**BOOKVISTA: MACHINE LEARNING -INFUSED BOOK RECOMMENDATION SYSTEM**

**A PROJECT REPORT SUBMITTED TO SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR**

**THE AWARD OF THE DEGREE OF MASTER OF APPLIED DATA SCIENCE**

**BY**

**VIJAYARAM BHARATHI V RA2232014010076**

**MADHAVAN P K RA2232014010082**

**SAI ESHWAR D RA2232014010085**

**ELANGO G RA2232014010110**

**UNDER THE GUIDANCE OF**

**Dr. R. KIRUTHIGA, MCA, M.Phil, Ph.D**

****

**DEPARTMENT OF COMPUTER APPLICATION**

**FACULTY OF SCIENCE AND HUMANITIES**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR-603203**

**CHENGALPATTU, TAMIL NADU**

**NOVEMBER 2023**

**BONAFIDE CERTIFICATE**

This is to certify that the project report titled **“BOOKVISTA: MACHINE LEARNING -INFUSED BOOK RECOMMENDATION SYSTEM”** is a bonafide work carried out by **VIJAYA RAM BHARATHI V (RA2232014010076)**, **MADHAVAN P K (RA2232014010082), SAI ESHWAR D (RA2232014010085),**

**ELANGO G (RA2232014010110)** under my supervision for the Degree of Master of Applied Data Science

. To my knowledge the work reported here in is the original work done by these students.

**DR. R. KIRUTHIGA Dr. S.ALBERT ANTONY RAJ**

**Assistant Professor Professor & Head**

**Department of Computer Applications Department of Computer Applications (GUIDE)**

**Internal Examiner External Examiner**

# Declaration of Association of Research Project with SDG Goals

This is to certify that the research project entitled

carried out by Mr / Miss

under the supervision of Dr/Mr./Ms. (Designation)

-------------- , of (Department) in partial fulfillment of the requirement for

the award of Under Graduation/Post Graduation/ Diploma/ Ph.D. program has been significantly or potentially associated with SDG Goal No

titled

This study has clearly shown the extent to which its goals and objectives have been met in terms of filling the research gaps, identifying needs, resolving problems, and developing innovative solutions locally for achieving the above-mentioned SDG on a National and/or on an International level.

Signature of the Student Guide and Supervisor

Head of the Department

**ACKNOWLEDGEMENT**

With profound gratitude to the **ALMIGHTY**, We take this chance to thank people who helped us to complete this project.

We take this as a right opportunity to say **THANKS** to our **Parents** who are there to stand with us always with the words “**YOU CAN** ”

We are thankful to **Dr.T.R.Paarivendhar** Chancellor, SRM Institute of Science & Technology who gave us the platform to establish myself to reach Greater heights.

We earnestly thank **Dr. A. Duraisamy, Ph.D.,** Dean, College of Science and Humanities, SRM Institute of Science & Technology who always encourage us to do novel things.

We express our sincere thanks to **Dr. S. Albert Antony Raj, Ph.D.,** Professor and Head, Department of Computer Applications for his valuable guidance and support to execute all incline in learning.

It is our delight to thank our project guide **Dr. R. Kiruthiga**, Assistant Professor, Department of Computer Applications for her help, support, encouragement, suggestions and guidance throughout development phases of the project.

We convey our gratitude to all the **family members of the department** who extended their support through valuable comments and suggestions during the reviews.

A great note of gratitude to **friends** and people who are **known** and **unknown** to us who helped in carrying out this project work a successful one.

# ABSTRACT

The Book Recommendation System using K-Nearest Neighbors (KNN) algorithm in Machine Learning is a project aimed at developing an intelligent system that can recommend books to users based on their preferences and similarities with other books. The project leverages the power of the KNN algorithm to make accurate and personalized recommendations.

The project begins by importing the necessary libraries, including pandas, numpy, matplotlib, and scikit-learn. A dataset containing book information is loaded into a pandas Data Frame for further analysis. Initial data exploration is conducted to understand the structure and characteristics of the dataset. Preprocessing steps are performed to clean and transform the data. This includes handling missing values, removing irrelevant columns, and converting categorical features into numerical representations. The KNN algorithm requires feature scaling, so MinMaxScaler is used to normalize the data. Next, the KNN model is implemented and trained using the preprocessed dataset. The KNN algorithm identifies books that are similar based on their features, such as author, genre, or publication year. The algorithm assigns each book a label based on the labels of its k nearest neighbors.

The trained KNN model is then used to make book recommendations. Given a user's preferences or a specific book, the KNN algorithm identifies the k most similar books and recommends them to the user. The recommendations are based on the calculated distances or similarities between the features of the books. The Book Recommendation System using KNN algorithm in Machine Learning offers a personalized and effective solution for book enthusiasts to discover new books based on their interests and preferences. By leveraging the power of the KNN algorithm, the system provides accurate and relevant recommendations, enhancing the user's reading experience.

# TABLE OF CONTENT

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **CONTENT** | **PAGE.NO** |
| **1** | **INTRODUCTION**  1.1 OVERVIEW OF THE PROJECT  1.2 MODULE DESCRIPTION  1.3 REQUIRED LIBRARIES | 1 |
| **2** | **PROBLEM STATEMENT** 2.1OBJECTIVES | 2 |
| **3** | **SYSTEM ANALYSIS**  3.1 EXISTING SYSTEM  3.2 PROPOSED SYSTEM | 3 |
| **4** | **TECHNOLOGY ADOPTED**  4.1 TECHNIQUES  4.2 METHODOLGY | 5 |
| **5** | **SYSTEM CONFIGURATION**  5.1 HARDWARE SPECIFICATION  5.2 SOFTWARE SPECIFICATION  5.3 SOFTWARE DESCRIPTION | 7 |
| **6** | **SCREEN SHOTS** | 14 |
| **7** | **SAMPLE CODE** | 40 |
| **8** | **CONCLUSION AND FUTURE ENHANCEMENT** | 47 |
| **9** | **REFERENCES** | 48 |

**CHAPTER 1 INTRODUCTION**

**1.1 OVERVIEW OF THE PROJECT**

This is a Python-based book recommendation project focuses on building a basic book

recommendation system using collaborative filtering techniques. By analyzing a dataset containing book information and user ratings, the system generates personalized recommendations based on similar user behavior. It involves data preprocessing, exploratory data analysis, implementation of collaborative filtering algorithms, and evaluation of recommendation performance using metrics such as precision, recall, and accuracy. Through fine-tuning and parameter experimentation, the project aims to enhance the accuracy and effectiveness of the recommendations. These systems leverage machine learning algorithms and data analysis techniques to suggest books based on various factors such as genre, author and user ratings. In this Python-based book recommendation project, how to build a basic book recommendation system is explored. The dataset containing information about books, including their titles, authors, genres, and user ratings is utilized. By analyzing this data and applying collaborative filtering techniques, personalized book recommendations for users is generated.

To implement this project,Python libraries such as Pandas for data manipulation and analysis, and kNN for implementing the recommendation algorithm are used. It started by loading the dataset, cleaning and preprocessing the data, and performing exploratory data analysis to gain insights into the book characteristics and user preferences. Then, on implementing the recommendation algorithm. Collaborative filtering, one of the popular techniques used in recommendation systems. Collaborative filtering predicts user preferences by finding similarities between users or items. The user-item collaborative filtering method, which recommends books based on similar user behavior is used. To evaluate the performance of our recommendation system,the dataset is splitted into training and testing sets. By fine-tuning the algorithm and experimenting with different parameters, It enhance the accuracy and effectiveness of the recommendations. By the end of this project, you will have a practical understanding of how to build a book recommendation system using Python. You will be able to apply the techniques and methodologies learned to create more sophisticated recommendation systems for other domains as well

**1.2 MODULE DESCRIPTION**

**Dependencies used**

As we all know, in machine learning model development libraries are essential. The libraries in a programming language helps finishing the model easier. For this project Analyzing and Visualizing plays the main role. So, Pandas, NumPy, Matplotlib, and Seaborn Libraries are used in this Project.

**Dataset**

A dataset with customers data is essential for this project. Data including customer’s Age, Income, etc. For this Project we’ve chosen an old Dataset which contains Customer’s Age details, their Annual Income and their Spending Score.

**Analyzing Data**

We are doing this project completely on Google Colab platform which is completely Cloud Based. We are analyzing the dataset by describing it, checking its shape, checking for missing data, viewing its properties, and cleaning the dataset for further Development.

**Visualizing Data**

Visualizing data includes all types of graph plots which are Bar plot, Relation Plot, Displot, Violin Plot, 2D Graph, 3D Graph… etc.

**1.3.REQUIRED LIBRARIES**

* Numpy (pip install numpy)
* Pandas (pip install pandas)
* Matplotlib (pip install matplotlib)
* Seaborn (pip install seaborn)
* Sklearn (pip install sklearn)
* mpl\_toolkits (pip install mpl\_toolkits)

# CHAPTER 2 PROBLEM STATEMENT

To develop a book recommendation system based on the dataset of books. The system

should provide personalized book recommendations to users based on different criteria such as publishers, authors, and languages. The system should also allow users to explore popular books, analyze book ratings, and discover similar books based on their preferences.

## 2.1 Objectives of the book recommendation system:

* Provide top book recommendations based on publishers: Given a publisher's name, recommend the top-rated books published by that publisher.
* Provide top book recommendations based on authors: Given an author's name, recommend the highest-rated books written by that author.
* Provide top book recommendations based on language: Given a language, recommend the best-rated books available in that language.
* Implement a data preprocessing step: Convert the average rating feature to numeric values and handle any missing or erroneous data.
* Build a book recommendation model: Use the features such as average rating, ratings count, and categorical variables like rating objective and language to build a book recommendation model.
* Implement a book recommender function: Develop a function that takes a book's name as input and returns a list of similar books based on the nearest neighbors algorithm.
* Create interactive widgets: Allow users to interact with the recommendation system by selecting a book and getting recommendations based on their preferences.
* The book recommendation system aims to enhance the book browsing experience for users by providing personalized recommendations and helping them discover new books based on their interests and preferences.

# CHAPTER 3

# SYSTEM ANALYSIS

**3.1 EXISTING SYSTEM:**

In existing book recommendation systems, users can manually search for highly-rated books by utilizing search features or filters that allow them to refine their choices based on factors like user ratings, reviews, and popularity. These systems provide users with the ability to explore and discover books that have garnered positive feedback from other readers, ensuring they can make informed decisions when selecting their next read. While this manual approach relies on user input, it remains a valuable feature in book recommendation platforms, offering users the freedom to find books based on their preferences, trusted reviews, and high ratings.

**3.2 PROPOSED SYSTEM:**

Traditional book recommendation systems encompass various techniques, including collaborative filtering (user-based and item-based), content-based filtering, matrix factorization, hybrid systems, demographic filtering, popularity-based recommendations, item-to-item recommendations, association rule mining, machine learning algorithms, clustering methods, temporal recommendations, and evaluation metrics. These systems leverage user behavior, content analysis, demographics, and other data to offer personalized book suggestions. While effective, modern systems extend beyond traditional methods, incorporating advanced technologies like deep learning and natural language processing to enhance recommendation accuracy and user satisfaction. These contemporary systems adapt to evolving user preferences, consider contextual factors, ensure privacy, and provide explanations, ultimately delivering more tailored and diverse book recommendations**.**

# CHAPTER 4 TECHNOLOGY ADOPTED

Here Machine learning concept is used, It has significantly impacted the field of book recommendation

systems. Traditional book recommendation systems often relied on basic algorithms like collaborative filtering or content-based filtering, which had limitations in providing personalized and accurate recommendations. However, with the advent of machine learning techniques, book recommendation systems have become more sophisticated and effective.

## 4.1 Techniques used in machine learning-based book recommendation systems:

1. Collaborative Filtering: Collaborative filtering is a popular technique that analyzes user behavior and preferences to recommend books. It identifies patterns by examining the behavior and preferences of similar users and uses this information to suggest books to individual users. Collaborative filtering can be further divided into two types: user-based filtering and item-based filtering.
2. Content-Based Filtering: Content-based filtering focuses on the characteristics of the books themselves, such as genre, author, language, or keywords. Machine learning algorithms can analyze these features and create recommendations based on the similarities between books. For example, if a user has shown interest in science fiction books in the past, the system may recommend other science fiction books.
3. Hybrid Approaches: Hybrid approaches combine collaborative filtering and content-based filtering to provide more accurate recommendations. Machine learning algorithms can be used to leverage the strengths of both approaches and offer personalized recommendations based on user preferences and book characteristics.
4. Deep Learning: Deep learning techniques, such as neural networks, have been employed in book recommendation systems to handle complex patterns and relationships in data. Neural networks can learn representations of books and users that capture their preferences and interests, enabling more accurate recommendations.
5. Natural Language Processing (NLP): NLP techniques are used to extract meaningful information from book descriptions, summaries, or reviews. By analyzing the text content, machine learning algorithms can identify similarities between books and provide relevant recommendations.

## 4.2 Methodalogy used in machine learning-based book recommendation systems:

Machine Learning has revolutionized book recommendation systems by enabling more accurate, personalized, and context-aware recommendations. These techniques continue to evolve, leveraging advancements in AI to enhance the user experience and help readers discover books that align with their interests and preferences.

In this project, The aim is to create a book recommendation algorithm using the K-Nearest

Neighbors (KNN) algorithm. The algorithm measures the distance between instances to determine their "closeness" and recommends books based on similarity and utilized the Books dataset and ratings dataset to train our model. The Books dataset contains 11,128 rows and 12 columns, while the ratings dataset contains 1,048,576 rows and 4 columns. Our goal was to develop a model that can suggest books similar to a given book by leveraging the NearestNeighbors class.

1. Data Import and Cleaning: The project started by importing necessary libraries such as pandas, matplotlib and seaborn. Then the dataset was loaded using the **pd.read\_csv()** function and performed initial data cleaning tasks. Renaming Columns: If necessary, rename columns to make them more descriptive using rename().
2. Exploratory Data Analysis: To gain insights into the dataset, by conducting exploratory data analysis. This phase involved examining the dataset's shape, column names, data types, missing values, and descriptive statistics. Feature engineering, extracting the publication year from the publication date and converting it into an integer data type. This allowed us to perform further analysis based on the publication year.
3. Visualizations and Insights: During the exploratory data analysis, Various visualizations to gain a better understanding of the dataset is created. I visualized the distribution of books published in specific years, identified the top authors with the most published books, and examined the distribution of books across different languages and publishers is visualized. Additionally, the most occurring books based on their titles is identified. Also visualized the average ratings of books and identified the books with the highest average rating.
4. Interactive Recommendation Functions: To provide an interactive user experience, The functions that recommend books based on specific criteria such as publishers, authors, and languages is developed. By selecting a specific publisher, author, or language, the functions returned a list of recommended books based on the chosen criterion.
5. Data Preprocessing: Before building the recommendation model, data preprocessing steps is performed. The average rating column to numeric values and created dummy variables for the rating object and language code columns using one-hot encoding is converted. These features were then concatenated with other relevant features and scaled using the MinMaxScaler.
6. Building the Recommendation Model: Using the preprocessed features, a Nearest Neighbors model using the NearestNeighbors class is constructed. The model was trained on the transformed data and could be used to find the nearest neighbors and recommend similar books based on a given book name.
7. Interactive Recommendation Widget: To provide a user-friendly experience, an interactive widget where users could input a book name is implemented. The widget then returned a list of recommended books based on similarity. These recommendations were determined by finding the nearest neighbors in the feature space.

This project demonstrated the process of building a book recommendation system using the K-Nearest Neighbors algorithm. It is started by importing and cleaning the data, followed by conducting exploratory data analysis to gain insights into the dataset. Then develop interactive functions for recommending books based on publishers, authors, and languages. Data preprocessing was performed to transform the data, and a Nearest Neighbors model was built using NearestNeighbors class. Finally, interactive widget that allowed users to input a book name and receive recommendations based on similarity is created. This project showcased the application of machine learning algorithms in creating personalized book recommendations, enhancing the user's reading experience.

# CHAPTER 5

**SYSTEM CONFIGURATION**

**5.1 HARDWARE SPECIFICATION**

Processor : Intel core2

Memory : 4 GB RAM or More

Hard disk Requirement : Free 500GB on installation drive

**5.2 SOFTWARE SPECIFICATION**

* Browser with latest version Installed with Good Web Application Support
* Google Colab Platform
* Language: Python in Machine Learning
* Customer Dataset.

# 5.3 SOFTWARE DESCRIPTION:

The tools used are Google Colab using Python and various libraries such as Pandas,

NumPy, Matplotlib, Seaborn. Here is a breakdown of the tools and concepts used .

## About Python

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes. It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming. Python combines remarkable power with very clear syntax. It has interfaces to many systems calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants including Linux and macOS, and on Windows.

## Python in Machine learning

This Machine Learning, as the name suggests, is the science of programming a computer by which they are able to learn from different kinds of data. In the modern days, it is become very much easy and efficient compared to the olden days with various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reasons is its vast collection of libraries.

## Python in Google Colab

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

7

## Features of Python

* Open Source Libraries
* Support of Community
* Machine Learning
* Data Science

## PANDAS

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis/manipulation tool available in any language. It is already well on its way toward this goal.

## Pandas is well suited for many different kinds of data:

* Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
* Ordered and unordered (not necessarily fixed-frequency) time series data.
* Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
* Any other form of observational / statistical data sets. The data need not be labeled at all to be placed into a panda’s data structure

## Data Structures:

* + SERIES: 1D labeled homogeneously-typed array
  + DATAFRAME: General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column

**NUMPY**

NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python.

NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.

NumPy Arrays are faster than Python Lists because of the following reasons: An array is a collection of homogeneous data-types that are stored in contiguous memory locations. On the other hand, a list in Python is a collection of heterogeneous data types stored in non-contiguous memory locations.

**MATPLOTLIB**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. Matplotlib makes it easy to generate plots, histograms, power spectra, bar charts, error charts, and other kinds of plots, with just a few lines of code. Interactive Applications Using Matplotlib will teach you how to turn your plots into fully interactive applications for data exploration and information synthesis.

**SEABORN**

Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with data frames and the Pandas library. The graphs created can also be customized easily. Below are a few benefits of Data Visualization. Matplotlib and Seaborn are popular libraries among the data science community that generate beautiful plots and charts for visualization, so what's the requirement

of Interactive Visualization.

**IPY WIDGETS**

ipywidgets, also known as jupyter-widgets or simply widgets, are interactive HTML widgets for Jupyter notebooks and the IPython kernel. Notebooks come alive when interactive widgets are used. Users gain control of their data and can visualize changes in the data. Learning becomes an immersive, fun experience. ipywidgets are visual elements that allow users to specify parameter values in notebook cells. You can use ipywidgets to make your Databricks Python notebooks interactive.

## Process of my Project:

1. Libraries:Here several libraries for data manipulation, visualization, and interactive widgets is utilized. These include pandas for data handling, numpy for numerical operations, matplotlib and seaborn for data visualization, and ipywidgets for creating interactive elements in Colab.
2. Data Preprocessing: Then data preprocessing steps such as cleaning the column names, handling missing values, and converting data types were performed. It also includes feature engineering by extracting the year from the publication date.
3. Exploratory Data Analysis (EDA): It contains EDA to gain insights into the dataset. It includes analyzing book counts by year, top authors and publishers, language distribution, and visualizations of book occurrences and average ratings.
4. Book Recommendation: In this project three types of book recommendations based on publishers, authors, and language are used. It allows users to interactively select a publisher, author, or language and retrieves the top-rated books associated with the chosen category.
5. Data Preprocessing for Model Building: Then the dataset for model building by encoding categorical variables (rating objects and language codes) using one-hot encoding and scaling the numerical features using MinMaxScaler is performed.
6. Model Building: Then the k-nearest neighbors (KNN) algorithm for building the recommendation model is used. It creates a KNN model using the NearestNeighbors class from sklearn.neighbors module and fits it with the preprocessed features. The model finds the nearest neighbors based on Euclidean distance.
7. Book Recommendation with Model: Then the KNN model to provide book recommendations based on user input is trained. Users can interactively select a book title, and then it retrieves the top similar books based on the fitted KNN model.

# 

# 

# FIG-5.1

# CHAPTER 6

**SCREEN SHOTS**

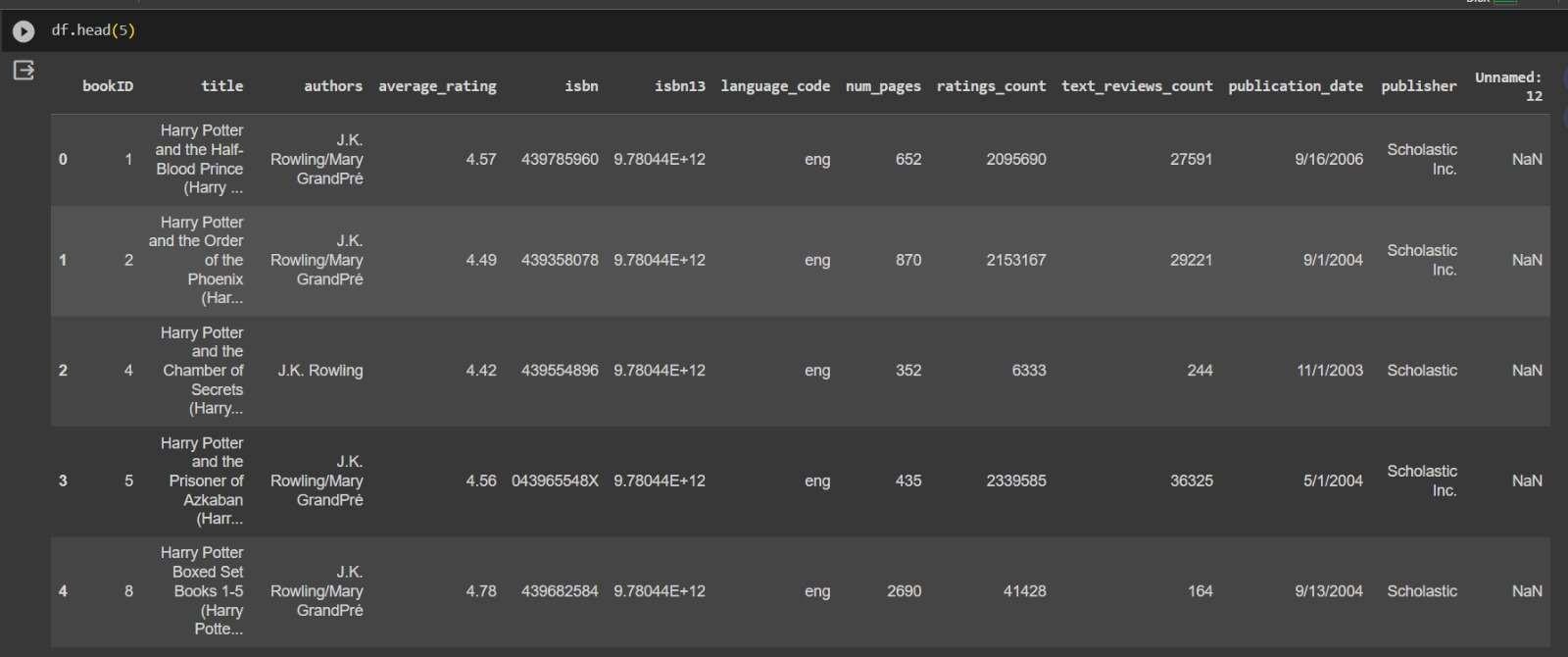
## Import libraries:

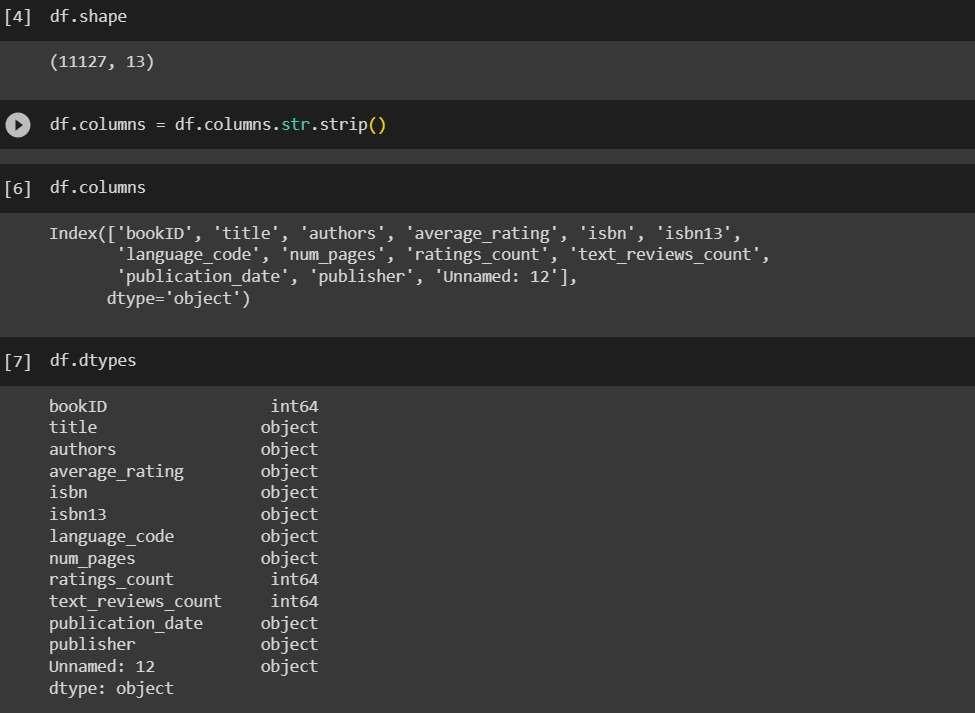


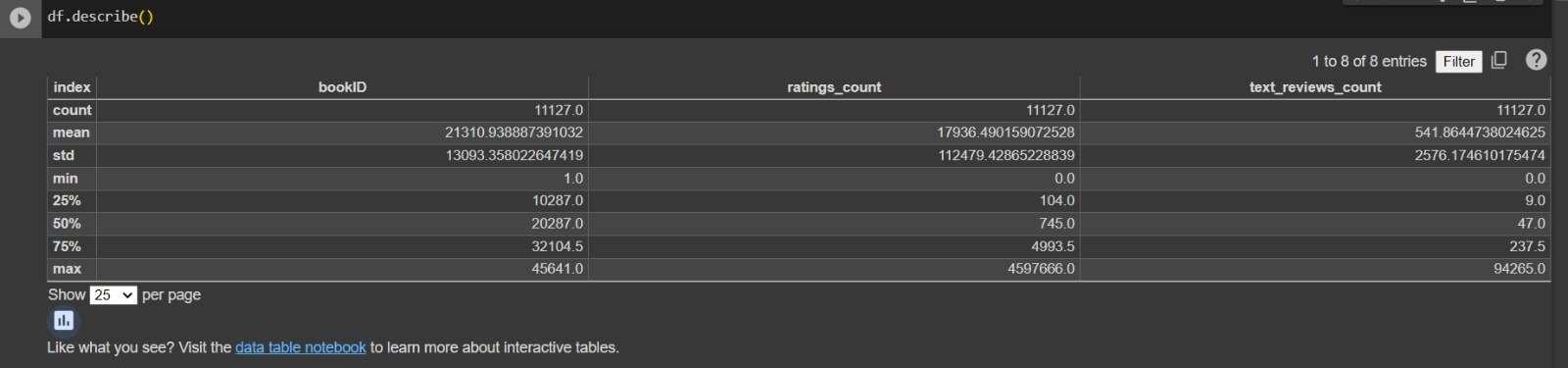
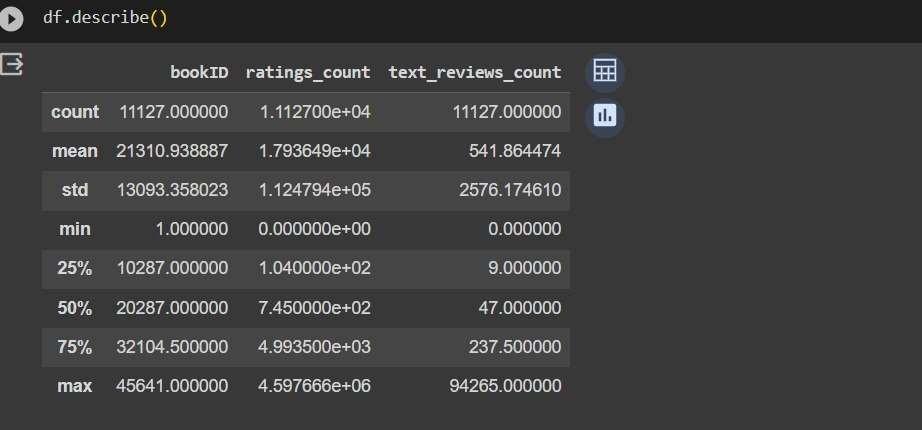
**Connecting Google Drive in Colab to Load the Dataset:**



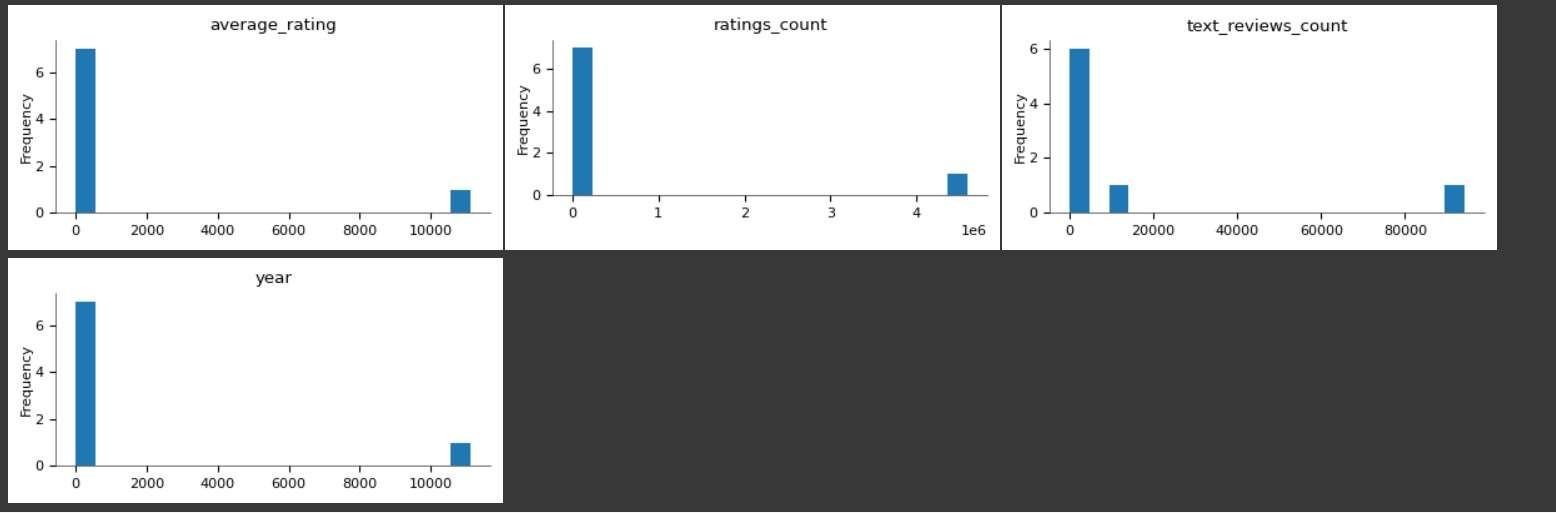
## Data Pre processing:



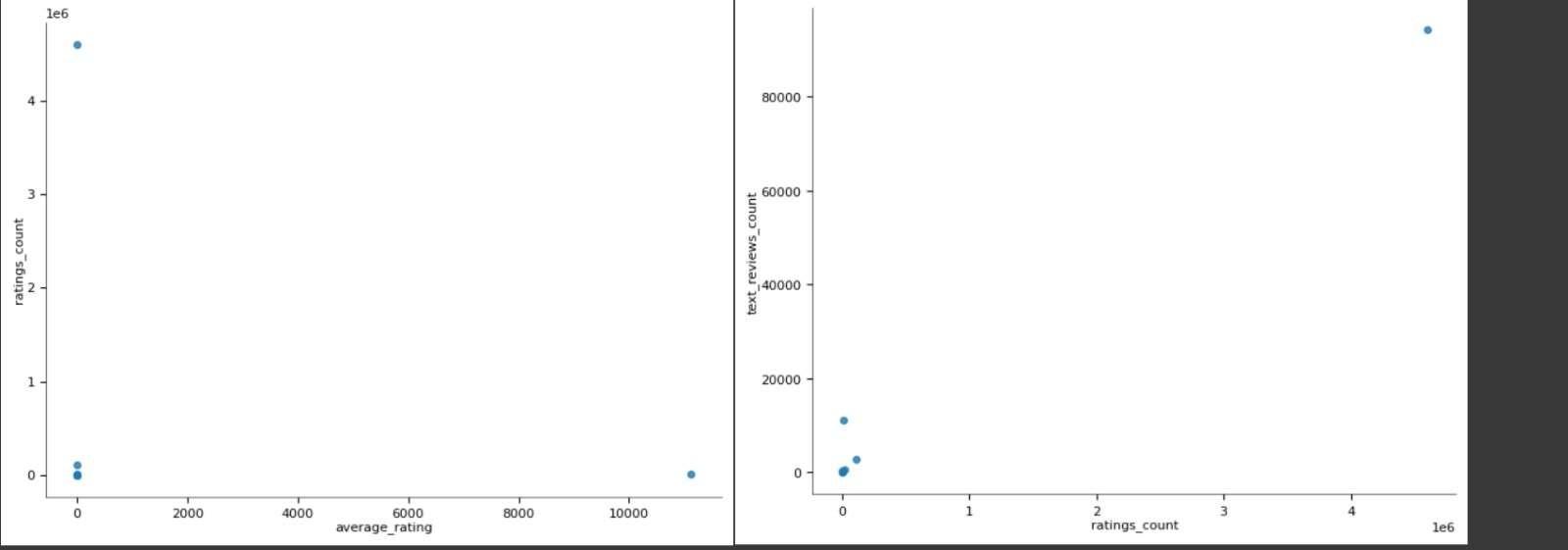




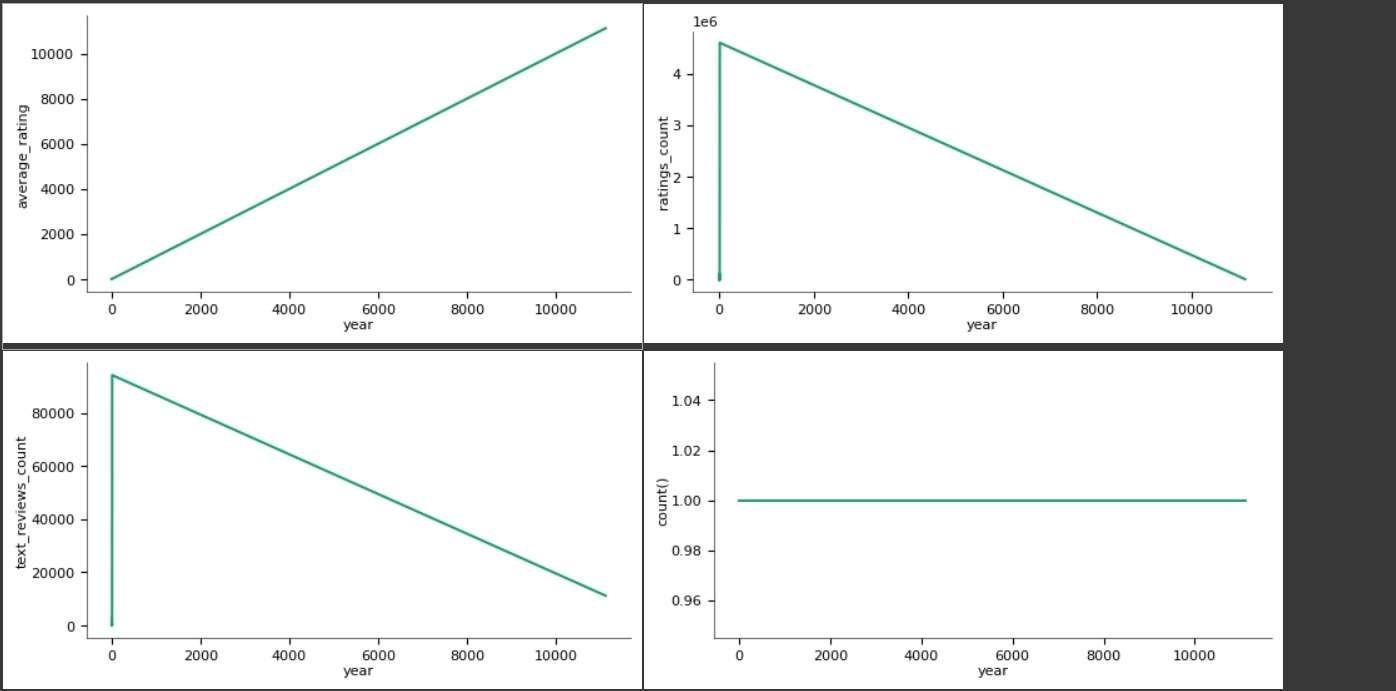
**DISTRIBUTIONS**:



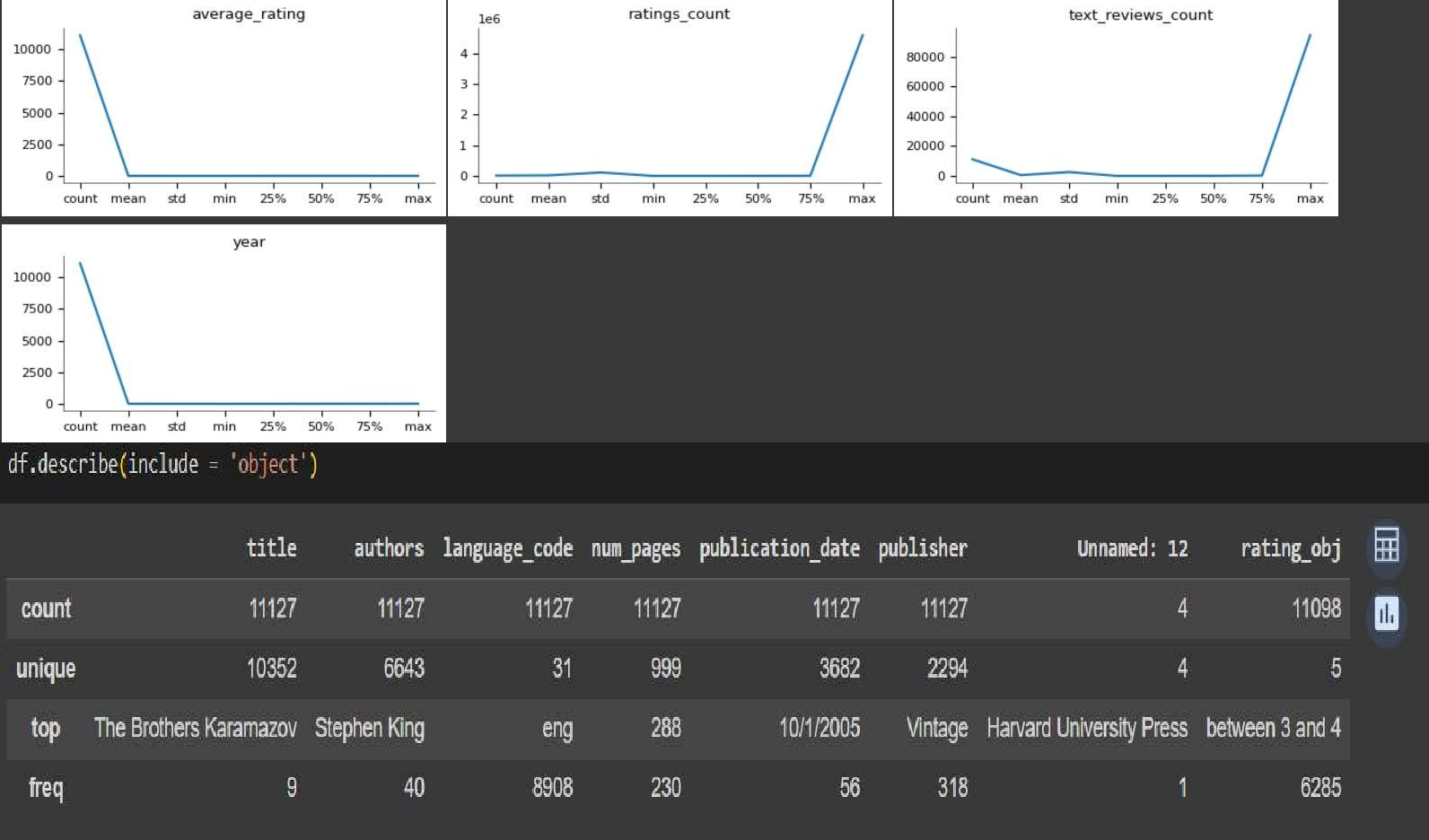
## 2d DISTRIBUTION:



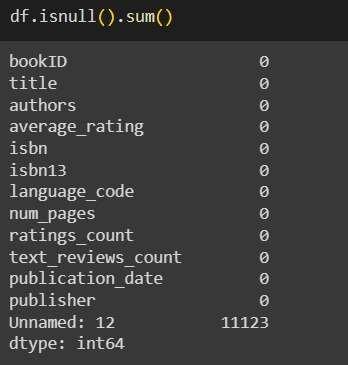
**TIME SERIES:**



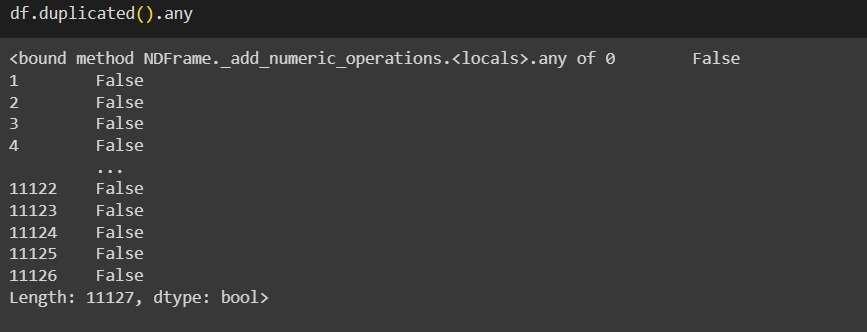
## VALUES:



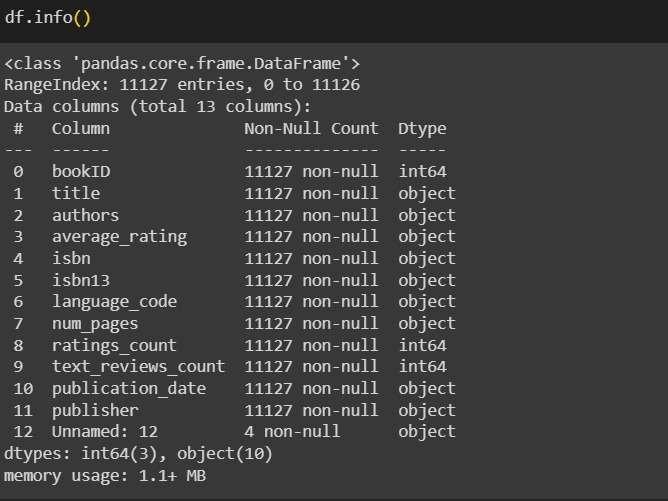
**CHECKING NULL VALUES:**



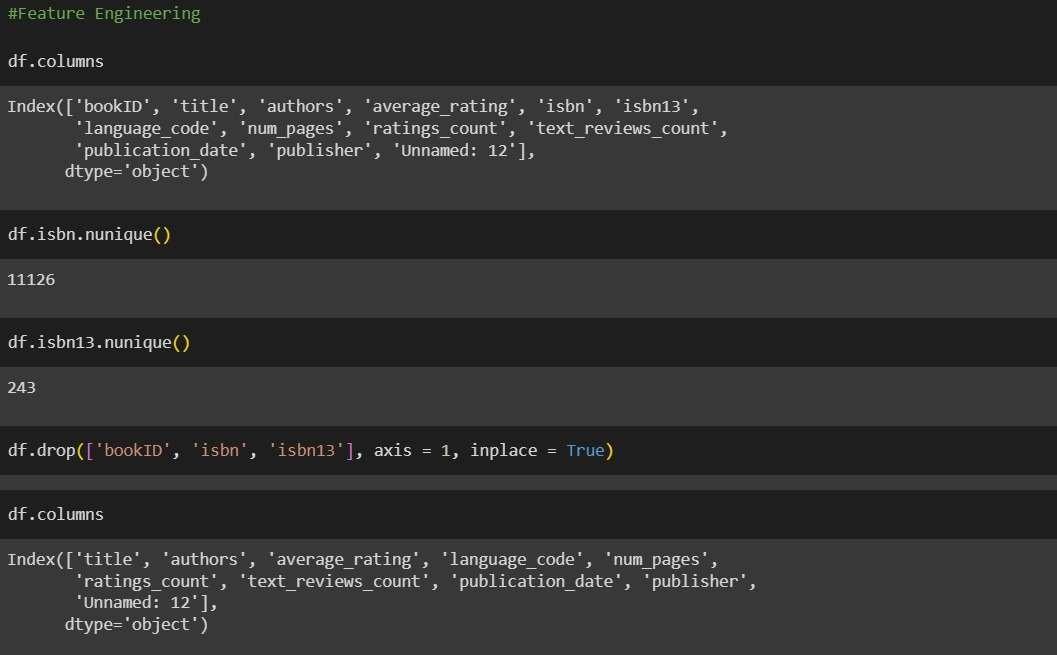
## CHECKING DUPLICATES:

