", "Number of Pages")

[28]: <seaborn.axisgrid.FacetGrid at 0x169aab93640>

      <Figure size 1080x720 with 0 Axes>
```

THE MODEL

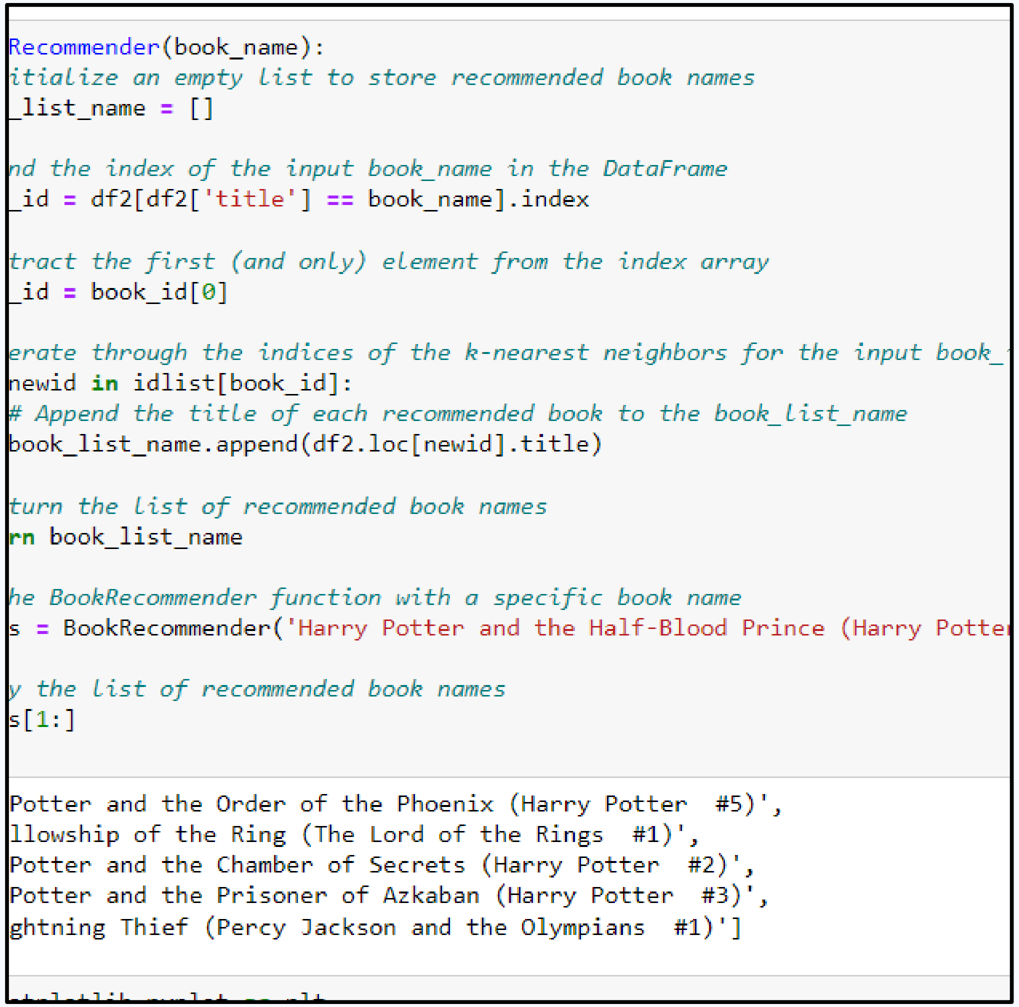


11123 rows × 34 columns

```
#Scaling the values
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler()
features = min_max_scaler.fit_transform(features)

#Building the model
model = neighbors.NearestNeighbors(n_neighbors=6, algorithm='ball_tree')
model.fit(features)
dist, idlist = model.kneighbors(features)
```

```
def BookRecommender(book_name):
```

EXAMPLE OUTPUT

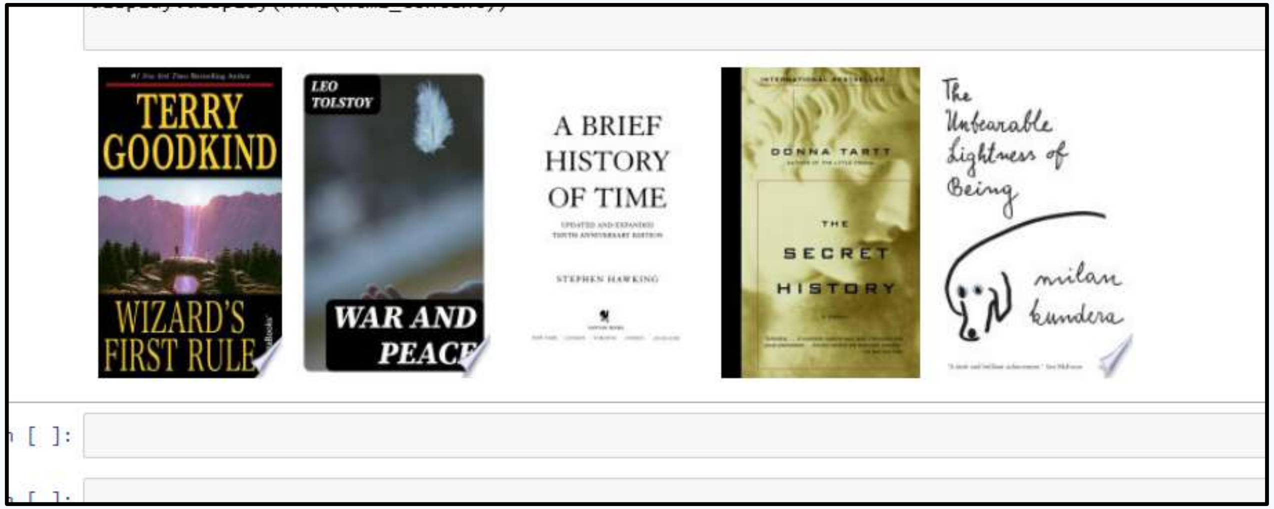


```
Recommender(book_name):
itialize an empty list to store recommended book names
_list_name = []

nd the index of the input book_name in the DataFrame
_id = df2[df2['title'] == book_name].index

tract the first (and only) element from the index array
_id = book_id[0]

erate through the indices of the k-nearest neighbors for the input book_
newid in idlist[book_id]:
# Append the title of each recommended book to the book_list_name
book_list_name.append(df2.loc[newid].title)

turn the list of recommended book names
rn book_list_name

he BookRecommender function with a specific book name
s = BookRecommender('Harry Potter and the Half-Blood Prince (Harry Potte

y the list of recommended book names
s[1:]
```

```
Potter and the Order of the Phoenix (Harry Potter  #5)',
llowship of the Ring (The Lord of the Rings  #1)',
Potter and the Chamber of Secrets (Harry Potter  #2)',
Potter and the Prisoner of Azkaban (Harry Potter  #3)',
ghtning Thief (Percy Jackson and the Olympians  #1)']
```

```
atplotlib.pyplot as plt
```

OUTPUT WITH BOOK COVER


TERRY GOODKIND
WIZARD'S FIRST RULE
LEO TOLSTOY
WAR AND PEACE
A BRIEF HISTORY OF TIME
STEPHEN HAWKING
DONNA TARTT
THE SECRET HISTORY
The Unbearable Lightness of Being
milan kundera
[ ]:
[ ]: