# Import libraries

import numpy as np  
import pandas as pd  
from datetime import datetime  
from fuzzywuzzy import process, fuzz  
import matplotlib.pyplot as plt  
import matplotlib.ticker as mtick  
import seaborn as sns  
import plotly.express as px  
import warnings  
warnings.filterwarnings("ignore")  
  
from pandas\_profiling import ProfileReport

When we first tried to load the csv file into a dataframe, we got this error ParserError: Error tokenizing data. C error: Expected 12 fields in line 3350, saw 13

Meaning that at row 3350, there is an extra column. We modified the file trough excel where we’ve noticed that the columns were shifted to the right. The same for the rows 4704,5879 and 8981

# IMPORT DATA

#import book dataset

df=pd.read\_csv("C:/Users/romai/Documents/DSTI/4-Python Labs/Project/Python/dataset/books\_v2.csv", index\_col="bookID", on\_bad\_lines = 'error')

# EXPLORATORY DATA ANALYSIS

1. Create a EDA summary report using Pandas Profiling
2. Describe the data
3. Look for missing values
4. Look for “weird” values (outliers? errors?)
5. Check if data are balanced (if not, model can be biaise

## 1. Initial EDA summary report

using Pandas Profiling library

# Generate the profile report with Pandas Profiling  
profile = ProfileReport(df,title="Summarization of book rating data")

# Generate the dataset profile  
# This is a nice and simple way to document the data  
profile.to\_notebook\_iframe()

## 2. Traditional EDA

#Return first 5 rows of the dataset  
df.head(5)

Publication dates have different formats

isbn and isbn13 relevant for our model? Does a barcode can predict the ratings? In general, related to year of publication and identification of book (information that we already have) + we strongly assume that a reader doesn’t look at that information when he/she evaluates a book (neither to select a book in order to read it) => we decide to remove it from our future model

Authors: single author vs multiple authors -> new column “Authors\_2” = Single/Multiple -> boolean

df.shape

(11127, 11)

df.columns

Index(['title', 'authors', 'average\_rating', 'isbn', 'isbn13', 'language\_code',  
 ' num\_pages', 'ratings\_count', 'text\_reviews\_count',  
 'publication\_date', 'publisher'],  
 dtype='object')

We can see a space upfront the column name “num\_pages”

=> **Action (Feature Eng)**: remove space upfront ” num\_pages” \*\*\*

df.info()

<class 'pandas.core.frame.DataFrame'>  
Int64Index: 11127 entries, 1 to 45641  
Data columns (total 11 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 title 11127 non-null object   
 1 authors 11127 non-null object   
 2 average\_rating 11127 non-null float64  
 3 isbn 11127 non-null object   
 4 isbn13 11127 non-null float64  
 5 language\_code 11127 non-null object   
 6 num\_pages 11127 non-null int64   
 7 ratings\_count 11127 non-null int64   
 8 text\_reviews\_count 11127 non-null int64   
 9 publication\_date 11127 non-null object   
 10 publisher 11127 non-null object   
dtypes: float64(2), int64(3), object(6)  
memory usage: 1.3+ MB

#Columns type  
df.dtypes

title object  
authors object  
average\_rating float64  
isbn object  
isbn13 float64  
language\_code object  
 num\_pages int64  
ratings\_count int64  
text\_reviews\_count int64  
publication\_date object  
publisher object  
dtype: object

#Is there any NA values in the dataset?  
df.isna().sum()

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

#check for row duplication  
df.duplicated().sum()  
#no duplicate row but let's look at such as title, authors, etc (later) that could be repeated

0

#Exploring statistical summary for numerical columns and some information for non-numerical columns   
df.describe(include = "all")

* Books with no pages (76 of them) -> how it’s possible ? replace by mean in FE
* Average rating equal to 0 is possible
* Ratings\_count = 0 -> irrelevant if there is an average rating not null- remove in FE
* Unique title < total count -> duplicate titles - how to deal with that?