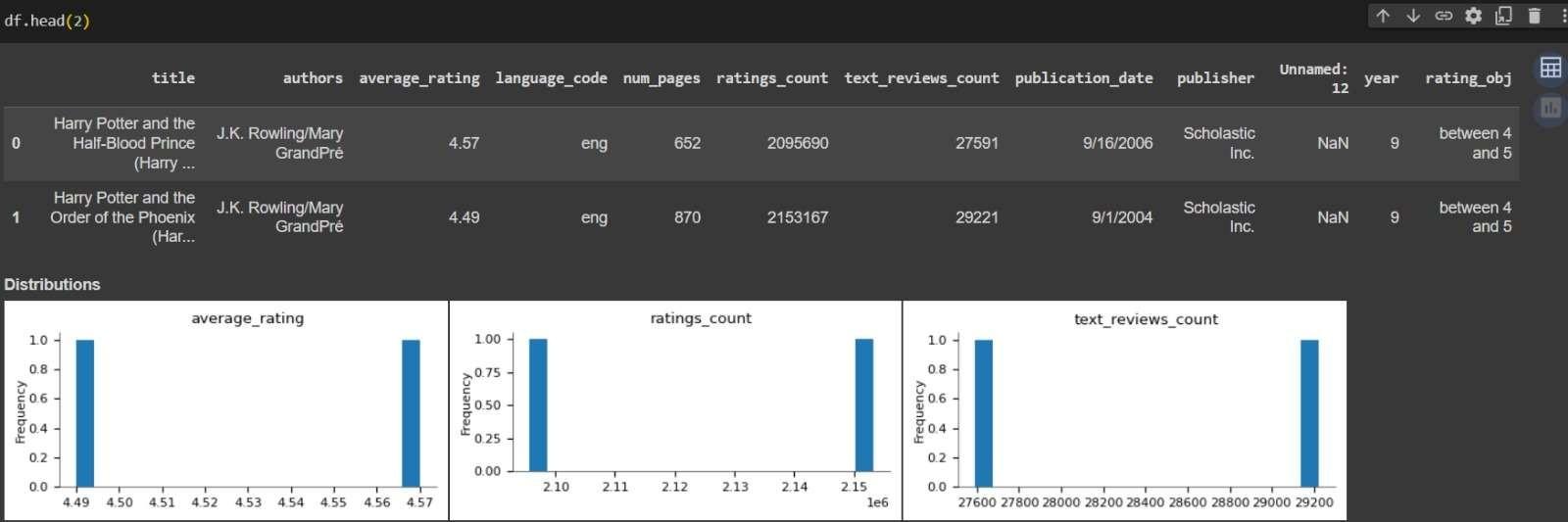


**INFORMATION ABOUT THE DATASET:**

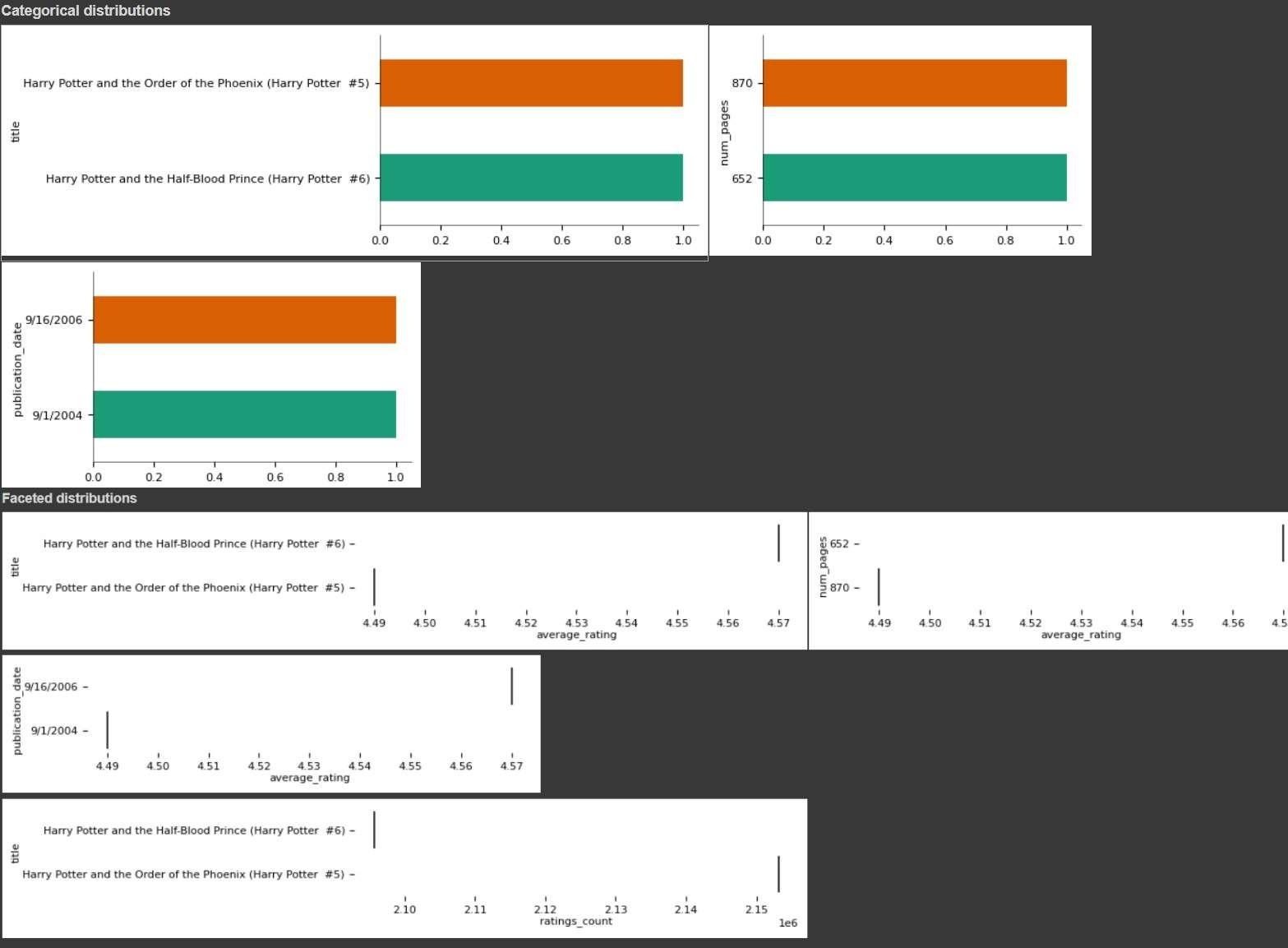


## FEATURE ENGINEERING:

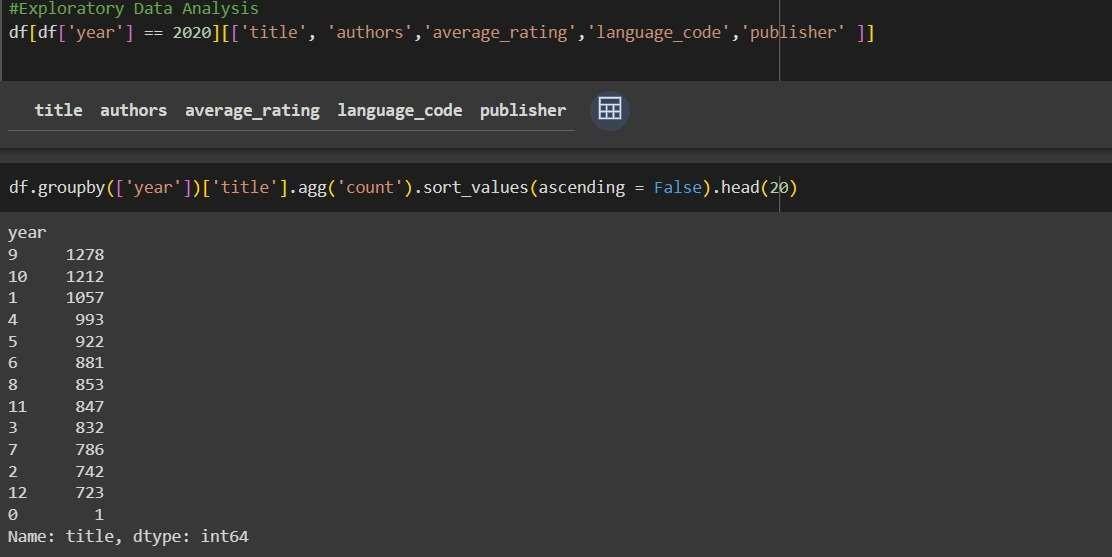




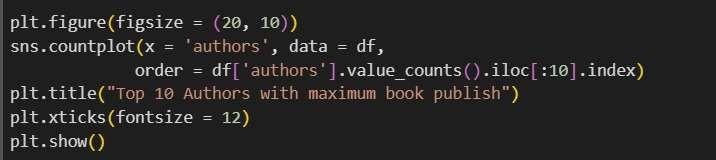
**CATEGORICAL DISTRIBUTION:**



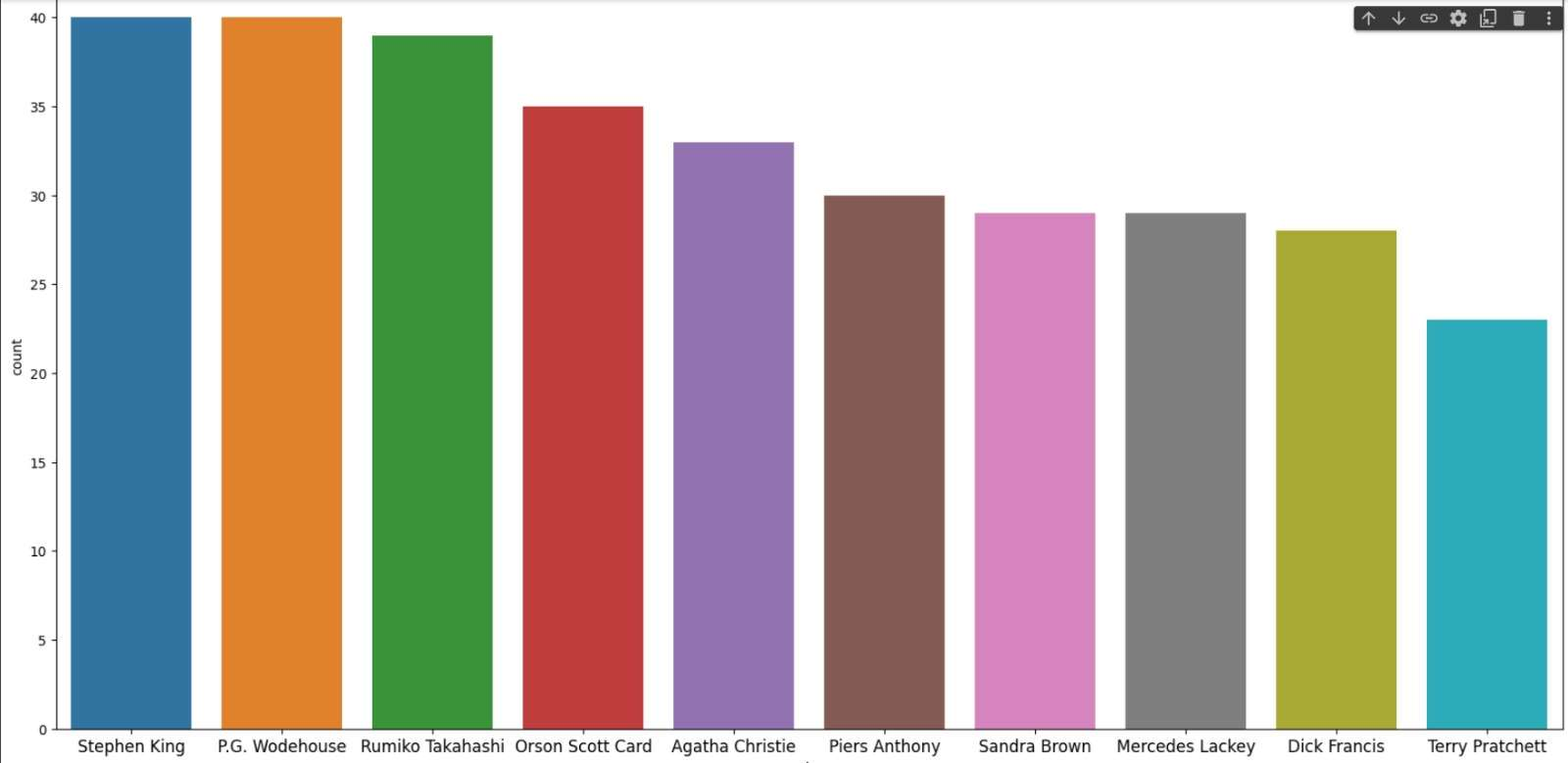
## EXPLORATORY DATA ANALYSIS:

