Projet_Python_Rating_Book_MM

December 18, 2022

1 Projet Python : Predicts a book's rating

The project consists of predicting the coast of a book through a dataset. For this, we must use the "machine learning" system. This makes it possible to apply predictive analysis algorithms to different types of data in order to predict the future.

In python, we need to import several libraries:

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

To meet our need, we have to go through different steps: - Data analysis: data processing, data cleaning, exploratory analysis and relevant graphics - "Feature selection": Engineering of characteristics, pruning of characteristics and justification of choices; - "Model Training": Model selection and justification and comparison with other models; - "Model evaluation": Measurement and interpretation of results; - Project Report

1.1 Import Data with CSV

First, we have to import the books.csv file into a dataframe using the `read_csv` function of the `pandas` library.

[2]:
```
dfBook = pd.read_csv("books.csv",sep=",",index_col="bookID",␣
↪skipinitialspace=True, on_bad_lines='warn')
```

```
b'Skipping line 3350: expected 12 fields, saw 13\nSkipping line 4704: expected
12 fields, saw 13\nSkipping line 5879: expected 12 fields, saw 13\nSkipping line
8981: expected 12 fields, saw 13\n'
```

> Remark: There is a data problem in the csv file due to empty fields or poorly constructed fields. To solve this problem, we must specify that lines that encounter a problem must trigger a warning and move on to the next one.
>
> Remark: The first parameter corresponds to the name of our file that we want to import. The sep parameter specifies the delimiter to use, in our case, it is a comma that separates the different fields in the file. The `index_col`parameter corresponds to

the column to be used as the row labels of the `DataFrame`, this corresponds to an id most of the time. The optional parameter `skipinitialspace` allows you to remove spaces after a delimiter, this one waits for a boolean (TRUE/FALSE), we noticed that in the dataset, there were spaces after the comma and this was a problem in the structure of the `DataFrame`.

Selection of the first 30 lines:

[3]:
```
dfBook.head(30)
```

[3]:

bookID	title	authors	average_rating
1	Harry Potter and the Half-Blood Prince (Harry …	J.K. Rowling/Mary GrandPré	4.57
2	Harry Potter and the Order of the Phoenix (Har…	J.K. Rowling/Mary GrandPré	4.49
4	Harry Potter and the Chamber of Secrets (Harry…	J.K. Rowling	4.42
5	Harry Potter and the Prisoner of Azkaban (Harr…		
8	Harry Potter Boxed Set Books 1-5 (Harry Potte…		
9	Unauthorized Harry Potter Book Seven News: "Ha…		
10	Harry Potter Collection (Harry Potter #1-6)		
12	The Ultimate Hitchhiker's Guide: Five Complete…		
13	The Ultimate Hitchhiker's Guide to the Galaxy …		
14	The Hitchhiker's Guide to the Galaxy (Hitchhik…		
16	The Hitchhiker's Guide to the Galaxy (Hitchhik…		
18	The Ultimate Hitchhiker's Guide (Hitchhiker's …		
21	A Short History of Nearly Everything		
22	Bill Bryson's African Diary		
23	Bryson's Dictionary of Troublesome Words: A Wr…		
24	In a Sunburned Country		
25	I'm a Stranger Here Myself: Notes on Returning…		
26	The Lost Continent: Travels in Small Town America		
27	Neither Here nor There: Travels in Europe		
28	Notes from a Small Island		
29	The Mother Tongue: English and How It Got That…		
30	J.R.R. Tolkien 4-Book Boxed Set: The Hobbit an…		
31	The Lord of the Rings (The Lord of the Rings …		
34	The Fellowship of the Ring (The Lord of the Ri…		
35	The Lord of the Rings (The Lord of the Rings …		
36	The Lord of the Rings: Weapons and Warfare		
37	The Lord of the Rings: Complete Visual Companion		
45	Agile Web Development with Rails: A Pragmatic …		
50	Hatchet (Brian's Saga #1)		
51	Hatchet: A Guide for Using "Hatchet" in the Cl…		

5	J.K. Rowling/Mary GrandPré	4.56
8	J.K. Rowling/Mary GrandPré	4.78
9	W. Frederick Zimmerman	3.74
10	J.K. Rowling	4.73
12	Douglas Adams	4.38
13	Douglas Adams	4.38
14	Douglas Adams	4.22
16	Douglas Adams/Stephen Fry	4.22
18	Douglas Adams	4.38
21	Bill Bryson	4.21
22	Bill Bryson	3.44
23	Bill Bryson	3.87
24	Bill Bryson	4.07
25	Bill Bryson	3.90
26	Bill Bryson	3.83
27	Bill Bryson	3.86
28	Bill Bryson	3.91
29	Bill Bryson	3.93
30	J.R.R. Tolkien	4.59
31	J.R.R. Tolkien	4.50
34	J.R.R. Tolkien	4.36
35	J.R.R. Tolkien/Alan Lee	4.50
36	Chris Smith/Christopher Lee/Richard Taylor	4.53
37	Jude Fisher	4.50
45	Dave Thomas/David Heinemeier Hansson/Leon Bree…	3.84
50	Gary Paulsen	3.72
51	Donna Ickes/Edward Sciranko/Keith Vasconcelles	4.00

bookID	isbn	isbn13	language_code	num_pages	ratings_count	\
1	0439785960	9780439785969	eng	652	2095690	
2	0439358078	9780439358071	eng	870	2153167	
4	0439554896	9780439554893	eng	352	6333	
5	043965548X	9780439655484	eng	435	2339585	
8	0439682584	9780439682589	eng	2690	41428	
9	0976540606	9780976540601	en-US	152	19	
10	0439827604	9780439827607	eng	3342	28242	
12	0517226952	9780517226957	eng	815	3628	
13	0345453743	9780345453747	eng	815	249558	
14	1400052920	9781400052929	eng	215	4930	
16	0739322206	9780739322208	eng	6	1266	
18	0517149257	9780517149256	eng	815	2877	
21	076790818X	9780767908184	eng	544	248558	
22	0767915062	9780767915069	eng	55	7270	
23	0767910435	9780767910439	eng	256	2088	
24	0767903862	9780767903868	eng	335	72451	
25	076790382X	9780767903820	eng	304	49240	

26	0060920084	9780060920081	eng	299	45712
27	0380713802	9780380713806	eng	254	48701
28	0380727501	9780380727506	eng	324	80609
29	0380715430	9780380715435	eng	270	28489
30	0345538374	9780345538376	eng	1728	101233
31	0618517650	9780618517657	eng	1184	1710
34	0618346252	9780618346257	eng	398	2128944
35	0618260587	9780618260584	en-US	1216	1618
36	0618391002	9780618391004	eng	218	19822
37	0618510826	9780618510825	eng	224	359
45	097669400X	9780976694007	eng	558	1430
50	0689840926	9780689840920	eng	208	270244
51	1557344493	9781557344496	eng	48	36

