The primary goal of this Machine Learning endeavor is to anticipate the rating of a book using a dataset extracted from the Goodreads website, an authentic community of readers.

Foreseeing the rating of a book can be framed as a regression challenge, given that it necessitates forecasting a continuous numerical value—specifically, the mean rating. As a result, employing a Supervised learning approach is apt, as it empowers us to forecast an uninterrupted output variable (the book's rating) grounded in one or more input variables (comprising titles, authors, page count, rating count, publishers, and more).

This examination will be segmented into three core segments:

1. Data Exploration
2. Data Visualization
3. Feature Engineering
4. Data Modeling
5. Conclusion

Project team's members:

- MAMADOU SALIOU DIALLO
- AGA Tangenssé Webana Julien
- ALIOU BA

Now, let's delve into the furnished dataset to glean further insights.

Data Exploration

```
! pip install xgboost
! pip install keras
! pip install tensorflow
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
```

```
Requirement already satisfied: xgboost in c:\users\s.diallo\anaconda3\lib\site-packages (1.7.6)
Requirement already satisfied: numpy in c:\users\s.diallo\anaconda3\lib\site-packages (from xgboost) (1.22.4)
Requirement already satisfied: scipy in c:\users\s.diallo\anaconda3\lib\site-packages (from xgboost) (1.6.2)
Requirement already satisfied: keras in c:\users\s.diallo\anaconda3\lib\site-packages (2.13.1)
Requirement already satisfied: tensorflow in c:\users\s.diallo\anaconda3\lib\site-packages (2.13.0)
Requirement already satisfied: tensorflow-intel==2.13.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow) (2.13.0)
Requirement already satisfied: setuptools in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorfl
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0-
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorfl
Requirement already satisfied: six>=1.12.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorf
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-in
Requirement already satisfied: astunparse>=1.6.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->t
Requirement already satisfied: h5py>=2.9.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorf
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\s.diall
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->t
Requirement already satisfied: keras<2.14,>=2.13.1 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0-
Requirement already satisfied: wrapt>=1.11.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tenso
Requirement already satisfied: tensorflow-estimator<2.14,>=2.13.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow
Requirement already satisfied: numpy<=1.24.3,>=1.22 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0
Requirement already satisfied: termcolor>=1.1.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->te
Requirement already satisfied: packaging in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorflo
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0-
Requirement already satisfied: libclang>=13.0.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->te
Requirement already satisfied: tensorboard<2.14,>=2.13 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.1
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0-
Requirement already satisfied: absl-py>=1.0.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tens
Requirement already satisfied: flatbuffers>=23.1.21 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from astunparse>=1.6.0->tensorf
Requirement already satisfied: markdown>=2.6.8 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorboard<2.14,>=2.13->tens
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorboard<2.14,>=2.13
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorboard<2.14,>=2.13->tens
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorboard<2.
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorboard<2.14,>=2.13->
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from tensorb
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\s.diallo\anaconda3\lib\site-packages (from google-auth<3,>=1.6.3->tensorbo
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from google-auth<3,>=1.6.3-
Requirement already satisfied: urllib3<2.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from google-auth<3,>=1.6.3->tensorboar
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\s.diallo\anaconda3\lib\site-packages (from google-auth<3,>=1.6.3->
```

```
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from google-auth-oauthlib
Requirement already satisfied: importlib-metadata>=4.4 in c:\users\s.diallo\anaconda3\lib\site-packages (from markdown>=2.6.8->tens
Requirement already satisfied: zipp>=0.5 in c:\users\s.diallo\anaconda3\lib\site-packages (from importlib-metadata>=4.4->markdown>=
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in c:\users\s.diallo\anaconda3\lib\site-packages (from pyasn1-modules>=0.2.1->g
Requirement already satisfied: idna<3,>=2.5 in c:\users\s.diallo\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorboard
Requirement already satisfied: certifi>=2017.4.17 in c:\users\s.diallo\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tenso
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\s.diallo\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\s.diallo\anaconda3\lib\site-packages (from requests-oauthlib>=0.7.0->goo
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\s.diallo\anaconda3\lib\site-packages (from packaging->tensorflow-intel=
```

```
'''%sql
SELECT * FROM `masterdata_p`.`default`.`books`;'''
pd_df1 = pd.read_csv(r"C:\Users\S.Diallo\Downloads\books.csv", error_bad_lines=False)
#pd_df1 = pd.read_csv(r"C:\Users\S.Diallo\Downloads\books.csv", sep = ';')
```

```
b'Skipping line 3350: expected 12 fields, saw 13\nSkipping line 4704: expected 12 fields, saw 13\nSkipping line 5879: expected 12 f
```

```
# DataFrame
pd_df1
```

	bookID	title	authors	average_rating	isbn	isbn13
0	1	Harry Potter and the Half-Blood	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969

```
pd_df1.head(10)
```

	bookID	title	authors	average_rating	isbn	isbn13	langu
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9780439554893	
3	5	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9780439655484	
4	8	Harry Potter Boxed Set Books 1-5 (Harry Potte...	J.K. Rowling/Mary GrandPré	4.78	0439682584	9780439682589	
5	9	Unauthorized Harry Potter Book Seven News: "Ha...	W. Frederick Zimmerman	3.74	0976540606	9780976540601	
6	10	Harry Potter Collection (Harry Potter #1-6)	J.K. Rowling	4.73	0439827604	9780439827607	
7	12	The Ultimate Hitchhiker's Guide: Five Complete...	Douglas Adams	4.38	0517226952	9780517226957	
8	13	The Ultimate Hitchhiker's Guide to the Galaxy ...	Douglas Adams	4.38	0345453743	9780345453747	
9	14	The Hitchhiker's Guide to the Galaxy (Hitchhik...	Douglas Adams	4.22	1400052920	9781400052929	

de Tom

```
pd_df1.tail(10)
```

	bookID	title	authors	average_rating	isbn	isbn13
11113	45617	O Cavalo e o Seu Rapaz (As Crónicas de Nárnia ...	C.S. Lewis/Pauline Baynes/Ana Falcão Bastos	3.92	9722330551	9789722330558
11114	45623	O Sobrinho do Mágico (As Crónicas de Nárnia #1)	C.S. Lewis/Pauline Baynes/Ana Falcão Bastos	4.04	9722329987	9789722329989
11115	45625	A Viagem do Caminheiro da Alvorada (As Crónica...	C.S. Lewis/Pauline Baynes/Ana Falcão Bastos	4.09	9722331329	9789722331326
11116	45626	O Príncipe Caspian (As Crónicas de Nárnia #4)	C.S. Lewis/Pauline Baynes/Ana Falcão Bastos	3.97	9722330977	9789722330978
11117	45630	Whores for Gloria	William T. Vollmann	3.69	0140231579	9780140231571
11118	45631	Expelled from Eden: A William T. Vollmann Reader	William T. Vollmann/Larry McCaffery/Michael He...	4.06	1560254416	9781560254416

```
pd_df1.describe()
```

	bookID	average_rating	isbn13	num_pages	ratings_count	text_r
count	11123.000000	11123.000000	1.112300e+04	11123.000000	1.112300e+04	
mean	21310.856963	3.934075	9.759880e+12	336.405556	1.794285e+04	
std	13094.727252	0.350485	4.429758e+11	241.152626	1.124992e+05	
min	1.000000	0.000000	8.987060e+09	0.000000	0.000000e+00	
25%	10277.500000	3.770000	9.780345e+12	192.000000	1.040000e+02	
50%	20287.000000	3.960000	9.780582e+12	299.000000	7.450000e+02	
75%	32104.500000	4.140000	9.780872e+12	416.000000	5.000500e+03	
max	45641.000000	5.000000	9.790008e+12	6576.000000	4.597666e+06	

```
print(pd_df1.columns)
```

```
Index(['bookID', 'title', 'authors', 'average_rating', 'isbn', 'isbn13',
       'language_code', '  num_pages', 'ratings_count', 'text_reviews_count',
       'publication_date', 'publisher'],
      dtype='object')
```

```
#Remove spaces at the beginning and end of column names
pd_df1.columns = pd_df1.columns.str.strip()
```

```
pd_df1.info()
```

```
<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
```

```
# Null values verification
pd_df1.isnull().sum()
```

```
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
```

There are no NA values.

```
# Check duplicates values
pd_df1.duplicated().any()
```

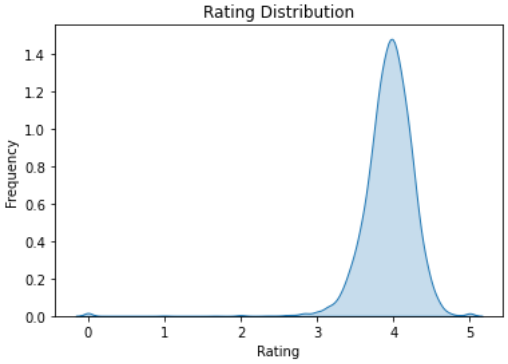
```
False
```

There are no duplicated values.

Data Visualization

```
# Average ratings distribution
import seaborn as sns
sns.kdeplot(pd_df1['average_rating'], fill = True)
plt.title('Rating Distribution')
plt.xlabel('Rating')
plt.ylabel('Frequency')
```

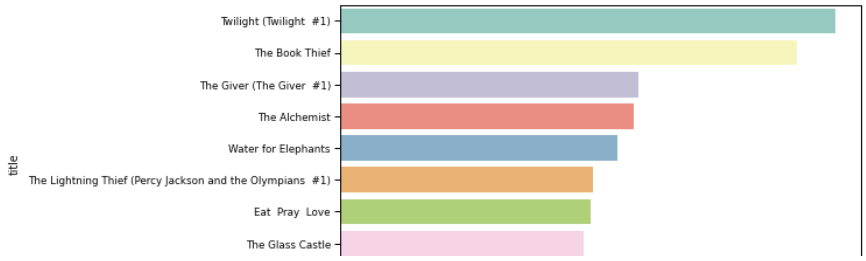
```
Text(0, 0.5, 'Frequency')
```



This plot provide a visual representation of how the average ratings are distributed. The KDE (Kernel Density Estimation) plot helps us understand the shape and concentration of data points along the rating scale. The x-axis represents different rating values and the y-axis represents the frequency of those ratings.

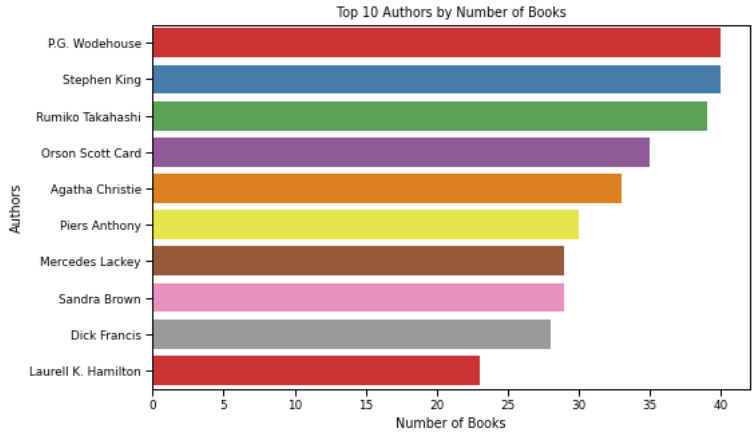
```
# Books with more written text reviews
most_reviews = pd_df1.sort_values('text_reviews_count', ascending = False).head(10).set_index('title')
plt.figure(figsize=(8,5))
sns.barplot(x=most_reviews['text_reviews_count'],y= most_reviews.index, palette='Set3')
```

```
<AxesSubplot:xlabel='text_reviews_count', ylabel='title'>
```



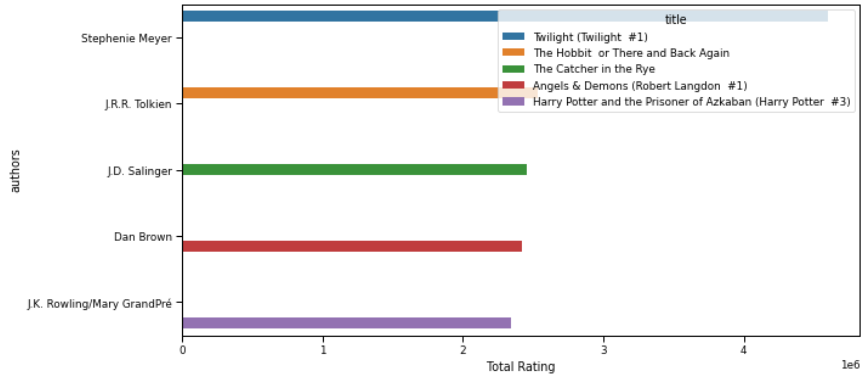
This plot gives us a quick overview of which books are generating the most text reviews, which could be an indicator of their popularity or engagement among readers. We visualize the top 10 books with the highest number of text reviews. Each bar in the plot represents a book, and its length indicates the number of text reviews that particular book has received. The longer the bar, the more text reviews the book has garnered.

```
# Let's visualize the top 10 authors in our dataset based on the number of books they have authored.
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 5))
plot = sns.countplot(
    y="authors",
    data=pd_df1,
    order=pd_df1["authors"].value_counts().iloc[:10].index,
    palette="Set1"
)
plt.xlabel("Number of Books")
plt.ylabel("Authors")
plt.title("Top 10 Authors by Number of Books")
plt.show()
```




```
# Let's identify the authors whose books have received the highest ratings count.
plt.figure(figsize=(10, 5))
a = pd_df1.nlargest(5, ['ratings_count']).set_index('authors')
sns.barplot(x=a['ratings_count'], y=a.index, ci = None, hue = a['title'])
plt.xlabel('Total Rating')
```

```
Text(0.5, 0, 'Total Rating')
```

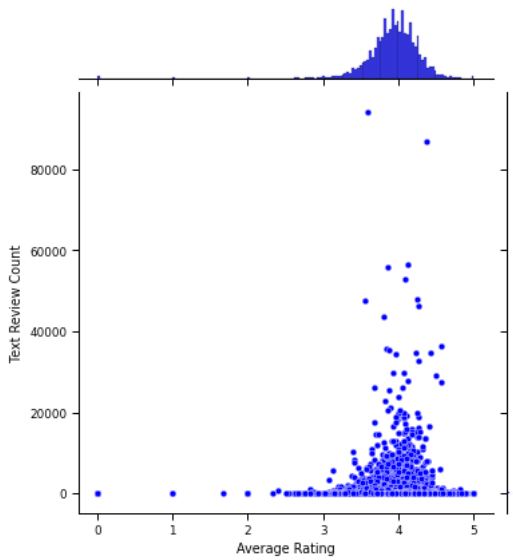


Here a meaningful plot that highlights the top authors based on the total ratings count for their books. With this visualization, we can better understand which authors have successfully captured readers' interest, as well as the specific books that have contributed significantly to their high ratings counts.

Overall, this plot provides a comprehensive overview of the authors who have achieved the highest levels of engagement and recognition from readers.

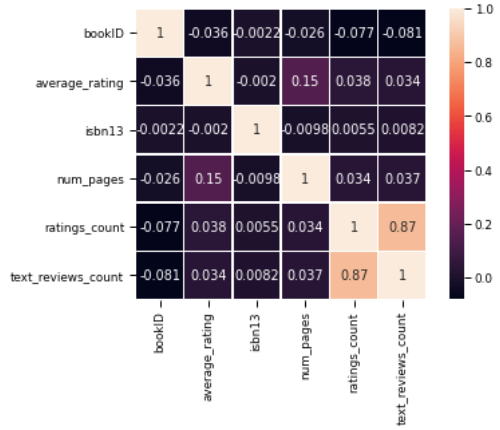
```
# Distribution between Rating and Text Reviews

plt.figure(figsize=(8,5))
pd_df1.dropna(0, inplace=True)
sns.set_context('paper')
ax = sns.jointplot(x="average_rating", y="text_reviews_count", kind="scatter", data=pd_df1[['text_reviews_count', 'average_rating']], col
ax.set_axis_labels("Average Rating", "Text Review Count")
plt.show()
```

```
<Figure size 576x360 with 0 Axes>
```




```
sns.heatmap(data=pd_df1.corr(),
linewidths=0.5, square=True,
linecolor="white", annot=True)
```

```
<AxesSubplot:>
```



This insightful heatmap plot provide a visual representation of the correlations between different numerical variables in our dataset. We see a high correlation between the ratings_count and the text_reviews_count around 87%.

```
# Let's take a look at the 10 top-rated books.
top_rated = pd_df1.sort_values(by="ratings_count", ascending = False).head(10)
top_rated_titles = pd.DataFrame(top_rated.title).join(pd.DataFrame(top_rated.ratings_count))
top_rated_titles
```

	title	ratings_count
10336	Twilight (Twilight #1)	4597666
1697	The Hobbit or There and Back Again	2530894
1462	The Catcher in the Rye	2457092
307	Angels & Demons (Robert Langdon #1)	2418736
3	Harry Potter and the Prisoner of Azkaban (Harr...	2339585
4415	Harry Potter and the Chamber of Secrets (Harry...	2293963
1	Harry Potter and the Order of the Phoenix (Har...	2153167
23	The Fellowship of the Ring (The Lord of the Ri...	2128944

Feature Engineering

```
#Select the relevant columns for linear regression
selected_columns = ['authors', 'language_code', 'num_pages', 'ratings_count', 'text_reviews_count', 'average_rating', 'publication_date']


#Create a DataFrame with the selected columns from pd_df1
pd_df1_selected = pd_df1[selected_columns]



#Convert the 'publication_date' column to date format using various formats
pd_df1_selected['publication_date'] = pd.to_datetime(pd_df1_selected['publication_date'], errors='coerce', format='%m/%d/%Y')
pd_df1_selected['publication_month'] = pd_df1_selected['publication_date'].dt.month
pd_df1_selected['publication_day'] = pd_df1_selected['publication_date'].dt.day
pd_df1_selected['publication_year'] = pd_df1_selected['publication_date'].dt.year


#Display the updated DataFrame
print(pd_df1_selected)
```

```
11118  William T. Vollmann/Larry McCaffery/Michael He...          eng
11119                              William T. Vollmann          eng
11120                              William T. Vollmann          eng
11121                              William T. Vollmann          eng
11122                                       Mark Twain          spa

       num_pages  ratings_count  text_reviews_count  average_rating  \
0            652        2095690               27591            4.57
1            870        2153167               29221            4.49
2            352           6333                 244            4.42
3            435        2339585               36325            4.56
4           2690          41428                 164            4.78
...          ...            ...                 ...             ...
11118        512            156                  20            4.06
11119        635            783                  56            4.08
11120        415            820                  95            3.96
11121        434            769                 139            3.72
11122        272            113                  12            3.91

      publication_date  publication_month  publication_day  publication_year
0           2006-09-16                9.0             16.0            2006.0
1           2004-09-01                9.0              1.0            2004.0
2           2003-11-01               11.0              1.0            2003.0
3           2004-05-01                5.0              1.0            2004.0
4           2004-09-13                9.0             13.0            2004.0
...                ...                ...              ...               ...
11118       2004-12-21               12.0             21.0            2004.0
11119       1988-12-01               12.0              1.0            1988.0
11120       1993-08-01                8.0              1.0            1993.0
11121       2007-02-27                2.0             27.0            2007.0
11122       2006-05-28                5.0             28.0            2006.0

[11123 rows x 10 columns]
<ipython-input-208-adfa2385ffde>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
  pd_df1_selected['publication_date'] = pd.to_datetime(pd_df1_selected['publication_date'], errors='coerce', format='%m/%d/%Y')
<ipython-input-208-adfa2385ffde>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
  pd_df1_selected['publication_day'] = pd_df1_selected['publication_date'].dt.day
<ipython-input-208-adfa2385ffde>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
  pd_df1_selected['publication_year'] = pd_df1_selected['publication_date'].dt.year
```

```
pd_df1_selected['publication_month'] = pd_df1_selected['publication_month'].fillna(0).astype(int)
pd_df1_selected['publication_day'] = pd_df1_selected['publication_day'].fillna(0).astype(int)
pd_df1_selected['publication_year'] = pd_df1_selected['publication_year'].fillna(0).astype(int)
```

```
<ipython-input-20-0da2699fcd8b>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  pd_df1_selected['publication_month'] = pd_df1_selected['publication_month'].fillna(0).astype(int)
<ipython-input-20-0da2699fcd8b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  pd_df1_selected['publication_day'] = pd_df1_selected['publication_day'].fillna(0).astype(int)
<ipython-input-20-0da2699fcd8b>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  pd_df1_selected['publication_year'] = pd_df1_selected['publication_year'].fillna(0).astype(int)
```

```
pd_df1_selected
```

	authors	language_code	num_pages	ratings_count	text_reviews_count
0	J.K. Rowling/Mary GrandPré	eng	652	2095690	27591
1	J.K. Rowling/Mary GrandPré	eng	870	2153167	29221
2	J.K. Rowling	eng	352	6333	244
3	J.K. Rowling/Mary GrandPré	eng	435	2339585	36325
4	J.K. Rowling/Mary GrandPré	eng	2690	41428	164
...	...	...	...	...	...
11118	William T. Vollmann/Larry McCaffery/Michael He...	eng	512	156	20
11119	William T. Vollmann	eng	635	783	56
11120	William T. Vollmann	eng	415	820	95
11121	William T. Vollmann	eng	434	769	139
11122	Mark Twain	spa	272	113	12

11123 rows × 10 columns

```
pd_df1_selected.drop('publication_date', axis=1, inplace=True)
```

```
C:\Users\S.Diallo\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  return super().drop(
```

```
print(pd_df1_selected)
```

```
                                          authors language_code  \
0                          J.K. Rowling/Mary GrandPré           eng
1                          J.K. Rowling/Mary GrandPré           eng
2                                        J.K. Rowling           eng
3                          J.K. Rowling/Mary GrandPré           eng
4                          J.K. Rowling/Mary GrandPré           eng
...                                               ...           ...
11118  William T. Vollmann/Larry McCaffery/Michael He...           eng
11119                            William T. Vollmann           eng
11120                            William T. Vollmann           eng
11121                            William T. Vollmann           eng
11122                                     Mark Twain           spa

       num_pages  ratings_count  text_reviews_count  average_rating  \
0            652        2095690               27591            4.57
1            870        2153167               29221            4.49
2            352           6333                 244            4.42
3            435        2339585               36325            4.56
4           2690          41428                 164            4.78
...          ...            ...                 ...             ...
11118        512            156                  20            4.06
11119        635            783                  56            4.08
11120        415            820                  95            3.96
11121        434            769                 139            3.72
11122        272            113                  12            3.91

       publication_month  publication_day  publication_year
0                      9               16              2006
1                      9                1              2004
2                     11                1              2003
3                      5                1              2004
4                      9               13              2004
...                  ...              ...               ...
11118                 12               21              2004
11119                 12                1              1988
11120                  8                1              1993
11121                  2               27              2007
11122                  5               28              2006

[11123 rows x 9 columns]
```

```
#Perform one-hot encoding for the 'language_code' column
pd_df1_encoded = pd.get_dummies(pd_df1_selected, columns=['language_code'], prefix_sep='')
print(pd_df1_encoded)
```

```
11119           783                  56           4.08                0
11120           820                  95           3.96                0
11121           769                 139           3.72                0
11122           113                  12           3.91                0

       language_codeara  language_codeen-CA  language_codeen-GB  \
0                     0                   0                   0
1                     0                   0                   0
2                     0                   0                   0
3                     0                   0                   0
4                     0                   0                   0
...                 ...                 ...                 ...
11118                 0                   0                   0
11119                 0                   0                   0
11120                 0                   0                   0
11121                 0                   0                   0
11122                 0                   0                   0

       language_codeen-US  ...  language_codenl  language_codenor  \
0                       0  ...                0                 0
1                       0  ...                0                 0
```

```
11122                 0                 0                 1                 0

       language_codeswe  language_codetur  language_codewel  language_codezho
0                     0                 0                 0                 0
1                     0                 0                 0                 0
2                     0                 0                 0                 0
3                     0                 0                 0                 0
4                     0                 0                 0                 0
...                 ...               ...               ...               ...
11118                 0                 0                 0                 0
11119                 0                 0                 0                 0
11120                 0                 0                 0                 0
11121                 0                 0                 0                 0
11122                 0                 0                 0                 0

[11123 rows x 32 columns]
```

```
print(pd_df1_encoded.columns)
```

```
Index(['authors', 'num_pages', 'ratings_count', 'text_reviews_count',
       'average_rating', 'publication_month', 'publication_day',
       'publication_year', 'language_codeale', 'language_codeara',
       'language_codeen-CA', 'language_codeen-GB', 'language_codeen-US',
       'language_codeeng', 'language_codeenm', 'language_codefre',
       'language_codeger', 'language_codegla', 'language_codeglg',
       'language_codegrc', 'language_codeita', 'language_codejpn',
       'language_codelat', 'language_codemsa', 'language_codemul',
       'language_codenl', 'language_codenor', 'language_codepor',
       'language_coderus', 'language_codespa', 'language_codesrp',
       'language_codeswe', 'language_codetur', 'language_codewel',
       'language_codezho'],
      dtype='object')
```

```
pd_df1_encoded
```