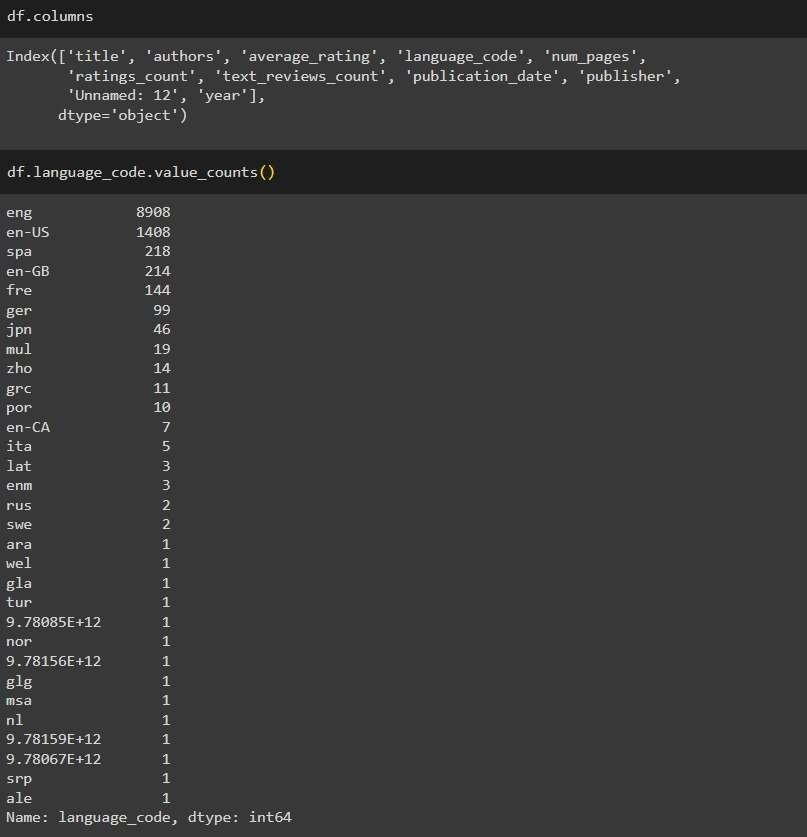


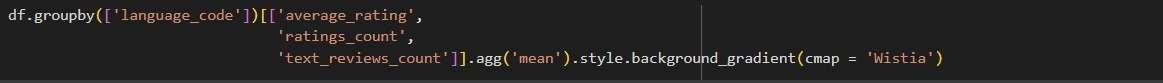
**VISUALIZATION:**

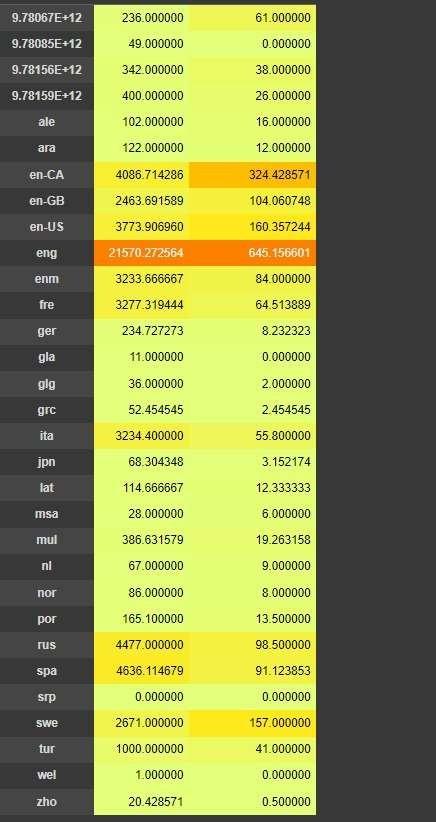


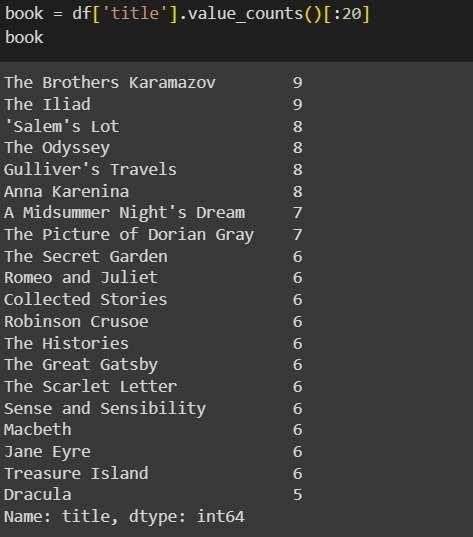
## TOP 10 AUTHORS WITH MAXIMUM BOOK PUBLISH



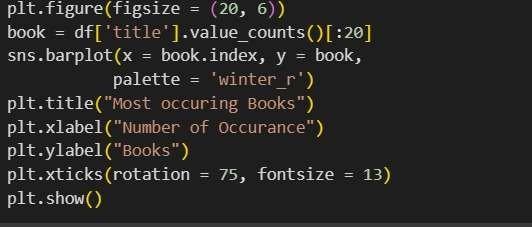


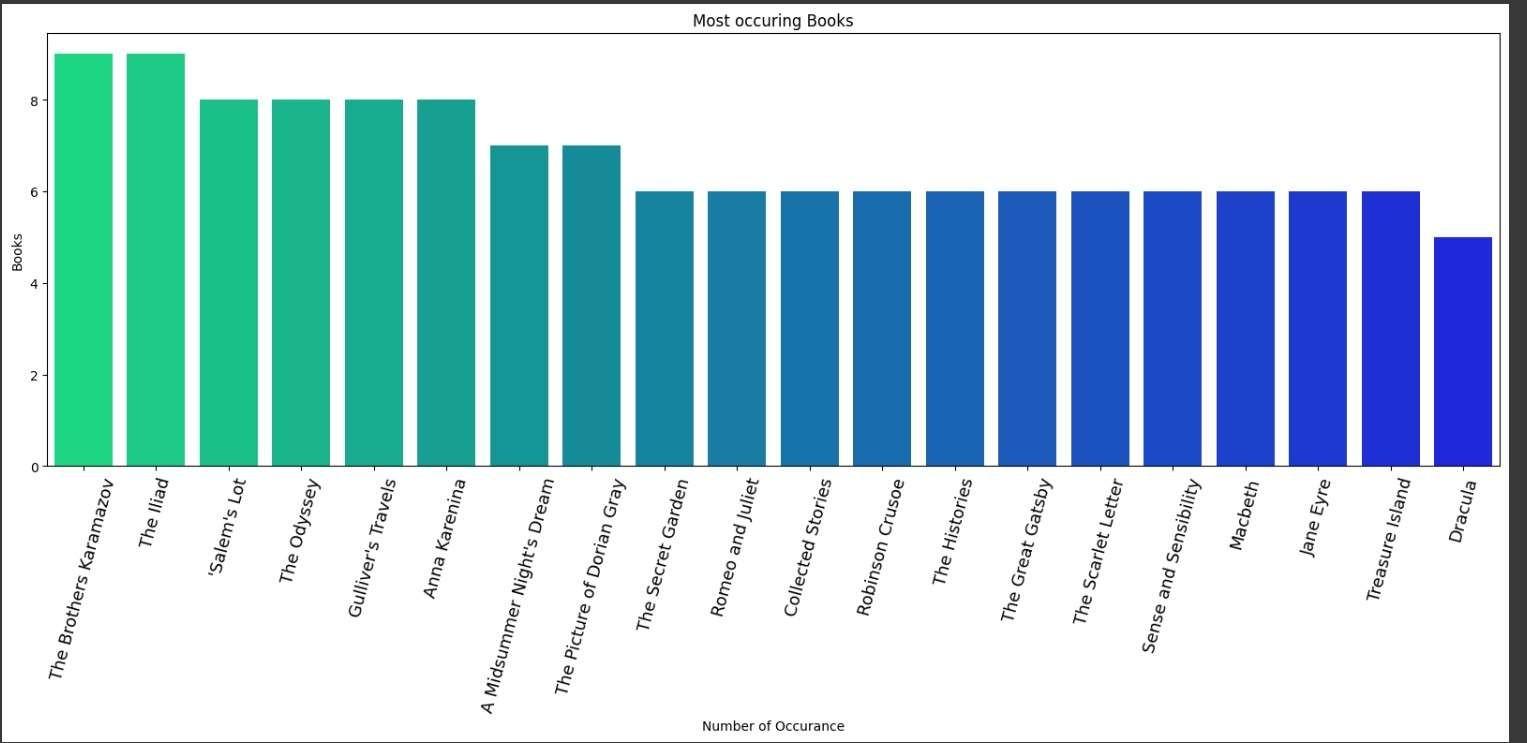


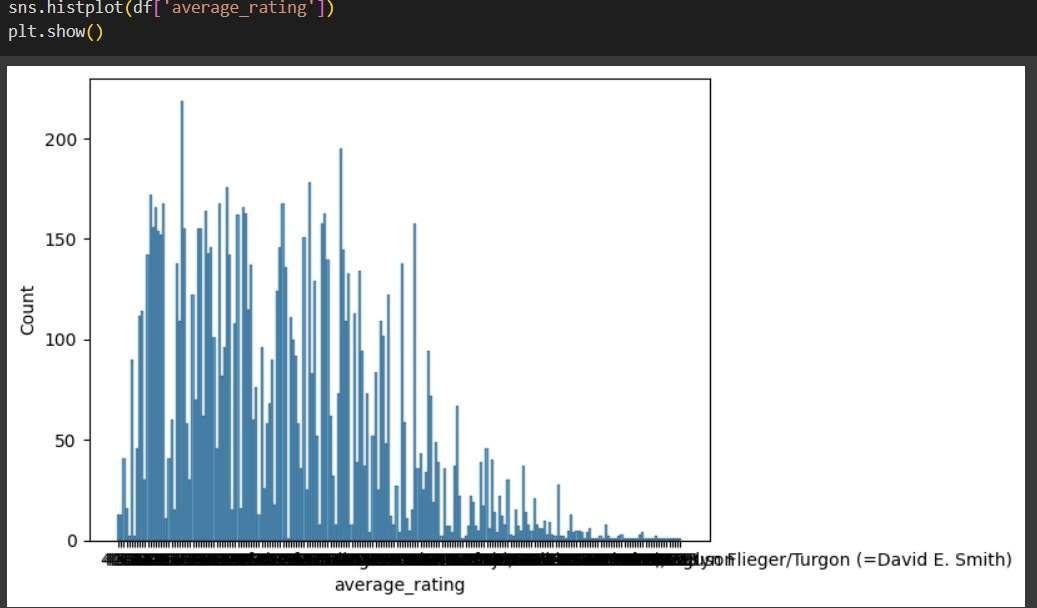




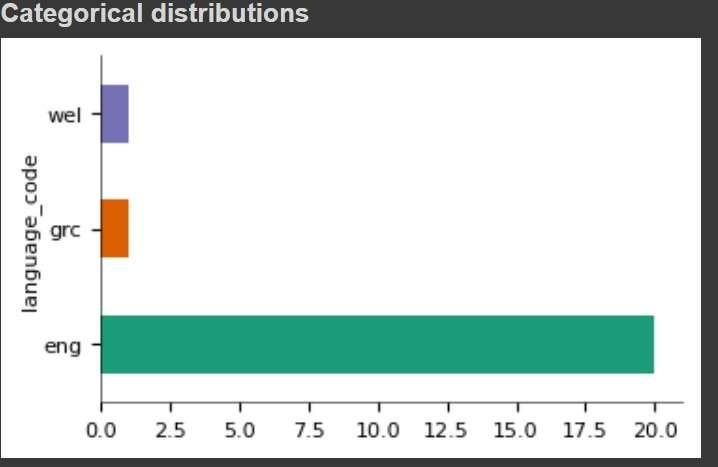
**FINDING THE MOST OCCURING BOOK IN DATA:**

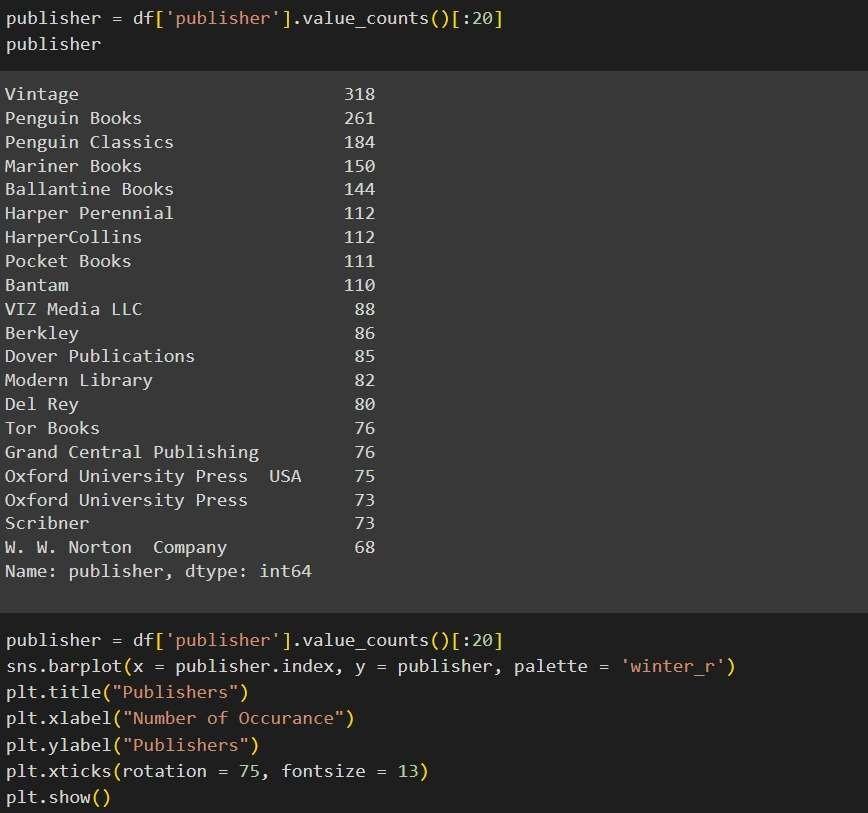