bookID	text_reviews_count	publication_date
1	27591	9/16/2006
2	29221	9/1/2004
4	244	11/1/2003
5	36325	5/1/2004
8	164	9/13/2004
9	1	4/26/2005
10	808	9/12/2005
12	254	11/1/2005
13	4080	4/30/2002
14	460	8/3/2004
16	253	3/23/2005
18	195	1/17/1996
21	9396	9/14/2004
22	499	12/3/2002
23	131	9/14/2004
24	4245	5/15/2001
25	2211	6/28/2000
26	2257	8/28/1990
27	2238	3/28/1993
28	3301	5/28/1997
29	2085	9/28/1991
30	1550	9/25/2012
31	91	10/21/2004
34	13670	9/5/2003
35	140	10/1/2002
36	46	11/5/2003
37	6	11/15/2004
45	59	7/28/2005
50	12017	4/1/2000
51	2	8/28/1994

```
                                                publisher
bookID
1                                        Scholastic Inc.
2                                        Scholastic Inc.
4                                             Scholastic
5                                        Scholastic Inc.
8                                             Scholastic
9                                           Nimble Books
10                                            Scholastic
12                                         Gramercy Books
13                                          Del Rey Books
14                                                  Crown
16                                     Random House Audio
18                                            Wings Books
21                                         Broadway Books
22                                         Broadway Books
23                                         Broadway Books
24                                         Broadway Books
25                                         Broadway Books
26                               William Morrow Paperbacks
27                               William Morrow Paperbacks
28                               William Morrow Paperbacks
29                               William Morrow Paperbacks
30                                        Ballantine Books
31                               Houghton Mifflin Harcourt
34                               Houghton Mifflin Harcourt
35                               Houghton Mifflin Harcourt
36                               Houghton Mifflin Harcourt
37                               Houghton Mifflin Harcourt
45                                     Pragmatic Bookshelf
50      Atheneum Books for Young Readers: Richard Jack…
51                               Teacher Created Resources
```

Selection of the last 10 lines:

[4]:
```
dfBook.tail(10)
```

[4]:
```
                                                    title  \
bookID
45617   O Cavalo e o Seu Rapaz (As Crónicas de Nárnia …
45623    O Sobrinho do Mágico (As Crónicas de Nárnia  #1)
45625   A Viagem do Caminheiro da Alvorada (As Crónica…
45626      O Príncipe Caspian (As Crónicas de Nárnia  #4)
45630                                  Whores for Gloria
45631    Expelled from Eden: A William T. Vollmann Reader
45633                          You Bright and Risen Angels
45634                    The Ice-Shirt (Seven Dreams #1)
45639                                          Poor People
```

```
45641                          Las aventuras de Tom Sawyer

                                                    authors  average_rating  \
bookID
45617         C.S. Lewis/Pauline Baynes/Ana Falcão Bastos            3.92
45623         C.S. Lewis/Pauline Baynes/Ana Falcão Bastos            4.04
45625         C.S. Lewis/Pauline Baynes/Ana Falcão Bastos            4.09
45626         C.S. Lewis/Pauline Baynes/Ana Falcão Bastos            3.97
45630                                 William T. Vollmann            3.69
45631   William T. Vollmann/Larry McCaffery/Michael He…            4.06
45633                                 William T. Vollmann            4.08
45634                                 William T. Vollmann            3.96
45639                                 William T. Vollmann            3.72
45641                                          Mark Twain            3.91

              isbn         isbn13 language_code  num_pages  ratings_count  \
bookID
45617   9722330551  9789722330558           por        160            207
45623   9722329987  9789722329989           por        147            396
45625   9722331329  9789722331326           por        176            161
45626   9722330977  9789722330978           por        160            215
45630   0140231579  9780140231571         en-US        160            932
45631   1560254416  9781560254416           eng        512            156
45633   0140110879  9780140110876           eng        635            783
45634   0140131965  9780140131963           eng        415            820
45639   0060878827  9780060878825           eng        434            769
45641   8497646983  9788497646987           spa        272            113

        text_reviews_count publication_date           publisher
bookID
45617                   16        8/15/2003  Editorial Presença
45623                   37         4/8/2003  Editorial Presença
45625                   14         9/1/2004  Editorial Presença
45626                   11       10/11/2003  Editorial Presença
45630                  111         2/1/1994       Penguin Books
45631                   20       12/21/2004       Da Capo Press
45633                   56        12/1/1988       Penguin Books
45634                   95         8/1/1993       Penguin Books
45639                  139        2/27/2007                Ecco
45641                   12        5/28/2006       Edimat Libros
```

Selecting a 10 lines random dataset:

[5]:
```
dfBook.sample(10)
```

[5]:
```
                                                    title  \
bookID
9742    The Audacity of Hope: Thoughts on Reclaiming t…
```

```
3631                                      Catching Alice
7841    Another Day in Paradise: The Fourth Sherman's …
10802   Beyond the Hundredth Meridian: John Wesley Pow…
15549     Armageddon's Children (Genesis of Shannara  #1)
30348                                  A Whole New Light
17289      Galactic Goodnight (Disney's Little Einsteins)
37339                     On a Wicked Dawn (Cynster  #9)
15075   The Art of Alfred Hitchcock: Fifty Years of Hi…
14060   Little House Friends (Little House Chapter Boo…

                                authors  average_rating        isbn  \
bookID
9742                        Barack Obama            3.75  0307237699
3631                        Clare Naylor            3.38  0449005577
7841                          Jim Toomey            4.35  0740720120
10802     Wallace Stegner/Bernard DeVoto            4.07  0140159940
15549                       Terry Brooks            4.10  0345484088
30348                       Sandra Brown            3.55  055329783X
17289     Susan Ring/Kirk Albert Etienne            4.42  0786849738
37339                  Stephanie Laurens            4.06  0060002050
15075                       Donald Spoto            4.08  0385418132
14060   Laura Ingalls Wilder/Renée Graef            4.04  0064420809

              isbn13 language_code  num_pages  ratings_count  \
bookID
9742    9780307237699           eng        375         127324
3631    9780449005576         en-US        328            948
7841    9780740720123           eng        130             68
10802   9780140159943           eng        438           3010
15549   9780345484086           eng        371          12609
30348   9780553297836           eng        227           2955
17289   9780786849734           eng         20             12
37339   9780060002053           eng        448           5708
15075   9780385418133         en-US        496            468
14060   9780064420808           eng         80             73

        text_reviews_count publication_date                 publisher
bookID
9742                  4496       10/17/2006                     Crown
3631                    27       10/31/2000           Ballantine Books
7841                     1        9/6/2001  Andrews McMeel Publishing
10802                  244        3/1/1992              Penguin Books
15549                  524        9/30/2006           Ballantine Books
30348                  203        8/26/2008   Bantam (Fanfare Imprint)
17289                    2        9/1/2006               Disney Press
37339                  133        4/30/2002                     Avon
15075                   18        12/1/1991                   Anchor
```

```
14060                    2         9/5/1998              HarperCollins
```

We get the information from our dataframe `dfBook` through the `info()` function of the `pandas` library.

[6]:
```
dfBook.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11123 entries, 1 to 45641
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   title               11123 non-null  object
 1   authors             11123 non-null  object
 2   average_rating      11123 non-null  float64
 3   isbn                11123 non-null  object
 4   isbn13              11123 non-null  int64
 5   language_code       11123 non-null  object
 6   num_pages           11123 non-null  int64
 7   ratings_count       11123 non-null  int64
 8   text_reviews_count  11123 non-null  int64
 9   publication_date    11123 non-null  object
 10  publisher           11123 non-null  object
dtypes: float64(1), int64(4), object(6)
memory usage: 1.0+ MB
```

In order to better understand our dataset, we can also use the `describe()` function, which displays basic statistical details such as the mean, minimum, maximum, … columns.