	authors	num_pages	ratings_count	text_reviews_count	average_rating
0	J.K. Rowling/Mary GrandPré	652	2095690	27591	4.57
1	J.K. Rowling/Mary GrandPré	870	2153167	29221	4.49
2	J.K. Rowling	352	6333	244	4.42
3	J.K. Rowling/Mary GrandPré	435	2339585	36325	4.56
4	J.K. Rowling/Mary GrandPré	2690	41428	164	4.78
...	...	...	...	...	...
11118	William T. Vollmann/Larry McCaffery/Michael He...	512	156	20	4.06
11119	William T. Vollmann	635	783	56	4.08
11120	William T. Vollmann	415	820	95	3.96
11121	William T. Vollmann	434	769	139	3.72
11122	Mark Twain	272	113	12	3.91

11123 rows × 32 columns

```
#Create a LabelEncoder for the 'authors' column
author_encoder = LabelEncoder()
pd_df1_encoded['authors_encoded'] = author_encoder.fit_transform(pd_df1_selected['authors'])
```

```
print(pd_df1_encoded.columns)
```

```
Index(['num_pages', 'ratings_count', 'text_reviews_count', 'language_codeale',
       'language_codeara', 'language_codeen-CA', 'language_codeen-GB',
       'language_codeen-US', 'language_codeeng', 'language_codeenm',
       'language_codefre', 'language_codeger', 'language_codegla',
       'language_codeglg', 'language_codegrc', 'language_codeita',
       'language_codejpn', 'language_codelat', 'language_codemsa',
       'language_codemul', 'language_codenl', 'language_codenor',
       'language_codepor', 'language_coderus', 'language_codespa',
```

```
       'language_codesrp', 'language_codeswe', 'language_codetur',
       'language_codewel', 'language_codezho', 'authors_encoded',
       'average_rating_encoded'],
      dtype='object')
```

```
pd_df1_selected
```

	authors	language_code	num_pages	ratings_count	text_reviews_count
0	J.K. Rowling/Mary GrandPré	eng	652	2095690	27591
1	J.K. Rowling/Mary GrandPré	eng	870	2153167	29221
2	J.K. Rowling	eng	352	6333	244
3	J.K. Rowling/Mary GrandPré	eng	435	2339585	36325
4	J.K. Rowling/Mary GrandPré	eng	2690	41428	164
...	...	...	...	...	...
11118	William T. Vollmann/Larry McCaffery/Michael He...	eng	512	156	20
11119	William T. Vollmann	eng	635	783	56
11120	William T. Vollmann	eng	415	820	95
11121	William T. Vollmann	eng	434	769	139
11122	Mark Twain	spa	272	113	12

11123 rows × 10 columns

```
pd_df1_encoded
```

	num_pages	ratings_count	text_reviews_count	language_codeale	language_codea
0	0.099148	0.455816	0.292696	0	
1	0.132299	0.468317	0.309988	0	
2	0.053528	0.001377	0.002588	0	
3	0.066150	0.508864	0.385350	0	
4	0.409063	0.009011	0.001740	0	
...	...	...	...	...	
11118	0.077859	0.000034	0.000212	0	
11119	0.096563	0.000170	0.000594	0	
11120	0.063108	0.000178	0.001008	0	
11121	0.065998	0.000167	0.001475	0	
11122	0.041363	0.000025	0.000127	0	