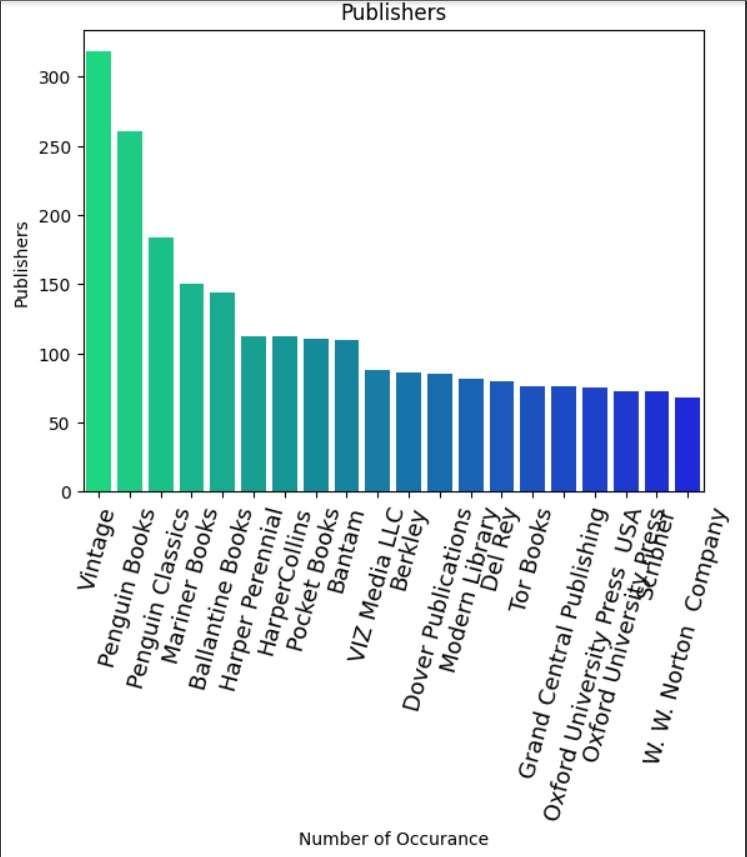




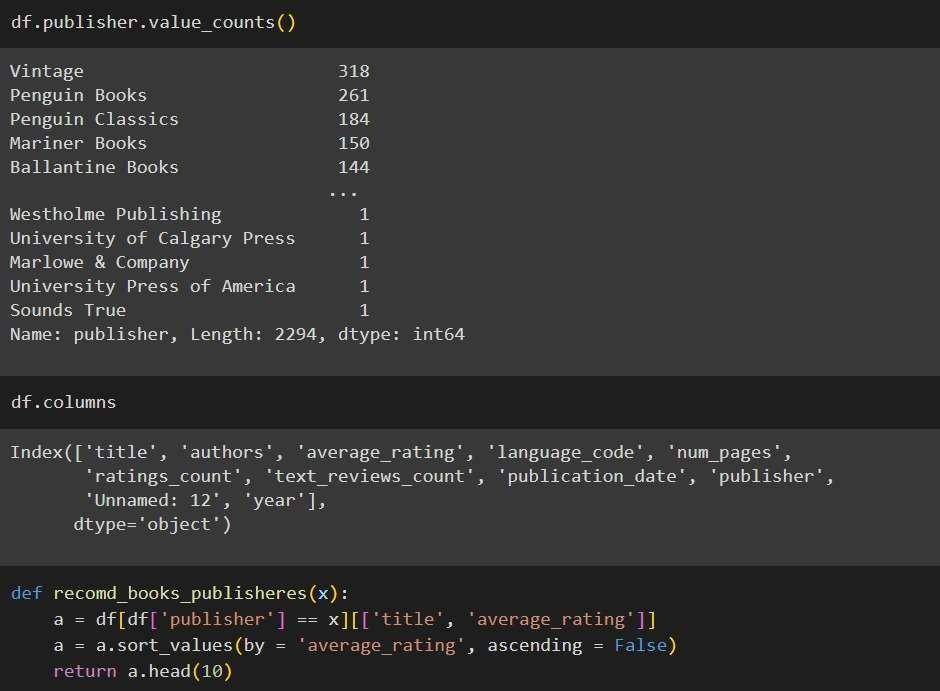






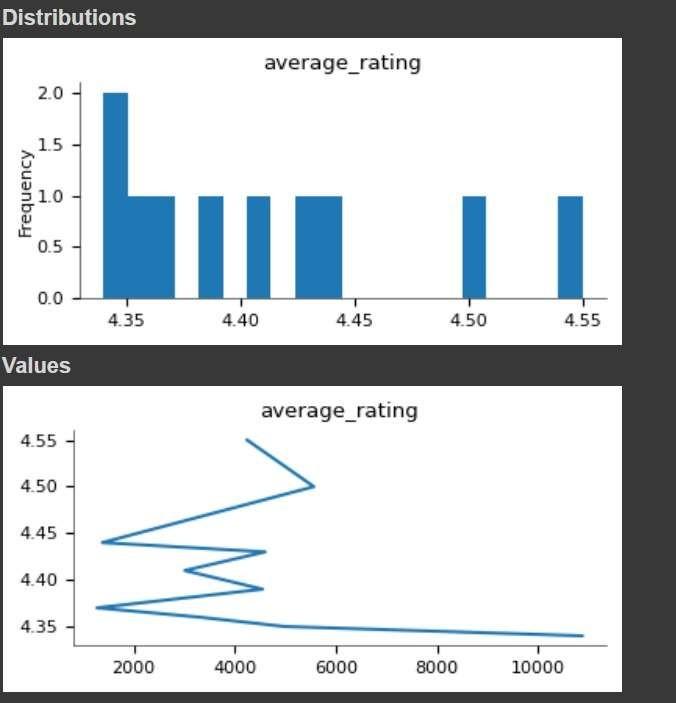


## RECOMMENDING BOOKS BASED ON PUBLISHERS:

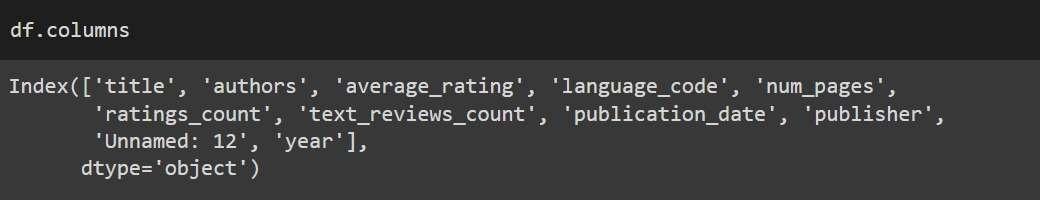




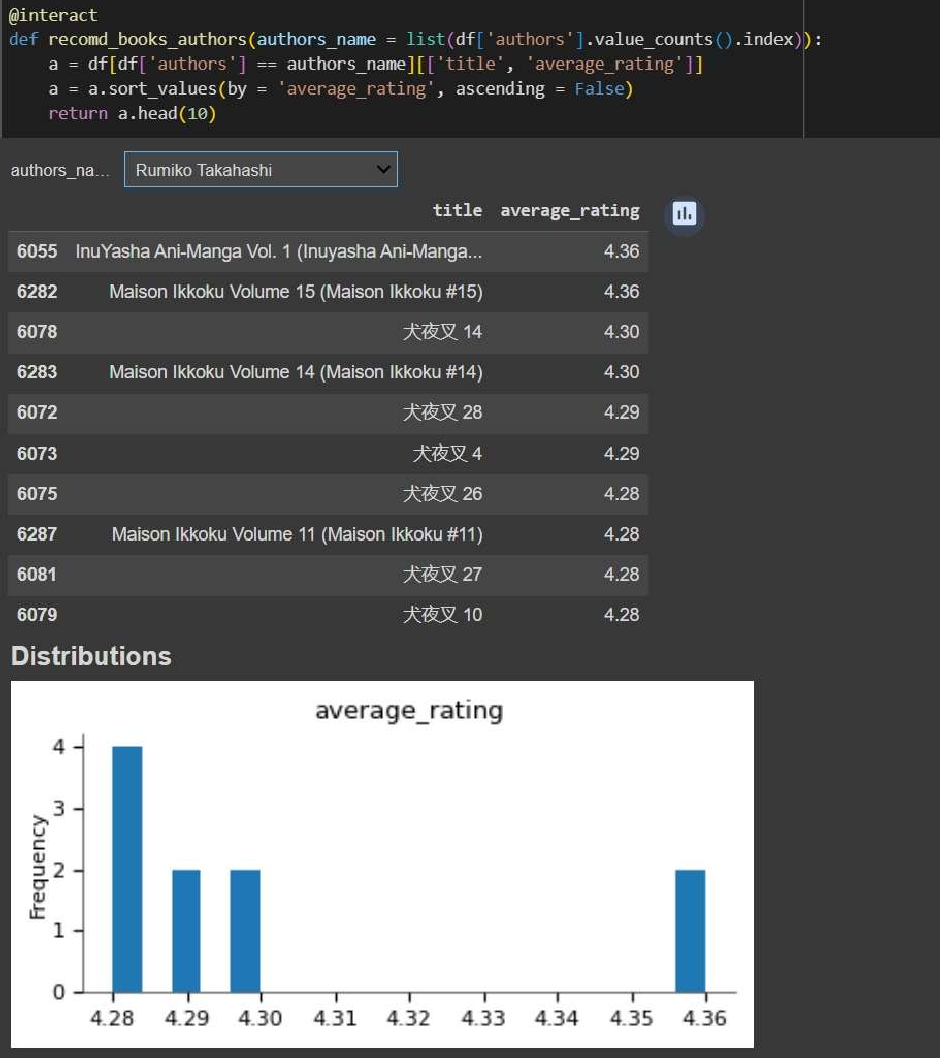


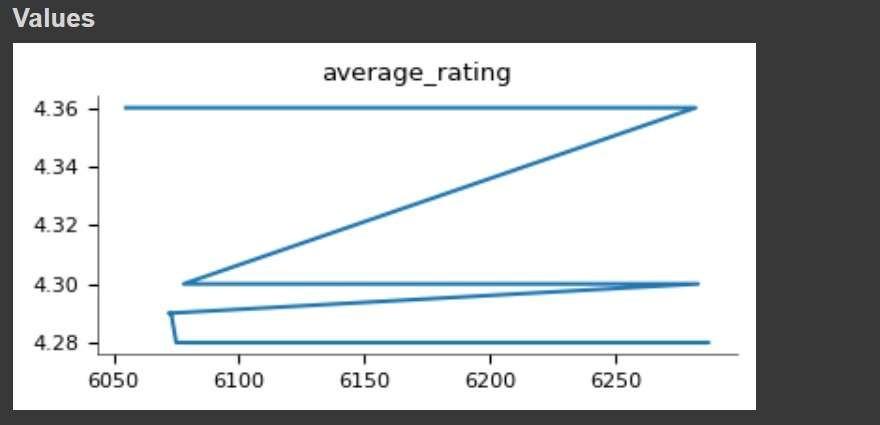




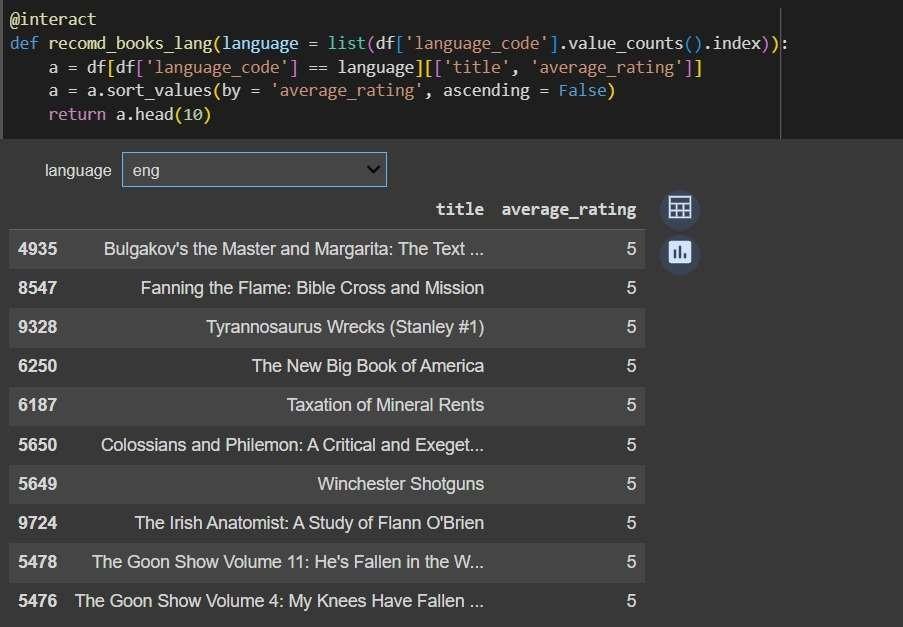


**RECOMMENDING BOOKS BASED ON AUTHORS:**

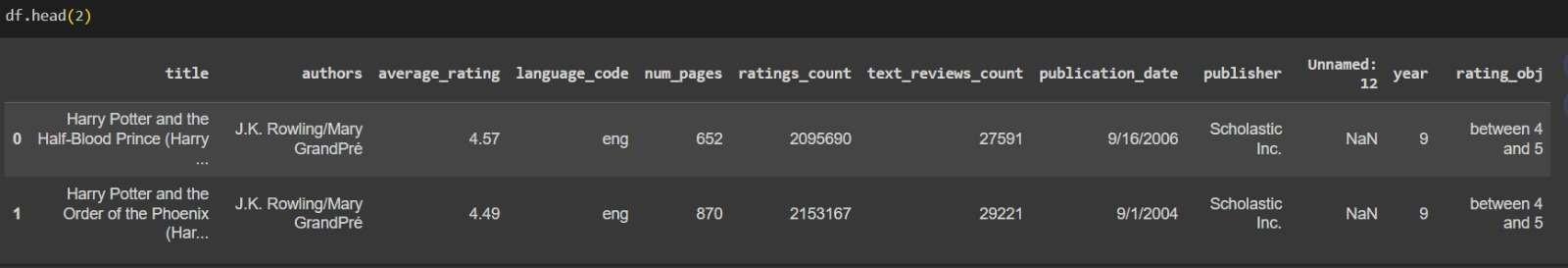


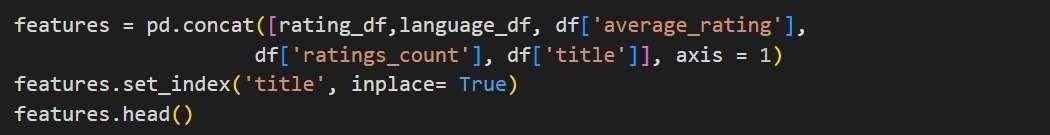
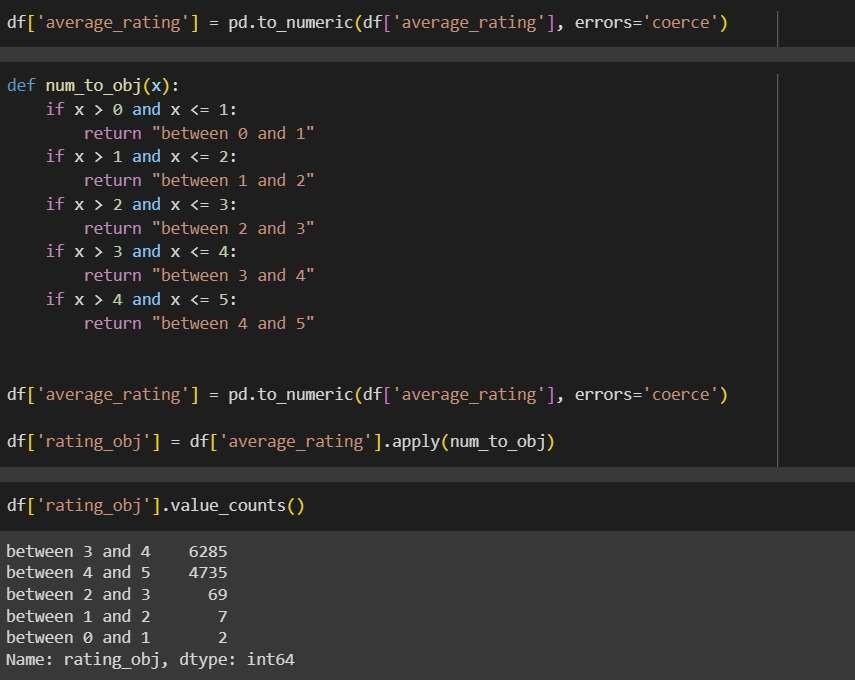