> Remark: The `describe()` function returns the different statistical details only on the numerical columns (float, int, …)

[7]:
```
dfBook.describe()
```

[7]:
```
       average_rating        isbn13     num_pages  ratings_count  \
count    11123.000000  1.112300e+04  11123.000000   1.112300e+04
mean         3.934075  9.759880e+12    336.405556   1.794285e+04
std          0.350485  4.429758e+11    241.152626   1.124992e+05
min          0.000000  8.987060e+09      0.000000   0.000000e+00
25%          3.770000  9.780345e+12    192.000000   1.040000e+02
50%          3.960000  9.780582e+12    299.000000   7.450000e+02
75%          4.140000  9.780872e+12    416.000000   5.000500e+03
max          5.000000  9.790008e+12   6576.000000   4.597666e+06

       text_reviews_count
count        11123.000000
mean           542.048099
std           2576.619589
min              0.000000
```

```
25%              9.000000
50%             47.000000
75%            238.000000
max          94265.000000
```

We can also use the `shape` tuple to just look at the number of rows and columns in our dataframe:

[8]:
```
dfBook.shape
```

[8]:
```
(11123, 11)
```

Remark: The first digit corresponds to the rows and the second digit corresponds to the columns.

2 Data Cleaning, Exploratory Analysis and Relevant Charts

In order to have a more concrete analysis process, we need to clean the data to make our data set more "clean" and closer to reality. This process allows you to modify or delete incorrect, incomplete, irrelevant, corrupted, duplicated or formatted data.

To do this, we need to look more closely at our data:

2.1 Nulls

Using the `isna()` and `sum()` function, we can see the number of null values (specified by `NA`) in each column of the dataframe:

[9]:
```
dfBook.isna().sum()
```

[9]:
```
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
```

Conclusion: We can see that we do not have null data.

2.2 The empty values

In order to better locate empty values, we must, using the function `replace()`, replace empty values (specified by `''`) by values `na` (specified by `np.nan`):

[10]:
```
dfBook.replace('', np.nan)
dfBook.isna().sum()
```

[10]:
```
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
```

Conclusion: We can also see that there are no empty values.

2.3 The columns "useless"

This step depends on which column we want to analyze. For our part, we delete the isbn13, isbn and publisher column using the `drop()` function. These columns will not be useful after our analysis. As long as the 'publisher' column could be analyzed.

[11]:
```
dfBook.shape
```

[11]:
```
(11123, 11)
```

[12]:
```
dfBookClear = dfBook.drop(columns=["isbn13", "isbn", "publisher"])
```

We look to see if the column number has decreased.

[13]:
```
dfBookClear.shape
```

[13]:
```
(11123, 8)
```

We can also check with the `info()` function:

[14]:
```
dfBookClear.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11123 entries, 1 to 45641
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   title               11123 non-null  object
 1   authors             11123 non-null  object
 2   average_rating      11123 non-null  float64
 3   language_code       11123 non-null  object
 4   num_pages           11123 non-null  int64
```

```
 5   ratings_count       11123 non-null  int64
 6   text_reviews_count  11123 non-null  int64
 7   publication_date    11123 non-null  object
dtypes: float64(1), int64(3), object(4)
memory usage: 782.1+ KB
```

Or with the `describe()` function:

[15]:
```
dfBookClear.describe()
```

[15]:

	average_rating	num_pages	ratings_count	text_reviews_count
count	11123.000000	11123.000000	1.112300e+04	11123.000000
mean	3.934075	336.405556	1.794285e+04	542.048099
std	0.350485	241.152626	1.124992e+05	2576.619589
min	0.000000	0.000000	0.000000e+00	0.000000
25%	3.770000	192.000000	1.040000e+02	9.000000
50%	3.960000	299.000000	7.450000e+02	47.000000
75%	4.140000	416.000000	5.000500e+03	238.000000
max	5.000000	6576.000000	4.597666e+06	94265.000000

2.4 Removing duplicates from title, language and authors

In a large dataset like this, most of the time we have duplicates. To locate them, we need to copy the initial dataframe into a temporary dataframe and display the lines that are duplicated and then compare the results.

In our case, we will say that a duplicate is a book that has title, the same author and the same language.

[16]:
```
dfDuplicatesRows = dfBookClear.loc[dfBookClear.sort_values('ratings_count').
↪duplicated(subset=['title','authors','language_code'], keep='last')]
dfDuplicatesRows
```

[16]:

bookID	title	authors	average_rating	language_code
69	The Known World			
123	The Power of One (The Power of One #1)			
297	Treasure Island			
324	Cien años de soledad			
403	Americana			
…	…			
45149	A Painted House			
45223	I Never Promised You a Rose Garden			
45280	Quicksilver (The Baroque Cycle #1)			
45306	Haruki Murakami and the Music of Words			
45316	Sputnik Sweetheart			

```
69                      Edward P. Jones            3.83          eng
123                     Bryce Courtenay            4.35          eng
297             Robert Louis Stevenson             3.83          eng
324             Gabriel García Márquez             4.07          spa
403                        Don DeLillo             3.43          eng
…                                  …                …            …
45149                      John Grisham            3.70          eng
45223   Hannah  Green/Joanne Greenberg             3.87          eng
45280                   Neal Stephenson            3.93          eng
45306                       Jay Rubin              3.83          eng
45316   Haruki Murakami/Philip Gabriel             3.83          eng

        num_pages  ratings_count  text_reviews_count publication_date
bookID
69            576             22                   3        6/15/2004
123           291             45                  13        9/13/2005
297           245           5967                 276         6/1/2005
324           448             63                   7         1/1/1990
403           377            393                  55         7/6/1989
…             …                …                   …            …
45149         466            484                  48        9/29/2002
45223         256              2                   0         3/1/1965
45280         927            171                  12        10/2/2003
45306         326             38                   2        12/3/2005
45316         210            582                  92        4/24/2001

[232 rows x 8 columns]
```

For example for the book “Treasure Island”:

[17]:
```
dfBookClear.loc[(dfBookClear['title'] == "Treasure Island") &␣
↪(dfBookClear['language_code'] == "eng") & (dfBookClear['authors'] == ␣
↪"Robert Louis Stevenson")]
```

[17]:
```
                title                authors  average_rating language_code  \
bookID
295     Treasure Island  Robert Louis Stevenson            3.83          eng
297     Treasure Island  Robert Louis Stevenson            3.83          eng

        num_pages  ratings_count  text_reviews_count publication_date
bookID
295           311         318753                6734        9/15/2001
297           245           5967                 276         6/1/2005
```

Remark: This step consists in verifying the duplicates.

We want to keep the book that has more votes so that the analysis is more reliable. To do this, we sort the dataset in ascending order (smaller to larger) according to the number of votes and then delete the first (lower).

[18]:
```
dfBookClear = dfBookClear.
 ↪drop_duplicates(subset=['title','authors','language_code'])
dfBookClear
```

[18]:

```
                                                  title  \
bookID
1       Harry Potter and the Half-Blood Prince (Harry …
2       Harry Potter and the Order of the Phoenix (Har…
4       Harry Potter and the Chamber of Secrets (Harry…
5       Harry Potter and the Prisoner of Azkaban (Harr…
8       Harry Potter Boxed Set  Books 1-5 (Harry Potte…
…                                                     …
45631    Expelled from Eden: A William T. Vollmann Reader
45633                          You Bright and Risen Angels
45634                    The Ice-Shirt (Seven Dreams #1)
45639                                         Poor People
45641                          Las aventuras de Tom Sawyer

                                                  authors  average_rating  \
bookID
1                              J.K. Rowling/Mary GrandPré            4.57
2                              J.K. Rowling/Mary GrandPré            4.49
4                                           J.K. Rowling             4.42
5                              J.K. Rowling/Mary GrandPré            4.56
8                              J.K. Rowling/Mary GrandPré            4.78
…                                                     …             …
45631   William T. Vollmann/Larry McCaffery/Michael He…             4.06
45633                                 William T. Vollmann            4.08
45634                                 William T. Vollmann            3.96
45639                                 William T. Vollmann            3.72
45641                                          Mark Twain            3.91

       language_code  num_pages  ratings_count  text_reviews_count  \
bookID
1                eng        652        2095690               27591
2                eng        870        2153167               29221
4                eng        352           6333                 244
5                eng        435        2339585               36325
8                eng       2690          41428                 164
…                …          …             …                     …
45631            eng        512            156                  20
45633            eng        635            783                  56
45634            eng        415            820                  95
45639            eng        434            769                 139
45641            spa        272            113                  12

       publication_date
```

```
bookID
1             9/16/2006
2              9/1/2004
4             11/1/2003
5              5/1/2004
8             9/13/2004
…                   …
45631        12/21/2004
45633         12/1/1988
45634          8/1/1993
45639         2/27/2007
45641         5/28/2006

[10891 rows x 8 columns]
```