11123 rows × 32 columns

```
print(pd_df1_encoded.columns)
```

```
Index(['num_pages', 'ratings_count', 'text_reviews_count', 'language_codeale',
       'language_codeara', 'language_codeen-CA', 'language_codeen-GB',
       'language_codeen-US', 'language_codeeng', 'language_codeenm',
       'language_codefre', 'language_codeger', 'language_codegla',
       'language_codeglg', 'language_codegrc', 'language_codeita',
       'language_codejpn', 'language_codelat', 'language_codemsa',
       'language_codemul', 'language_codenl', 'language_codenor',
       'language_codepor', 'language_coderus', 'language_codespa',
       'language_codesrp', 'language_codeswe', 'language_codetur',
       'language_codewel', 'language_codezho', 'authors_encoded',
       'average_rating_encoded'],
      dtype='object')
```

```
from sklearn.preprocessing import LabelEncoder

# Create a label encoder
label_encoder = LabelEncoder()

# Function to encode non-numeric elements and leave numeric ones untouched
def encode_average_rating(value):
    try:
        return float(value)  # If it's a number, return the value directly as a float
    except ValueError:  # If conversion to float fails, it's a non-numeric string
        return label_encoder.fit_transform([value])[0]

# Apply the function to the 'average_rating' column to encode the values
pd_df1_encoded['average_rating_encoded'] = pd_df1_encoded['average_rating_encoded'].apply(encode_average_rating)

# Display the DataFrame with the encoded numeric values
print(pd_df1_encoded)
```

```
11119                0                   0                   0
11120                0                   0                   0
11121                0                   0                   0
11122                0                   0                   0

       language_codeen-US  language_codeeng  language_codeenm  ...  \
0                       0                 1                 0  ...
1                       0                 1                 0  ...
2                       0                 1                 0  ...
3                       0                 1                 0  ...
4                       0                 1                 0  ...
...                   ...               ...               ...  ...
11118                   0                 1                 0  ...
11119                   0                 1                 0  ...
11120                   0                 1                 0  ...
11121                   0                 1                 0  ...
11122                   0                 0                 0  ...

       language_codepor  language_coderus  language_codespa  language_codesrp  \
0                     0                 0                 0                 0
1                     0                 0                 0                 0
2                     0                 0                 0                 0
3                     0                 0                 0                 0
4                     0                 0                 0                 0
...                 ...               ...               ...               ...
11118                 0                 0                 0                 0
11119                 0                 0                 0                 0
11120                 0                 0                 0                 0
11121                 0                 0                 0                 0
11122                 0                 0                 1                 0

       language_codeswe  language_codetur  language_codewel  language_codezho  \
0                     0                 0                 0                 0
1                     0                 0                 0                 0
2                     0                 0                 0                 0
3                     0                 0                 0                 0
4                     0                 0                 0                 0
...                 ...               ...               ...               ...
11118                 0                 0                 0                 0
11119                 0                 0                 0                 0
11120                 0                 0                 0                 0
11121                 0                 0                 0                 0
11122                 0                 0                 0                 0

       authors_encoded  average_rating_encoded
0             0.395601                    4.57
1             0.395601                    4.49
2             0.394998                    4.42
3             0.395601                    4.56
4             0.395601                    4.78
...                ...                     ...
11118         0.987647                    4.06
11119         0.987496                    4.08
11120         0.987496                    3.96
11121         0.987496                    3.72
11122         0.620066                    3.91

[11123 rows x 32 columns]
```

```
pd_df1_encoded
```