#What are the books that have 0 pages?   
pages = df[' num\_pages']  
df[pages == 0]

76 rows × 11 columns

=> **Action (Feature Eng)**: replacing the ‘num\_pages’ 0 values with the mean and checking the result \*\*\*

/! What does it mean NOT A BOOK in the authors column? bookID:3593, 3599 Let’s look at it in more details if we have others “NOT A BOOK” authors

df[df.authors == 'NOT A BOOK']  
#there are 5 books with "Not a Book" which are Audio Book -> suggestion to drop them

=> **Action (Feature Eng)**: remove those 5 rows \*\*\*

#describe our target variable average\_rating and plot it  
df.average\_rating.describe()

count 11127.000000  
mean 3.933631  
std 0.352445  
min 0.000000  
25% 3.770000  
50% 3.960000  
75% 4.135000  
max 5.000000  
Name: average\_rating, dtype: float64

plt.figure(figsize=(10,10))  
sns.histplot(df['average\_rating'], kde = True, bins = 20, palette = "hls" )  
plt.xticks(rotation = 90)  
plt.ylabel('Frequency')  
plt.grid()

#median   
print("Median (average\_rating):", df.average\_rating.median())  
print("Mean (average\_rating): ", df.average\_rating.mean())

Median (average\_rating): 3.96  
Mean (average\_rating): 3.9336308079446125

#check the normality of the distribution  
print(f"Skewness: {df['average\_rating'].skew()}")  
print(f"Kurtosis: {df['average\_rating'].kurt()}")

Skewness: -3.638311410554902  
Kurtosis: 36.721776547302916

Asymetry in distribution with outliers (https://medium.com/@atanudan/kurtosis-skew-function-in-pandas-aa63d72e20de): \* A negative skewness value in the output indicates an asymmetry in the distribution corresponding to row 2 and the tail is larger towards the left hand side of the distribution.

* High kurtosis in a data set is an indicator that data has heavy outliers.

=> **Action (Feature Eng)**: create 3 categories ([0-2]-[3]-[4-5]) \*\*\*

### Study of relationships between variables

sns.pairplot(df) #numeric variables

## 3. Data Cleaning

### Columns title

#Renaming ' num\_pages' in order to remove the extra spaces and displaying columns names  
df.rename(columns={' num\_pages':'num\_pages'}, inplace=True)

df.columns

### Languages

For the language\_code column, replacing the value eng-US, en-CA and eng-GB with the value eng

df.language\_code.value\_counts()

eng 8911  
en-US 1409  
spa 218  
en-GB 214  
fre 144  
ger 99  
jpn 46  
mul 19  
zho 14  
grc 11  
por 10  
en-CA 7  
ita 5  
enm 3  
lat 3  
swe 2  
rus 2  
srp 1  
nl 1  
msa 1  
glg 1  
wel 1  
ara 1  
nor 1  
tur 1  
gla 1  
ale 1  
Name: language\_code, dtype: int64

df['language\_code'] = df['language\_code'].replace('en-US','eng')  
df['language\_code'] = df['language\_code'].replace('en-GB','eng')  
df['language\_code'] = df['language\_code'].replace('en-CA','eng')  
#or #df.language\_code = books\_df.language\_code.replace({'en-US':'eng', 'en-GB':'eng', 'en-CA':'eng'})

df.language\_code.value\_counts()

eng 10541  
spa 218  
fre 144  
ger 99  
jpn 46  
mul 19  
zho 14  
grc 11  
por 10  
ita 5  
lat 3  
enm 3  
rus 2  
swe 2  
nl 1  
ara 1  
srp 1  
msa 1  
glg 1  
wel 1  
nor 1  
tur 1  
gla 1  
ale 1  
Name: language\_code, dtype: int64