[19]:
```
dfBookClear.loc[(dfBookClear['title'] == "Treasure Island")]
```

[19]:
```
                  title                                          authors  \
bookID
295     Treasure Island                              Robert Louis Stevenson
296     Treasure Island                 Robert Louis Stevenson/Scott McKowen
298     Treasure Island      Robert Louis Stevenson/N.C. Wyeth/Timothy Meis
299     Treasure Island                   Robert Louis Stevenson/Milo Winter
19347   Treasure Island  Chris Tait/Robert Louis Stevenson/Lucy Corvino…

        average_rating language_code  num_pages  ratings_count  \
bookID
295               3.83           eng        311         318753
296               3.83         en-US        213            420
298               3.83           eng         64            104
299               3.83         en-US        272             56
19347             3.83           eng        160            798

        text_reviews_count publication_date
bookID
295                   6734        9/15/2001
296                     42        10/1/2004
298                     14         7/1/2003
299                      4         9/3/2002
19347                   39         3/1/2005
```

We delete the duplicate lines seen above. 11 123 - 10 891 = 232 (obtained above)

2.4.1 Analysis of deleted duplicate data

The fields used above in the `drop_duplicates()`: Title, Authors and language_code function have been defined since this data analysis. It checks if certain data is deleted and should not be. For example, at the beginning of our analysis, we did not put the language_code column but thanks

to this analysis, we could see that the function removed from books with the same title but not with the same language. In conclusion, these are interesting data to keep, we do not want to delete them.

[20]:
```
dfDuplicatesRows.head(10)
```

[20]:

bookID	title	authors	average_rating	language_code	num_pages	ratings_count	text_reviews_count	publication_date
69	The Known World	Edward P. Jones	3.83	eng	576	22	3	6/15/2004
123	The Power of One (The Power of One #1)	Bryce Courtenay	4.35	eng	291	45	13	9/13/2005
297	Treasure Island	Robert Louis Stevenson	3.83	eng	245	5967	276	6/1/2005
324	Cien años de soledad	Gabriel García Márquez	4.07	spa	448	63	7	1/1/1990
403	Americana	Don DeLillo	3.43	eng	377	393	55	7/6/1989
411	The Crying of Lot 49	Thomas Pynchon	3.69	eng	152	2161	230	4/1/1999
412	Gravity's Rainbow	Thomas Pynchon	4.01	eng	784	762	213	1/1/2000
416	Slow Learner: Early Stories	Thomas Pynchon	3.50	eng	193	334	34	4/30/1985
524	Lord of the Flies	William Golding	3.68	eng	6	408	96	10/11/2005
538	The Lovely Bones	Alice Sebold	3.81	en-US	532	367	73	4/1/2004

2.5 Removal of inconsistent data

Using the `describe()` function, we noticed that some fields had data at 0 (min). In order to make the analysis more consistent, we decided to delete this data.

[21]:
```
dfBookClear.describe()
```

[21]:

	average_rating	num_pages	ratings_count	text_reviews_count
count	10891.000000	10891.000000	1.089100e+04	10891.000000
mean	3.934274	335.731521	1.763207e+04	535.108989
std	0.352151	241.699733	1.105703e+05	2556.424471
min	0.000000	0.000000	0.000000e+00	0.000000
25%	3.770000	192.000000	1.050000e+02	9.000000
50%	3.960000	297.000000	7.620000e+02	47.000000
75%	4.140000	416.000000	5.064000e+03	239.000000
max	5.000000	6576.000000	4.597666e+06	94265.000000

[22]:
```
dfBookClear.shape
```

[22]:
```
(10891, 8)
```

[23]:
```
# Histogramm for average rating

fig = px.histogram(dfBookClear, x="average_rating", title="Average Rating",␣
 ↪color_discrete_sequence =["#900C3F"])
fig.show()

# Histogram for numbers pages

fig = px.histogram(dfBookClear, x="num_pages", title="Numbers of pages",␣
 ↪color_discrete_sequence =["#FF5733"])
fig.show()
```

Interpretation of the "Average Rating" histogram: We can interpret the average vote is between 3.5 and 4.5. We can say that the analysis will be based only between that range. For me, we don't have a data set that is "broad enough" to have a precise analysis.

> Conclusion of the "Average Rating" histogram: A reader is likely to give a rating around 4.

Interpretation of the "Numbers Page" histogram: In addition, we chose to delete books with 0 pages so that our analysis is more accurate. That is inconsistent data.

[24]:
```
dfBookClear = dfBookClear.drop(dfBookClear[(dfBookClear['num_pages'] == 0)].
 ↪index)
```

[25]:
```
dfBookClear.describe()
```

[25]:

	average_rating	num_pages	ratings_count	text_reviews_count
count	10817.000000	10817.000000	1.081700e+04	10817.000000
mean	3.934341	338.028289	1.775003e+04	538.666359
std	0.351935	240.919077	1.109385e+05	2564.788024
min	0.000000	1.000000	0.000000e+00	0.000000
25%	3.770000	195.000000	1.100000e+02	9.000000
50%	3.960000	300.000000	7.780000e+02	48.000000
75%	4.140000	416.000000	5.141000e+03	242.000000
max	5.000000	6576.000000	4.597666e+06	94265.000000

[26]:
```
dfBookClear.shape
```

[26]:
```
(10817, 8)
```

After removing books with 0 pages, we have 10817 books in our data set.

3 Prediction Model

We try to predict the note of a book, for this we have the choice between two models of prediction: - Classification model - Regression model

3.1 Classification Model

The classification process searches for a function that helps divide the dataset into classes based on different parameters.

In our case, we cannot have a classification process because the value we want to predict is not a class but a continuous value.

If our data set included the genres of books, we could have performed a clustering model to determine the genre of a book.

3.2 Regression Model

The regression process involves finding correlations between dependent and independent variables. It helps to predict continuous variables as in our case with the rating of a book.

Thanks to the `corr()` function, we can see which data is dependent or not. This is done only on the numerical columns (float, int, ...).

[27]:
```
dfBookClear.corr()
```

[27]:

	average_rating	num_pages	ratings_count	\
average_rating	1.000000	0.152500	0.039558	
num_pages	0.152500	1.000000	0.034903	
ratings_count	0.039558	0.034903	1.000000	
text_reviews_count	0.033850	0.036226	0.864636	

	text_reviews_count
average_rating	0.033850

```
num_pages                  0.036226
ratings_count              0.864636
text_reviews_count         1.000000
```

[28]:
```
fig = px.imshow(dfBookClear.corr(), aspect="auto")
fig.show()
```

We can see through the heatmap the different correlations: - `ratings_count` and `text_reviews_count`: The number of notes depends on the number of comments. A reader will, most of the time, write down the book AND write a comment. We may think that adding a comment is mandatory to put a note. - `pages` and `average_rating`: The average rating depends on the number of pages.