	num_pages	ratings_count	text_reviews_count	language_codeale	language_codea
0	0.099148	0.455816	0.292696	0	
1	0.132299	0.468317	0.309988	0	
2	0.053528	0.001377	0.002588	0	
3	0.066150	0.508864	0.385350	0	
4	0.409063	0.009011	0.001740	0	
...	...	...	...	...	
11118	0.077859	0.000034	0.000212	0	
11119	0.096563	0.000170	0.000594	0	
11120	0.063108	0.000178	0.001008	0	

```
# Replace empty strings with null values (NaN)
pd_df1_encoded['num_pages'].replace('', pd.NA, inplace=True)

# Convert the column to numeric type
pd_df1_encoded['num_pages'] = pd.to_numeric(pd_df1_encoded['num_pages'], errors='coerce')

# Display the DataFrame with the 'num_pages' column converted to numeric type
print(pd_df1_encoded)
```

```
11119                0                  0                  0
11120                0                  0                  0
11121                0                  0                  0
11122                0                  0                  0

       language_codeen-US  language_codeeng  language_codeenm  ...  \
0                       0                 1                 0  ...
1                       0                 1                 0  ...
2                       0                 1                 0  ...
3                       0                 1                 0  ...
4                       0                 1                 0  ...
...                   ...               ...               ...  ...
11118                   0                 1                 0  ...
11119                   0                 1                 0  ...
11120                   0                 1                 0  ...
11121                   0                 1                 0  ...
11122                   0                 0                 0  ...

       language_codepor  language_coderus  language_codespa  language_codesrp  \
0                     0                 0                 0                 0
1                     0                 0                 0                 0
2                     0                 0                 0                 0
3                     0                 0                 0                 0
4                     0                 0                 0                 0
...                 ...               ...               ...               ...
11118                 0                 0                 0                 0
11119                 0                 0                 0                 0
11120                 0                 0                 0                 0
11121                 0                 0                 0                 0
11122                 0                 0                 1                 0

       language_codeswe  language_codetur  language_codewel  language_codezho  \
0                     0                 0                 0                 0
1                     0                 0                 0                 0
2                     0                 0                 0                 0
```

```
[11123 rows x 32 columns]
```

```
# Display the data types of columns in the pd_df1_encoded DataFrame
print(pd_df1_encoded.dtypes)
```

```
num_pages                float64
ratings_count            float64
text_reviews_count       float64
language_codeale           uint8
language_codeara           uint8
language_codeen-CA         uint8
language_codeen-GB         uint8
language_codeen-US         uint8
language_codeeng           uint8
language_codeenm           uint8
language_codefre           uint8
language_codeger           uint8
language_codegla           uint8
language_codeglg           uint8
language_codegrc           uint8
language_codeita           uint8
language_codejpn           uint8
language_codelat           uint8
language_codemsa           uint8
language_codemul           uint8
language_codenl            uint8
language_codenor           uint8
language_codepor           uint8
language_coderus           uint8
language_codespa           uint8
language_codesrp           uint8
language_codeswe           uint8
language_codetur           uint8
language_codewel           uint8
language_codezho           uint8
authors_encoded          float64
average_rating_encoded   float64
dtype: object
```

```
from sklearn.preprocessing import MinMaxScaler

# Create a MinMaxScaler object
scaler = MinMaxScaler()

# Select columns to normalize
columns_to_normalize = ['num_pages', 'ratings_count', 'text_reviews_count', 'authors_encoded']

# Apply normalization to the selected columns
pd_df1_encoded[columns_to_normalize] = scaler.fit_transform(pd_df1_encoded[columns_to_normalize])

# Display the DataFrame with normalized columns
print(pd_df1_encoded)
```

```
       num_pages  ratings_count  text_reviews_count  language_codeale  \
0       0.099148       0.455816            0.292696                 0
1       0.132299       0.468317            0.309988                 0
2       0.053528       0.001377            0.002588                 0
3       0.066150       0.508864            0.385350                 0
4       0.409063       0.009011            0.001740                 0
...          ...            ...                 ...               ...
11118   0.077859       0.000034            0.000212                 0
11119   0.096563       0.000170            0.000594                 0
11120   0.063108       0.000178            0.001008                 0
11121   0.065998       0.000167            0.001475                 0
11122   0.041363       0.000025            0.000127                 0

       language_codeara  language_codeen-CA  language_codeen-GB  \
0                     0                   0                   0
1                     0                   0                   0
2                     0                   0                   0
3                     0                   0                   0
4                     0                   0                   0
...                 ...                 ...                 ...
11118                 0                   0                   0
11119                 0                   0                   0
11120                 0                   0                   0
11121                 0                   0                   0
11122                 0                   0                   0

       language_codeen-US  language_codeeng  language_codeenm  ...  \
0                       0                 1                 0  ...
1                       0                 1                 0  ...
```

```
2                     0                1                0  ...
3                     0                1                0  ...
4                     0                1                0  ...
...                 ...              ...              ...  ...
11118                 0                1                0  ...
11119                 0                1                0  ...
11120                 0                1                0  ...
11121                 0                1                0  ...
11122                 0                0                0  ...

       language_codepor  language_coderus  language_codespa  language_codesrp  \
0                     0                 0                 0                 0
1                     0                 0                 0                 0
2                     0                 0                 0                 0
3                     0                 0                 0                 0
4                     0                 0                 0                 0
...                 ...               ...               ...               ...
11118                 0                 0                 0                 0
11119                 0                 0                 0                 0
11120                 0                 0                 0                 0
11121                 0                 0                 0                 0
11122                 0                 0                 1                 0

       language_codeswe  language_codetur  language_codewel  language_codezho  \
0                     0                 0                 0                 0
1                     0                 0                 0                 0
2                     0                 0                 0                 0
3                     0                 0                 0                 0
```

```
correlation_matrix = pd_df1_encoded.corr()
correlation_with_target = correlation_matrix['average_rating_encoded'].sort_values(ascending=False)
print(correlation_with_target)
```

```
average_rating_encoded    1.000000
num_pages                 0.150477
language_codejpn          0.061528
language_codezho          0.052910
ratings_count             0.038224
text_reviews_count        0.033663
language_codewel          0.028839
language_codemul          0.022690
authors_encoded           0.021078
language_codelat          0.019649
language_codegla          0.014500
language_codetur          0.013147
language_coderus          0.012280
language_codefre          0.012239
language_codeale          0.011524
language_codeita          0.008709
language_codenl           0.006654
language_codeen-CA        0.006562
language_codemsa          0.004760
language_codeger          0.004333
language_codepor          0.000935
language_codeeng         -0.000079
language_codespa         -0.001922
language_codeenm         -0.002847
language_codeen-GB       -0.004262
language_codenor         -0.009039
language_codeara         -0.010391
language_codeglg         -0.015532
language_codeswe         -0.018331
language_codegrc         -0.020361
language_codeen-US       -0.021091
language_codesrp         -0.106439
Name: average_rating_encoded, dtype: float64
```

```
# Display the elements of the target variable 'average_rating_encoded'
print(pd_df1_encoded['average_rating_encoded'])
```

```
0        4.57
1        4.49
2        4.42
3        4.56
4        4.78
         ...
11118    4.06
11119    4.08
11120    3.96
11121    3.72
11122    3.91
Name: average_rating_encoded, Length: 11123, dtype: float64
```

```
print(pd_df1_encoded)
print(pd_df1_encoded.columns)
print(pd_df1_encoded['language_codeeng'])
```

```
...                ...              ...              ...              ...
11118                0                0                0                0
11119                0                0                0                0
11120                0                0                0                0
11121                0                0                0                0
11122                0                0                1                0

       language_codeswe  language_codetur  language_codewel  language_codezho  \
0                     0                 0                 0                 0
1                     0                 0                 0                 0
2                     0                 0                 0                 0
3                     0                 0                 0                 0
4                     0                 0                 0                 0
...                 ...               ...               ...               ...
11118                 0                 0                 0                 0
11119                 0                 0                 0                 0
11120                 0                 0                 0                 0
11121                 0                 0                 0                 0
11122                 0                 0                 0                 0

       authors_encoded  average_rating_encoded
0             0.395601                    4.57
1             0.395601                    4.49
2             0.394998                    4.42
3             0.395601                    4.56
4             0.395601                    4.78
...                ...                     ...
11118         0.987647                    4.06
11119         0.987496                    4.08
11120         0.987496                    3.96
11121         0.987496                    3.72
11122         0.620066                    3.91

[11123 rows x 32 columns]
Index(['num_pages', 'ratings_count', 'text_reviews_count', 'language_codeale',
       'language_codeara', 'language_codeen-CA', 'language_codeen-GB',
       'language_codeen-US', 'language_codeeng', 'language_codeenm',
       'language_codefre', 'language_codeger', 'language_codegla',
       'language_codeglg', 'language_codegrc', 'language_codeita',
       'language_codejpn', 'language_codelat', 'language_codemsa',
       'language_codemul', 'language_codenl', 'language_codenor',
       'language_codepor', 'language_coderus', 'language_codespa',
       'language_codesrp', 'language_codeswe', 'language_codetur',
       'language_codewel', 'language_codezho', 'authors_encoded',
       'average_rating_encoded'],
      dtype='object')
0        1
1        1
2        1
3        1
4        1
        ..
11118    1
11119    1
11120    1
11121    1
11122    0
Name: language_codeeng, Length: 11123, dtype: uint8
```

```
# Display the data types of columns in the DataFrame pd_df1_encoded
print(pd_df1_encoded.dtypes)
```

```
num_pages                float64
ratings_count            float64
text_reviews_count       float64
language_codeale           uint8
language_codeara           uint8
language_codeen-CA         uint8
language_codeen-GB         uint8
language_codeen-US         uint8
language_codeeng           uint8
language_codeenm           uint8
language_codefre           uint8
language_codeger           uint8
language_codegla           uint8
language_codeglg           uint8
language_codegrc           uint8
language_codeita           uint8
language_codejpn           uint8
language_codelat           uint8
language_codemsa           uint8
language_codemul           uint8
language_codenl            uint8
```

```
language_codenor            uint8
language_codepor            uint8
language_coderus            uint8
language_codespa            uint8
language_codesrp            uint8
language_codeswe            uint8
language_codetur            uint8
language_codewel            uint8
language_codezho            uint8
authors_encoded           float64
average_rating_encoded    float64
dtype: object
```

```
# Display the content of the 'num_pages' column in the DataFrame pd_df1_encoded
print(pd_df1_encoded['num_pages'])
```

```
0        0.099148
1        0.132299
2        0.053528
3        0.066150
4        0.409063
           ...
11118    0.077859
11119    0.096563
11120    0.063108
11121    0.065998
11122    0.041363
Name: num_pages, Length: 11123, dtype: float64
```