### Publisher columns, many publishers have different spellings

pub = df.publisher.unique()  
elements = list(pub)  
# elements = ['vikash', 'vikas', 'Vinod', 'Vikky', 'Akash', 'Vinodh', 'Sachin', 'Salman', 'Ajay', 'Suchin', 'vIkas', 'salman', 'salMa', 'Akash', 'vikahs']  
  
results = [[name, [], 0] for name in elements]  
  
for (i, element) in enumerate(elements):  
 for (j, choice) in enumerate(elements[i+1:]):  
 if fuzz.ratio(element, choice) >= 90:  
 results[i][2] += 1  
 results[i][1].append(choice)  
 results[j+i+1][2] += 1  
 results[j+i+1][1].append(element)  
  
data = pd.DataFrame(results, columns=['name', 'duplicates', 'duplicate\_count'])  
  
data = data.sort\_values(by=['duplicate\_count'],ascending=False)

to\_remove = []  
test = {}  
  
for i,j in data.iterrows():  
 if data.name[i] in list(test.keys()):  
 for x in j.duplicates:  
 if x not in test :   
 test[x]=test[data.name[i]]  
 else :   
 test[data.name[i]] = data.name[i]  
 for x in j.duplicates:  
 if x not in test :   
 test[x]=data.name[i]  
# print(data.name[i], "\*\*\*\*", x,"--" , data.name[i])  
   
  
   
# print(test)  
data['pub'] = data['name'].map(test)  
display(data)

df.publisher.value\_counts()

Vintage 318  
Penguin Books 261  
Penguin Classics 184  
Mariner Books 150  
Ballantine Books 144  
 ...   
University of Calgary Press 1  
Marlowe & Company 1  
University Press of America 1  
Abstract Studio 1  
Sounds True 1  
Name: publisher, Length: 2292, dtype: int64

group=df.groupby('publisher')['title'].count().reset\_index().sort\_values('title', ascending = False).set\_index('publisher')  
group

2292 rows × 1 columns

### Date Column

\*\*move to FEATURE ENGINEERING

# Changing the publication\_date colummn type to a date format   
df['publication\_date'] = pd.to\_datetime(df['publication\_date'], dayfirst = False, yearfirst = False, errors = 'coerce')

df.head(5)

df.isna().sum()

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 2  
publisher 0  
dtype: int64

df.loc[df.publication\_date.isna(),:]

those 2 dates are unreal in the sense that # November doesn’t have 31 days (11/31/2000) # June doesn’t have 31 days (31/06/1982)

=> replace by correct date for those 2 rows

#31373 published October 31, 2000 by Bantam Books (https://www.goodreads.com/book/show/31373.In\_Pursuit\_of\_the\_Proper\_Sinner?ac=1&from\_search=true&qid=3A5Zotzgsy&rank=1)  
  
#45531 published June 30, 1982 by Folio histoire (https://www.goodreads.com/book/show/45531.Montaillou\_village\_occitan\_de\_1294\_1324)

#correcting 2 NaT by correct date  
df.loc[31373,"publication\_date"]='10/30/2000'  
df.loc[45531,"publication\_date"] ='6/30/1982'  
df.loc[[31373,45531]]

# Splitting date column into month and year columns - move in FE   
df['month'] = df['publication\_date'].dt.month  
df['year'] = df['publication\_date'].dt.year

# bookID: 31373 where the date value before converting the column into a datetime type was 11/31/2000 (30 days for November)  
# bookID: 45531 where the date value before converting the column into a datetime type was 6/31/1982 (30 days for June)  
# Dropping those 2 rows   
#df = df.drop(index=df.loc[df.publication\_date.isna(),:].index)

---------------------------------------------------------------------------

# Next Steps

1. In the correlation section
   * Avg rating // language (visual)
   * Avg rating for top published authors over time (year) (bof)
   * Avg rating for top publishers over time (bof)
   * Corr month publication // avg rating / rating counts / text review
   * Corr year publication // avg rating / rating counts / text review
2. jointplot for text\_reviews\_count and ratings\_count –> no histogram
3. After dropping all outliers, how does the heatmap look like?
4. Transforming categorical columns into numerical columns
   * Title
   * Authors
   * Language
   * Publisher
5. New column that identifies if the book has been written by a single or multiple authors Or trying to select only the first author of the book
6. Identify similar publishers ???
7. Dropping the rows where the avg rating is 0
8. Dropping the rows where rating counts and text review counts are set to 0