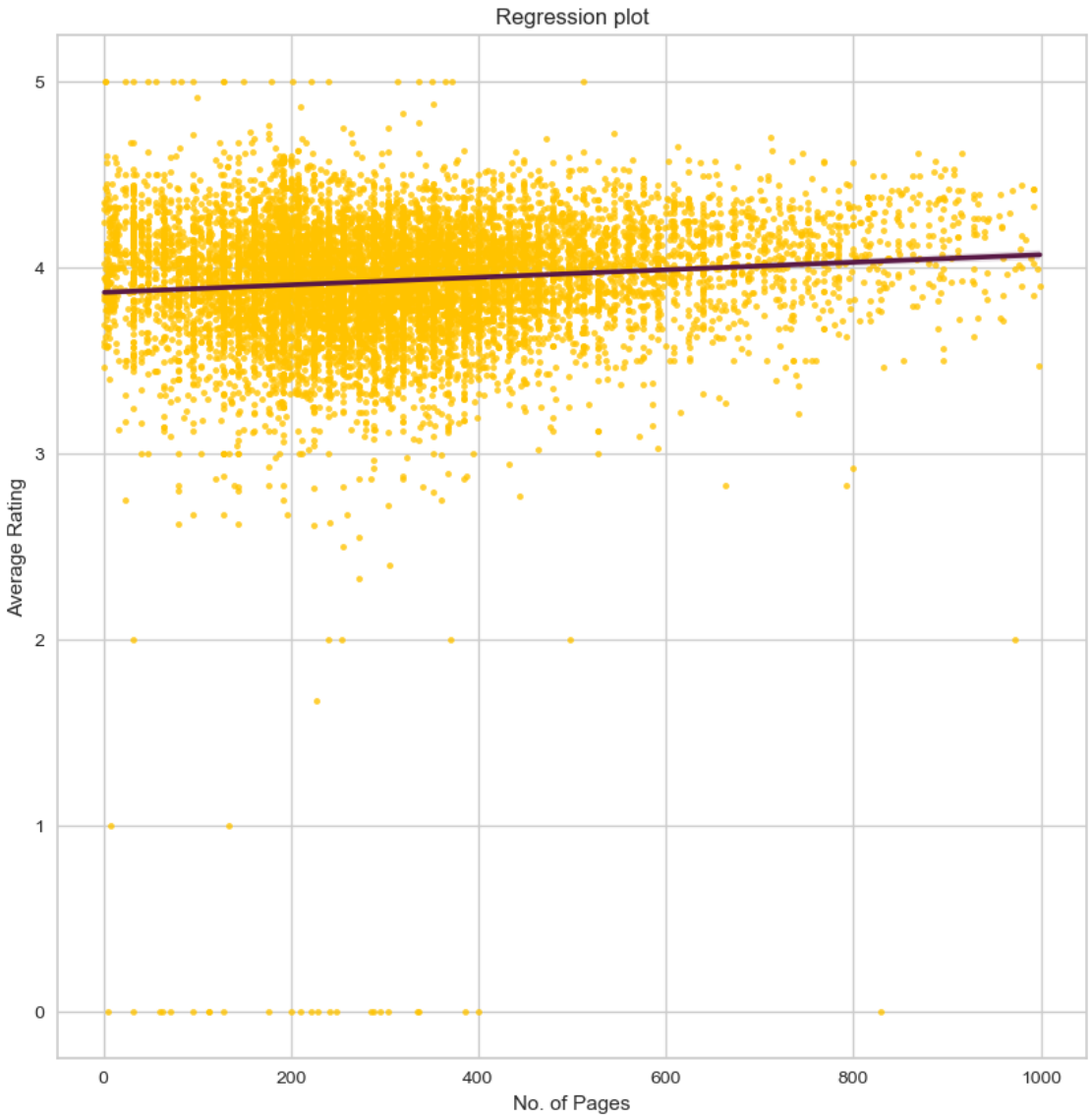
Exploratory phase: Number of votes by number of pages In this section, we visualize the correlation between the number of pages of a book and its voting average.

Regression plot

[118]:
```
fig = px.scatter(dfBookClear, x="average_rating", y="num_pages",␣
↪color='average_rating', title="Regression plot", trendline="ols")
fig.show()
```

We can also use the `seaborn` library with the `regplot()` function to make our regression graph.

[119]:
```
plt.figure(figsize=(10,10))
sns.regplot(data=dfBookClear,y="average_rating",x="num_pages",marker='.
↪',scatter_kws={"color": "#FFC300"}, line_kws={"color": "#581845"})
plt.xlabel('No. of Pages')
plt.ylabel('Average Rating')
plt.title("Regression plot")
plt.show()
```



With the regression charts, we notice an upward trend in the rating relative to the number of pages. The reader tends to put higher notes on books with more pages.

Graphic to show outliers points.

[114]:
```
fig = px.scatter(dfBookClear, x="average_rating", y="num_pages",␣
↪marginal_x="histogram", color_discrete_sequence = ['#F1C40F'],␣
↪marginal_y="violin", title="Outliers")
fig.show()
```

[115]:
```
fig = px.box(dfBookClear, x="num_pages", color_discrete_sequence = ['#900C3F'],␣
↪title="Outliers")
fig.show()
```

All values above 1000 are off-centered points (outliers). Therefore, you must delete them to have a more accurate dataset.

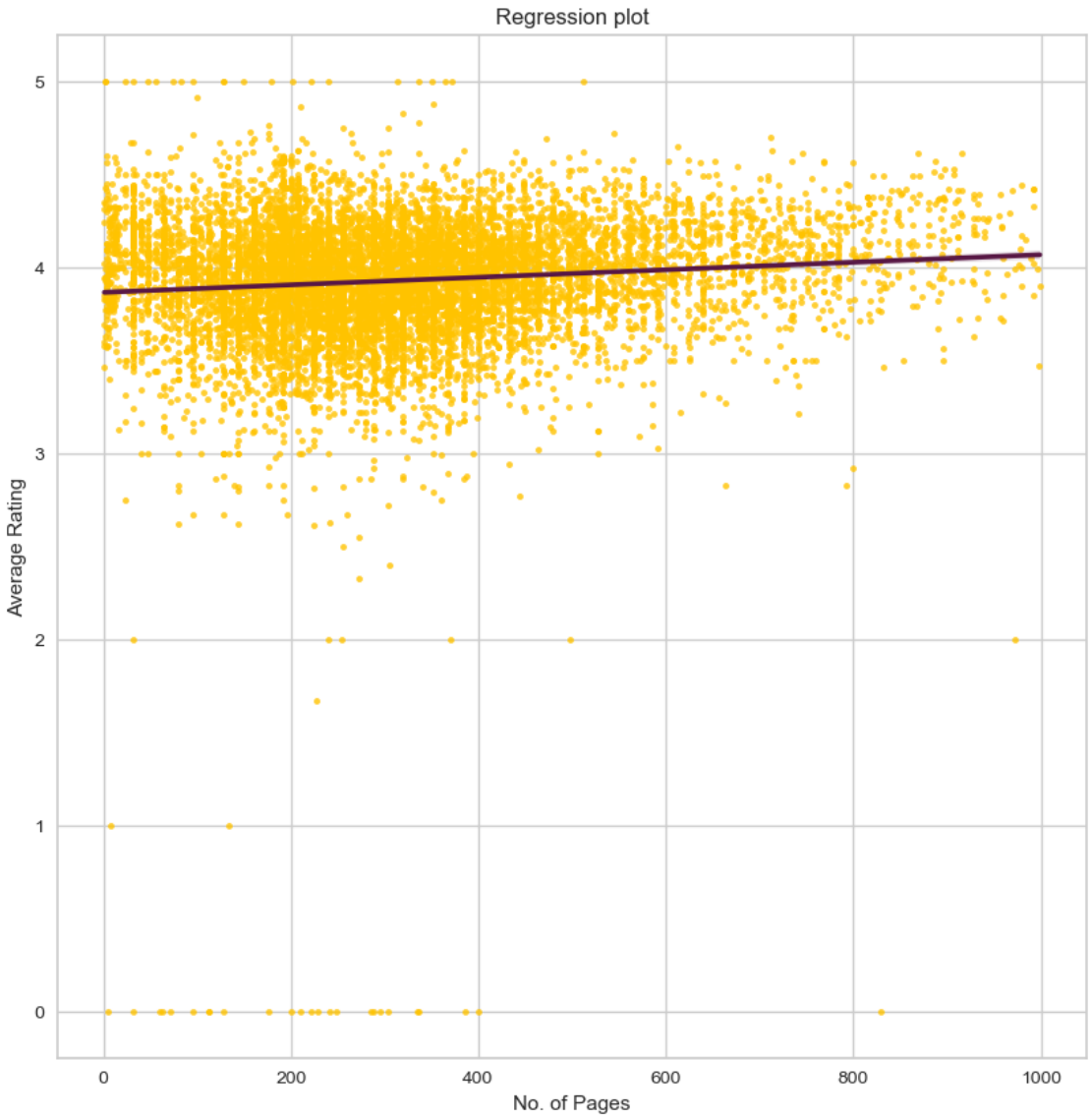
[33]:
```
dfBookClear = dfBookClear.drop(dfBookClear.index[dfBookClear['num_pages'] >=␣
↪1000])
```