Data Modeling

```
# Split the data into independent variables (X) and the target variable (y)
X = pd_df1_encoded.drop('average_rating_encoded', axis=1)  # Independent variables
y = pd_df1_encoded['average_rating_encoded']  # Target variable
```

```
print(X)
```

```
11119                0                   0                   0
11120                0                   0                   0
11121                0                   0                   0
11122                0                   0                   0

       language_codeen-US  language_codeeng  language_codeenm  ...  \
0                       0                 1                 0  ...
1                       0                 1                 0  ...
2                       0                 1                 0  ...
3                       0                 1                 0  ...
4                       0                 1                 0  ...
...                   ...               ...               ...  ...
11118                   0                 1                 0  ...
11119                   0                 1                 0  ...
11120                   0                 1                 0  ...
11121                   0                 1                 0  ...
11122                   0                 0                 0  ...

       language_codenor  language_codepor  language_coderus  language_codespa  \
0                     0                 0                 0                 0
1                     0                 0                 0                 0
2                     0                 0                 0                 0
3                     0                 0                 0                 0
```

```
      language_codezno  authors_encoded
0                    0         0.395601
1                    0         0.395601
2                    0         0.394998
3                    0         0.395601
4                    0         0.395601
...                ...              ...
11118                0         0.987647
11119                0         0.987496
11120                0         0.987496
11121                0         0.987496
11122                0         0.620066

[11123 rows x 31 columns]
```

```
y = pd_df1_encoded['average_rating_encoded']
print(y)
```

```
0        4.57
1        4.49
2        4.42
3        4.56
4        4.78
         ...
11118    4.06
11119    4.08
11120    3.96
11121    3.72
11122    3.91
Name: average_rating_encoded, Length: 11123, dtype: float64
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Instead of this (might trigger the warning):
# pd_df1_encoded['num_pages'].replace('', pd.NA, inplace=True)

# Use .loc to modify the DataFrame directly:
pd_df1_encoded.loc[pd_df1_encoded['num_pages'] == '', 'num_pages'] = pd.NA

X_train.fillna(X_train.mean(), inplace=True)
y_train.fillna(y_train.mean(), inplace=True)
```

```
import numpy as np
from sklearn.metrics import mean_squared_error

# Create the linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)

# Calculate the root mean squared error (RMSE)
rmse = np.sqrt(mse)

# Print the RMSE
print("Root Mean Squared Error:", rmse)
```

```
Root Mean Squared Error: 30877019095.418633
```

```
import numpy as np
from sklearn.metrics import mean_squared_error

# Create the decision tree model
tree_model = DecisionTreeRegressor(random_state=42)

# Train the model on the training data
tree_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred_tree = tree_model.predict(X_test)
```

```
# Calculate the mean squared error
mse_tree = mean_squared_error(y_test, y_pred_tree)

# Calculate the root mean squared error (RMSE)
rmse_tree = np.sqrt(mse_tree)

# Print the RMSE
print("Root Mean Squared Error for Decision Tree:", rmse_tree)
```

```
Root Mean Squared Error for Decision Tree: 0.5086307240849162
```

```
import numpy as np
from sklearn.metrics import mean_squared_error

# Create the random forest model
random_forest_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the random forest model on the training data
random_forest_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred_rf = random_forest_model.predict(X_test)

# Calculate the mean squared error
mse_rf = mean_squared_error(y_test, y_pred_rf)

# Calculate the root mean squared error (RMSE)
rmse_rf = np.sqrt(mse_rf)

# Print the RMSE
print("Root Mean Squared Error for Random Forest:", rmse_rf)
```

```
Root Mean Squared Error for Random Forest: 0.34318014997365454
```

```
import numpy as np
from sklearn.metrics import mean_squared_error

# Create the XGBoost model
xgb_model = xgb.XGBRegressor(n_estimators=100, random_state=42)

# Train the XGBoost model on the training data
xgb_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred_xgb = xgb_model.predict(X_test)

# Calculate the mean squared error
mse_xgb = mean_squared_error(y_test, y_pred_xgb)

# Calculate the root mean squared error (RMSE)
rmse_xgb = np.sqrt(mse_xgb)

# Print the RMSE
print("Root Mean Squared Error for XGBoost:", rmse_xgb)
```

```
Root Mean Squared Error for XGBoost: 0.3436460500308098
```

Conclusion

In wrapping up our Book Ratings Prediction project and summarizing the insights we've gathered, we find ourselves favoring the Random Forest and XGBoost models due to their lower RMSE values. These models have shown promise in accurately predicting book ratings, marking a notable stride in our analytical journey. As we conclude this endeavor, we stand poised to leverage these insights for informed decision-making and further advancements in the realm of book rating predictions.