## RECOMMENDING BOOKS BASED ON LANGUAGES:

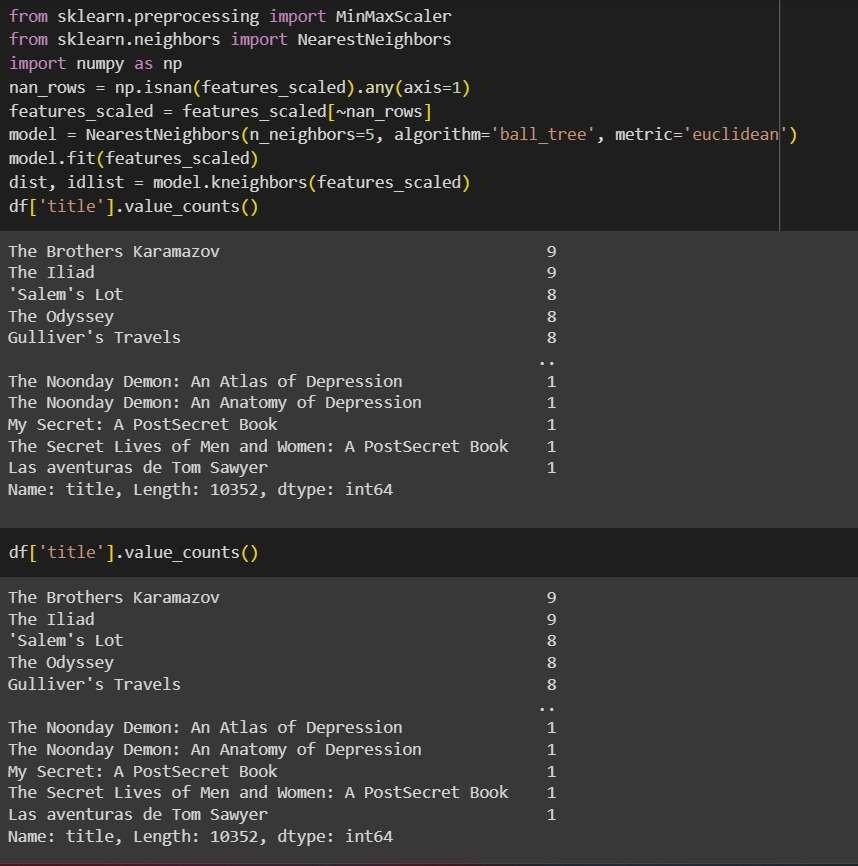


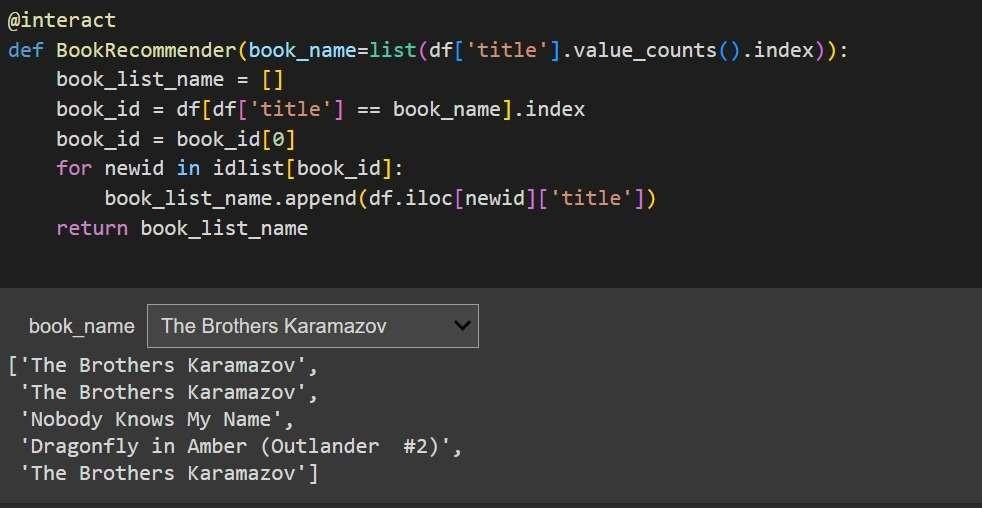
**DATA PREPROCESSING:**

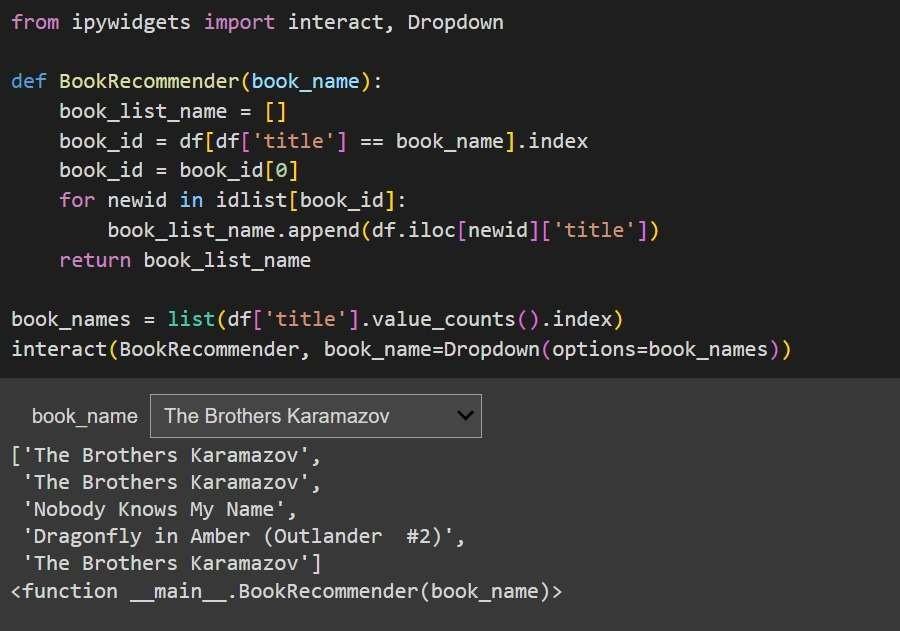




**MODEL BUILDING:**







**CHAPTER 7**

# PROGRAM

import pandas as pd import numpy as np

# For data visualisation import matplotlib.pyplot as plt import seaborn as sns

# For interactive plots import ipywidgets

from ipywidgets import interact

from ipywidgets import interact\_manual

df = pd.read\_csv("/content/books.csv", error\_bad\_lines = False)

df.head(5) df.shape df.columns df.columns =

df.columns.str.strip() df.columns df.dtypes

df.describe() df.describe(include = 'object') df.isnull().sum()

df.duplicated().any df.info()

#Feature Engineering df.columns df.isbn.nunique() df.isbn13.nunique()

df.drop(['bookID', 'isbn', 'isbn13'], axis = 1, inplace =

True) df.columns df.publication\_date

df['year'] = df['publication\_date'].str.split('/') df['year']

= df['year'].apply(lambda x:x[0])

df.head(2) df.dtypes

df['year'] = df['year'].astype('int') df.dtypes

df.columns df['year'].min()

df['year'].max()

#Exploratory Data Analysis

df[df['year'] == 2020][['title', 'authors','average\_rating','language\_code','publisher'

]] df.groupby(['year'])['title'].agg('count').sort\_values(ascending = False).head(20) plt.figure(figsize = (20, 10))

sns.countplot(x = 'authors', data = df, order =

df['authors'].value\_counts().iloc[:10].index) plt.title("Top 10 Authors with maximum book publish") plt.xticks(fontsize = 12)

plt.show()

df.language\_code.value\_counts() df.groupby(['language\_code'])[['average\_rating',

'ratings\_count', 'text\_reviews\_count']].agg('mean').style.background\_gradient(cmap = 'Wistia')

book = df['title'].value\_counts()[:20] book

# To find most occuring book in our data plt.figure(figsize = (20, 6)) book = df['title'].value\_counts()[:20] sns.barplot(x = book.index, y = book,

palette = 'winter\_r') plt.title("Most occuring Books") plt.xlabel("Number of Occurance") plt.ylabel("Books")

sns.histplot(df['average\_rating']) plt.show()

df[df.average\_rating == df.average\_rating.max()][['title','authors','language\_code','publisher']] publisher = df['publisher'].value\_counts()[:20]

publisher

publisher = df['publisher'].value\_counts()[:20] sns.barplot(x = publisher.index, y = publisher, palette = 'winter\_r') plt.title("Publishers")

plt.xlabel("Number of Occurance") plt.ylabel("Publishers") plt.xticks(rotation = 75, fontsize = 13) plt.show()

#Recommending Books based on Publishers #Recommending Books based on Authors #Recommending Books based on Language

df.publisher.value\_counts(

) df.columns

def recomd\_books\_publisheres(x):

a = df[df['publisher'] == x][['title', 'average\_rating']] a = a.sort\_values(by = 'average\_rating', ascending = False) return a.head(10)

recomd\_books\_publisheres('Vintage')

@interact

def recomd\_books\_publishers(publisher\_name = list(df['publisher'].value\_counts().index)): a = df[df['publisher'] == publisher\_name][['title', 'average\_rating']]

a = a.sort\_values(by = 'average\_rating', ascending = False) return a.head(10)

df.columns

#Based upon Authors

@interact

def recomd\_books\_authors(authors\_name = list(df['authors'].value\_counts().index)): a = df[df['authors'] == authors\_name][['title', 'average\_rating']]

a = a.sort\_values(by = 'average\_rating', ascending = False) return a.head(10)

df.columns

@interact

def recomd\_books\_lang(language = list(df['language\_code'].value\_counts().index)): a = df[df['language\_code'] == language][['title', 'average\_rating']]

a = a.sort\_values(by = 'average\_rating', ascending = False) return a.head(10)

#Data Preprocessing

df.head(2)

df['average\_rating'] = pd.to\_numeric(df['average\_rating'], errors='coerce') df['rating\_obj'].value\_counts()

rating\_df = pd.get\_dummies(df['rating\_obj']) rating\_df.head()

df.columns language\_df =

pd.get\_dummies(df['language\_code']) language\_df.head()

features = pd.concat([rating\_df,language\_df, df['average\_rating'], df['ratings\_count'], df['title']], axis = 1)

features.set\_index('title', inplace= True) features.head()

from sklearn.preprocessing import MinMaxScaler scaler = MinMaxScaler() features\_scaled = scaler.fit\_transform(features) features\_scaled

#Model Building

from sklearn import neighbors

model = neighbors.NearestNeighbors(n\_neighbors=5, algorithm =

'ball\_tree', metric = 'euclidean')

dist, idlist = model.kneighbors(features\_scaled) df['title'].value\_counts() df['title'].value\_counts()

@interact

def BookRecommender(book\_name=list(df['title'].value\_counts().index)): book\_list\_name = []

book\_id = df[df['title'] == book\_name].index book\_id = book\_id[0] for newid in idlist[book\_id]:

book\_list\_name.append(df.iloc[newid]['title']) return book\_list\_name

# 

# 

# 

# 

# CHAPTER 8

**CONCLUSION AND FUTURE ENHANCEMENTS**

Here the data preprocessing, exploratory data analysis, and building a book

recommender system based on a given dataset is focused. The data preprocessing steps involved cleaning column names, handling missing values, and converting data types to ensure the dataset's integrity and usability. This stage is crucial for preparing the data for further analysis and modeling. The exploratory data analysis provided valuable insights into the dataset. Various visualizations and analyses were performed to understand book publication trends, popular authors and publishers, language distribution, and average ratings. Some interesting findings included the identification of the top 10 authors with the maximum number of books published, the distribution of books across different languages, the most occurring books in the dataset, and the histogram of average ratings. Additionally, popular publishers were determined based on the number of books published. Moreover, interactive functions to recommend books based on publishers, authors, and languages is implemented. Users could select a specific criterion, and the system would suggest the top 10 books based on average ratings. This feature aimed to assist users in discovering books of interest based on their preferences.

For model building, the k-nearest neighbors (KNN) algorithm was employed to create a book recommender system is used. The processed features, including one-hot encoded categorical variables and scaled numerical features, were used to train a NearestNeighbors model. This model could find similar books based on a provided book name, suggesting the top 5 nearest neighbor books using the Euclidean distance metric. This functionality aimed to assist users in finding similar books based on their preferences or a specific book they enjoyed. Overall, thedata analysis, visualization, and the development of a book recommender system encompassed. It provided valuable insights into the dataset's characteristics, explored various aspects of the books, and offered personalized book recommendations based on different criteria.

# REFERENCES

1. "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili, Packt Publishing Ltd, 2017.
2. Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. Springer, 2011.
3. "Building Machine Learning Systems with Python" by Willi Richert and Luis Pedro Coelho.
4. Building A Book Recommender System – The Basics, kNN and Matrix Factorization by Susan Li (Sep 26, 2017).
5. "Book Recommendation System using k-Nearest Neighbors Algorithm" - This tutorial on GeeksforGeeks explains the implementation of a book recommendation system using kNN.
6. "Building a Book Recommendation System with KNN and Python" - This tutorial on Medium provides a step-by-step guide to building a book recommendation system using kNN.
7. "Practical Recommender Systems" by Kim Falk.