[116]:
```
fig = px.scatter(dfBookClear, x="average_rating", y="num_pages",␣
↪marginal_x="histogram", color_discrete_sequence = ['#F1C40F'],␣
↪marginal_y="violin", title="Outliers")
fig.show()
```

Results with regression charts We look with the regression charts to see if our approach above has impacted the trend between page count and voting average. We have smoothed our results by discarding out-of-center values.

[120]:
```
fig = px.scatter(dfBookClear, x="average_rating", y="num_pages",␣
↪title="Regression plot", color='average_rating', trendline="ols")
fig.show()
```

[121]:
```
plt.figure(figsize=(10,10))
sns.regplot(data=dfBookClear,y="average_rating",x="num_pages",marker='.
↪',scatter_kws={"color": "#FFC300"}, line_kws={"color": "#581845"})
plt.xlabel('No. of Pages')
plt.ylabel('Average Rating')
plt.title("Regression plot")
plt.show()
```



To use our regression model, we can also correlate a specific column to the other columns:

[37]:
```
dfBookClear.corr()["average_rating"]
```

[37]:
```
average_rating        1.000000
num_pages             0.103910
ratings_count         0.040306
text_reviews_count    0.035535
Name: average_rating, dtype: float64
```

> Remark: The closer the result is to 1, the more the data is correlated with the data in the `average_rating` column

If we want all columns to correlate, we need to change the `string` type columns to the `numeric`

type, that is, standardize our dataset.

3.2.1 Standardization

The standardization process is the conversion of data into a "standard" format.

[38]:
```
dfBookClear.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10603 entries, 1 to 45641
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   title               10603 non-null  object
 1   authors             10603 non-null  object
 2   average_rating      10603 non-null  float64
 3   language_code       10603 non-null  object
 4   num_pages           10603 non-null  int64
 5   ratings_count       10603 non-null  int64
 6   text_reviews_count  10603 non-null  int64
 7   publication_date    10603 non-null  object
dtypes: float64(1), int64(3), object(4)
memory usage: 745.5+ KB
```

We can see that the columns `title`, `authors`, `publication_date` are not numeric.

Transform the "string" fields into numbers: Authors, Title , Publication_date and Language_code We will use the functions of the `sklearn` library to transform our data types. We make a copy of the basic data frame so as not to lose our previous data.

[39]:
```
from sklearn import preprocessing
dfBookEncoded = dfBookClear.copy() # copy of the dataframe dfBookClean for the␣
 ↪create dataframe encoded
encoder = preprocessing.LabelEncoder()
dfBookEncoded['title'] = encoder.fit_transform(dfBookEncoded['title'])
dfBookEncoded['authors'] = encoder.fit_transform(dfBookEncoded['authors'])
dfBookEncoded['language_code'] = encoder.
 ↪fit_transform(dfBookEncoded['language_code'])
dfBookEncoded['publication_date'] = encoder.
 ↪fit_transform(dfBookEncoded['publication_date'])
```

[40]:
```
dfBookEncoded.sample(10)
```

[40]:

bookID	title	authors	average_rating	language_code	num_pages	\
19802	349	6318	3.85	5	322	
21803	3830	6296	4.26	21	480	
10370	9756	6409	4.37	5	288	
3403	5062	3641	3.72	4	406	

```
24115    2925     1453            4.29              5         777
8077      618     3284            4.11              5          32
44179    9080     5296            3.87              5         289
33057    1286     1815            3.86              5         368
8656     7869     3065            4.01              5         320
33978    2092     1417            3.91              5          96

         ratings_count  text_reviews_count  publication_date
bookID
19802             9807                 745              2414
21803               39                   7              2470
10370              121                   5              3244
3403              1170                 117              1086
24115              330                  26              1721
8077                 0                   0               600
44179             2143                 254              1640
33057              649                  56               40
8656               148                   6               726
33978              110                  33               878
```

[41]:
```
dfBookEncoded.sample(10)
```

[41]:
```
         title  authors  average_rating  language_code  num_pages  \
bookID
10990     4093     1359            3.70              5        368
89         387     2294            3.94              5        138
22126      914     3630            4.12              5        768
15124     3162     1997            3.71              5        320
17880     7452     1924            3.89              5        160
30228     4426     3611            3.63              4        245
28721     1598     2247            4.09              5        126
11980     3951     2873            3.96              7        165
38597     6182     1830            3.35              5        153
18337     4813      945            3.87              5        272

         ratings_count  text_reviews_count  publication_date
bookID
10990            32446                1345               630
89                  31                   1              1708
22126             1126                  50              3006
15124               16                   2               349
17880               82                   8               362
30228            39849                 868              1531
28721              180                  10              3249
11980             1951                  77              1250
38597                3                   0               975
18337              193                  34              1942
```

Now that we have implemented this process, we can look at the correlation only in the `average_rating` column with the other columns.

[42]:
```
dfBookEncoded.corr()["average_rating"].sort_values(ascending=False)
```

[42]:
```
average_rating        1.000000
num_pages             0.103910
ratings_count         0.040306
text_reviews_count    0.035535
language_code         0.032761
title                 0.020449
authors               0.019406
publication_date     -0.008691
Name: average_rating, dtype: float64
```

[43]:
```
dfBookEncoded.corr()
```

[43]:

	title	authors	average_rating	language_code	num_pages	ratings_count	text_reviews_count	publication_date
title	1.000000	0.026113	0.020449	-0.037810	0.003978	0.003930	0.010998	0.005233
authors	0.026113	1.000000	0.019406	-0.000118	-0.001667	0.001544	-0.005593	-0.010395
average_rating	0.020449	0.019406	1.000000	0.032761	0.103910	0.040306	0.035535	-0.008691
language_code	-0.037810	-0.000118	0.032761	1.000000	0.030613	-0.015692	-0.024355	0.000671
num_pages	0.003978	-0.001667	0.103910	0.030613	1.000000	0.047805	0.053483	0.003278
ratings_count	0.003930	0.001544	0.040306	-0.015692	0.047805	1.000000	0.864042	0.025317
text_reviews_count	0.010998	-0.005593	0.035535	-0.024355	0.053483	0.864042	1.000000	0.029930
publication_date	0.005233	-0.010395	-0.008691	0.000671	0.003278	0.025317	0.029930	1.000000

[44]:
```
fig = px.imshow(dfBookEncoded.corr(), aspect="auto")
fig.show()
```

Despite the standardization of the Authors, Title, Publication_date and Language_code columns, we note that there are no additional correlations.

Exploratory phase : Transform the "string" fields into numbers : Authors, Title and Publication_date We can try to define the `language_code` column as a category column to see if there is a better correlation. For this, we create another data frame without transforming the column.

[45]:
```
from sklearn import preprocessing
dfBookEncodedWithoutLanguageCode = dfBookClear.copy() # copy of the dataframe␣
 ↪dfBookClean for the create dataframe encoded
encoder = preprocessing.LabelEncoder()
dfBookEncodedWithoutLanguageCode['title'] = encoder.
 ↪fit_transform(dfBookEncoded['title'])
dfBookEncodedWithoutLanguageCode['authors'] = encoder.
 ↪fit_transform(dfBookEncoded['authors'])
dfBookEncodedWithoutLanguageCode['publication_date'] = encoder.
 ↪fit_transform(dfBookEncoded['publication_date'])
```

The `get_dummies()` function is used to convert category variables to dummy variables. This can also be used to add column names with a prefix, for example. In our case, the category variable is 'language_code':

[46]:
```
dfBookEncodedWithoutLanguageCode = pd.
 ↪get_dummies(dfBookEncodedWithoutLanguageCode)
```

[47]:
```
dfBookEncodedWithoutLanguageCode.sample(10)
```

[47]:

bookID	title	authors	average_rating	num_pages	ratings_count	\
36637	981	2982	4.13	240	3626	
841	2240	1395	3.95	272	3702	
5359	6932	3086	4.00	483	367399	
15669	6239	5889	4.38	392	999	
35178	3459	68	3.33	313	269	
2756	4761	2839	3.00	209	2	
4415	2104	3156	4.37	691	83	
38564	1117	3945	3.68	272	10669	
19571	5128	2811	3.85	264	257	
24525	8870	4688	4.36	304	6025	

bookID	text_reviews_count	publication_date	language_code_ale	\
36637	145	536	0	
841	149	2157	0	

```
5359                  1978              1606                   0
15669                   24              1026                   0
35178                   23              2132                   0
2756                     0              2259                   0
4415                    18                22                   0
38564                  498              2066                   0
19571                   33              1923                   0
24525                  139              2762                   0

        language_code_ara  language_code_en-CA  …  language_code_nl  \
bookID                                          …
36637                   0                    0  …                 0
841                     0                    0  …                 0
5359                    0                    0  …                 0
15669                   0                    0  …                 0
35178                   0                    0  …                 0
2756                    0                    0  …                 0
4415                    0                    0  …                 0
38564                   0                    0  …                 0
19571                   0                    0  …                 0
24525                   0                    0  …                 0

        language_code_nor  language_code_por  language_code_rus  \
bookID
36637                   0                  0                  0
841                     0                  0                  0
5359                    0                  0                  0
15669                   0                  0                  0
35178                   0                  0                  0
2756                    0                  0                  0
4415                    0                  0                  0
38564                   0                  0                  0
19571                   0                  0                  0
24525                   0                  0                  0

        language_code_spa  language_code_srp  language_code_swe  \
bookID
36637                   0                  0                  0
841                     0                  0                  0
5359                    0                  0                  0
15669                   0                  0                  0
35178                   0                  0                  0
2756                    0                  0                  0
4415                    0                  0                  0
38564                   0                  0                  0
19571                   0                  0                  0
24525                   0                  0                  0
```

```
        language_code_tur  language_code_wel  language_code_zho
bookID
36637                   0                  0                  0
841                     0                  0                  0
5359                    0                  0                  0
15669                   0                  0                  0
35178                   0                  0                  0
2756                    0                  0                  0
4415                    0                  0                  0
38564                   0                  0                  0
19571                   0                  0                  0
24525                   0                  0                  0

[10 rows x 34 columns]
```

[48]:
```
dfBookEncodedWithoutLanguageCode.corr()["average_rating"].
 ↪sort_values(ascending=False)
```

[48]:
```
average_rating         1.000000
num_pages              0.103910
language_code_jpn      0.063982
language_code_zho      0.054702
ratings_count          0.040306
text_reviews_count     0.035535
language_code_wel      0.029659
language_code_mul      0.023872
title                  0.020449
language_code_lat      0.020365
authors                0.019406
language_code_gla      0.014986
language_code_tur      0.013602
language_code_rus      0.012777
language_code_fre      0.012656
language_code_ale      0.011941
language_code_ita      0.009247
language_code_en-CA    0.007111
language_code_nl       0.006958
language_code_msa      0.005020
language_code_ger      0.002733
language_code_por      0.001432
language_code_enm      0.000591
language_code_spa     -0.000819
language_code_eng     -0.001202
language_code_en-GB   -0.006107
publication_date      -0.008691
language_code_nor     -0.009098
```

```
language_code_ara     -0.010483
language_code_glg     -0.015742
language_code_swe     -0.018545
language_code_grc     -0.020336
language_code_en-US   -0.020568
language_code_srp     -0.108760
Name: average_rating, dtype: float64
```

[49]:
```
fig = px.imshow(dfBookEncodedWithoutLanguageCode.corr(), aspect="auto")
fig.show()
```

There is no relevant data from this `heatmap`. But we can deepen our analysis on the languages of books in an exploratory phase.

3.2.2 Exploratory phase: Data analysis with the language_code column

We can also analyze our data with the language of the books. We take back the basic data frame because in the two data frames we have just built, we no longer have languages in the form of strings but in the form of numbers.

[50]:
```
dfBookClear['language_code']
```

[50]:
```
bookID
1          eng
2          eng
4          eng
5          eng
9        en-US
         …
45631      eng
45633      eng
45634      eng
45639      eng
45641      spa
Name: language_code, Length: 10603, dtype: object
```

[51]:
```
dfBookClear['language_code'].isna().sum() #result 0
dfBookClear['language_code'].unique()
```

[51]:
```
array(['eng', 'en-US', 'fre', 'spa', 'en-GB', 'mul', 'grc', 'en-CA',
       'ger', 'jpn', 'ara', 'nl', 'zho', 'lat', 'por', 'srp', 'ita',
       'rus', 'msa', 'glg', 'wel', 'swe', 'nor', 'enm', 'tur', 'gla',
       'ale'], dtype=object)
```

With the `unique()` function, we can see the list of different languages assigned to books without duplicates.

Highest-rated language and lower-rated languages We analyze the languages that have been rated the best and the least to see if the language of a book can be useful in our analysis.

The number of books per language Using the `value_counts()` function, we can see the number of books per language.

[52]:
```
dfBookClear['language_code'].value_counts()
```

[52]:
```
eng      8459
en-US    1366
en-GB     208
spa       207
fre       138
ger        95
jpn        46
mul        19
zho        14
grc        11
por        10
en-CA       7
ita         5
lat         3
rus         2
swe         2
nl          1
srp         1
ara         1
msa         1
glg         1
wel         1
nor         1
enm         1
tur         1
gla         1
ale         1
Name: language_code, dtype: int64
```

[123]:
```
fig = px.pie(dfBookClear, values=dfBookClear['language_code'].value_counts(),␣
↪title="Distribution of languages", names=dfBookClear['language_code'].
↪unique(), color_discrete_sequence=px.colors.sequential.RdBu)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```

[128]:
```
fig = px.histogram(dfBookClear,title="Distribution of languages",␣
↪y="ratings_count", x="language_code")
fig.show()
```

We can see the top 5 most used languages:

[55]:
```
dfBookClear['language_code'].value_counts().head()
```

[55]:
```
eng      8459
en-US    1366
en-GB     208
spa       207
fre       138
Name: language_code, dtype: int64
```

We find the values in percentages:

[56]:
```
dfCountValueLang = dfBookClear['language_code'].value_counts(normalize = True).
 ↪head()
dfCountValueLang
```

[56]:
```
eng      0.797793
en-US    0.128831
en-GB    0.019617
spa      0.019523
fre      0.013015
Name: language_code, dtype: float64
```

We notice that the most commonly used language is 'eng'. And the least used languages are 'srp', 'nl', 'msa', 'glg', 'wel', 'ara', 'nor', 'tur', 'gla

Conclusion: A reader will be more likely to rate an English book.

[130]:
```
fig = px.scatter(dfBookClear, x="average_rating", y="num_pages", title="Average␣
 ↪VS Num pages", color='language_code')
fig.show()
```

Through the `scatter`, we see that the English language is everywhere on the graph, meaning that a lot more diverse votes have been allocated for that language. But this one is concentrated on the range of 3.5 and 4.5. We clearly see that for any language, the voting range remains between 3.5 and 4.5. We find the same result as seen above in our analysis.

Conclusion: We can predict that readers will likely score between 3.5 and 4.5.

3.3 Model Training

3.3.1 Choose predictors from columns

Before we start making predictions, we need to select the relevant columns so that our prediction is as close as possible to reality. Based on our analysis, we only keep columns where `average_ratings` is correlated: `num_pages`, `text_review_counts`, and `ratings_count`.

[58]:
```
dfTraining = dfBookClear.copy() # Copy of the data frame
dfTraining = dfTraining.drop(columns=['title', 'authors', 'language_code',␣
 ↪'publication_date']) # delete unused columns
target = "average_rating" # Value to predict
dfTraining
```

[58]:
```
         average_rating  num_pages  ratings_count  text_reviews_count
bookID
1                  4.57        652        2095690               27591
2                  4.49        870        2153167               29221
4                  4.42        352           6333                 244
5                  4.56        435        2339585               36325
9                  3.74        152             19                   1
…                     …          …              …                   …
45631              4.06        512            156                  20
45633              4.08        635            783                  56
45634              3.96        415            820                  95
45639              3.72        434            769                 139
45641              3.91        272            113                  12

[10603 rows x 4 columns]
```

3.3.2 Split dataset : Train & Test

In order to make our predictions, we need to separate our dataset into two: - "Train" is used to drive the model. - "Test" is used to evaluate the model.

To do this, we need to import the `train_test_split()` function from the `sklearn`:

[61]:
```
from sklearn.model_selection import train_test_split
train = dfTraining.sample(frac=0.8, random_state=1) # Train dataset with 80% of␣
 ↪the data.
test = dfTraining.loc[~dfTraining.index.isin(train.index)] # Test dataset with␣
 ↪20% of the data.
```

[62]:
```
train.shape
```

[62]:
```
(8482, 4)
```

[63]:
```
test.shape
```

[63]:
```
(2121, 4)
```

We chose the Linear Regression and Random Forest models to perform the training and evaluation of the dataset. These regression models are known to perform well on unclassified data.

3.3.3 Model Training : Linear Regression

We are now trying to drive the linear regression model from the "train" dataset. For this we import the `LinearRegression()` class from the `sklearn` library:

[64]:
```
from sklearn.linear_model import LinearRegression

modelReg = LinearRegression() # regression model
```

```
modelReg.fit(train[dfTraining.columns], train[target]) # model training
```

[64]:
```
LinearRegression()
```

3.3.4 Model Evaluation : Linear Regression

After driving the model, we can calculate the prediction error rate using the `mean_squared_error` function in the `sklearn` library:

[65]:
```
predictionsReg = modelReg.predict(test[dfTraining.columns]) # prediction for␣
↪the test data set.
```

Mean Squared Error

[66]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(predictionsReg, test[target]) # calculate the error between␣
↪predictions values and the reals values.
```

[66]:
```
2.6365742287339695e-31
```

Here we have an average accuracy of **2.64e-31**. This gives us a very low error rate. The closer the result is to 0, the better the predictions.

Mean Absolute Error

[67]:
```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(predictionsReg, test[target]) # calculate the error between␣
↪predictions values and the reals values.
```

[67]:
```
2.956641591819389e-16
```

The absolute average is **2.96e-16**.

Max Error

[68]:
```
from sklearn.metrics import max_error
max_error(predictionsReg, test[target])
```

[68]:
```
3.120006962550202e-15
```

Our maximum error value is **3.12e-15**.

Visualization of the prediction error To better represent the effectiveness of the prediction, we try to visualize it in a graphical form.

[131]:
```
dfVisual = pd.DataFrame({'Actual': test[target].tolist(), 'Prediction':␣
↪predictionsReg.tolist()})
fig = px.scatter(dfVisual, x='Actual', y="Prediction", trendline='ols',␣
↪title="Visualization of the prediction error",␣
↪trendline_color_override="#900C3F")
fig.show()
```

In this chart, we can see that all the points are on the trend line, which means that there are few error rates.

3.3.5 Model Training : Random Forest

We use the `RandomForestRegressor` function in the `sklearn` library:

[94]:
```
from sklearn.ensemble import RandomForestRegressor
# Initialiser le modèle avec certains paramètres.
modelForest = RandomForestRegressor(n_estimators=100, min_samples_leaf=10,␣
 ↪random_state=1)
# Adapter le modèle aux données.
modelForest.fit(train[dfTraining.columns], train[target])
```

[94]:
```
RandomForestRegressor(min_samples_leaf=10, random_state=1)
```

3.3.6 Model Evaluation : Random Forest

We generate Random Forest predictions from the test game.

[95]:
```
predictionsForest = modelForest.predict(test[dfTraining.columns])
```

Mean Squared Error

[96]:
```
mean_squared_error(predictionsForest, test[target])
```

[96]:
```
0.00106413165189238
```

We obtain a value almost equal to 0 but always higher than the result of the linear regression model.

Mean Absolute Error

[97]:
```
mean_absolute_error(predictionsForest, test[target])
```

[97]:
```
0.0028897696604980685
```

The absolute average is 0.002.

Max Error

[98]:
```
max_error(predictionsForest, test[target])
```

[98]:
```
0.609719135272063
```

Our maximum error value is 0.61.

[132]:
```
dfVisual = pd.DataFrame({'Actual': test[target].tolist(), 'Prediction':␣
 ↪predictionsForest.tolist()})
fig = px.scatter(dfVisual, x='Actual', y="Prediction", trendline="ols",␣
 ↪title="Visualization of the prediction error",␣
 ↪trendline_color_override="#900C3F")
fig.show()
```

In this graph, we observe that points are outside the trend line.

3.4 Conclusion

We can conclude that the linear regression model is more reliable than the random forest model. There is more chance to predict a result close to the real, it will generally make less error.