*Une image contenant texte, bâtiment, brique, rayon

Description générée automatiquement*

*Books rating prediction*

*Une image contenant texte, intérieur

Description générée automatiquement Includes web scrap with selenium*

**TABLE OF CONTENTS**

Introduction

Import Libraries

**Part 1 Clean data and Web scrap…………………………………………………………………………………………………….p.5**

1. Clean the data
2. Clean data – step 1
3. Read the dataset
4. Locate shifted rows
5. Shift rows left
6. Observations on data frame
7. Check and transform data types
8. Web scrap

**Part 2 Data cleaning and Analysis………………………………………………………………………………………………….p.9**

1. Clean data
2. Drop
3. Deal with duplicates
4. Categories management
5. Language\_code cleaning
6. Handle Null values
7. Dates formatting
8. Data transformation
9. Categories
10. Authors
11. Occurrences : calculated columns
12. Data analysis

**Part 3 : Features engineering and Machine Learning………………………………………………………………..……p.18**

1. Features selection and engineering
2. Features selection
3. Correlation matrix
4. Remove variables
5. Remove outliers
6. Normalize
7. Machine Learning
8. ML including the ‘categories’

Linear Regression

Random Forest Regressor

Ridge

1. ML without the ‘categories’

Linear Regression

Random Forest Regressor

Ridge

Introduction

This report aims at, step by step, demonstrating the process to web scrap, clean data, analyse them and finally train the models . The target variable that needs to be predicted corresponds to a regression model. Therefore 3 models fitting with continuous variable were tested in the notebook.

The IDE used for this project is VS Code

Programming language : Python 3.9.13

Environment : test

Hosted @ : [GitHub repository](https://github.com/SophieRouiller8/Books_rating_pred_python)

Libraries installed/imported :

*#Import Librairies*

import **pandas** as **pd**

import **numpy** as **np**

import **os**

import **glob**

import **csv**

*#import Selenium for webscraping*

import selenium

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.support.wait import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

*# visualization*

import **seaborn** as **sns**

import **matplotlib**.**pyplot** as **plt**

import **plotly**.**graph\_objects** as **go**

import **plotly**.**express** as **px**

%matplotlib inline

**sns**.**set**(style= 'darkgrid')

**sns**.**set\_palette**('deep')

*#transformations*

from **sklearn** import **preprocessing**

from **sklearn**.**preprocessing** import **StandardScaler**

from **sklearn**.**preprocessing** import **LabelEncoder**

from **sklearn**.**preprocessing** import **OneHotEncoder**

*# machine learning*

from **sklearn**.**model\_selection** import **train\_test\_split**

from **sklearn**.**linear\_model** import **LogisticRegression**

from **sklearn**.**linear\_model** import **LinearRegression**

from **sklearn**.**linear\_model** import **SGDRegressor**

from **sklearn**.**svm** import **LinearSVR**

from **sklearn**.**ensemble** import **RandomForestRegressor**

from **sklearn**.**ensemble** import **GradientBoostingRegressor**

from **sklearn**.**ensemble** import **AdaBoostRegressor**

from **sklearn**.**neighbors** import **KNeighborsRegressor**

from **sklearn**.**linear\_model** import **Perceptron**

from **sklearn**.**model\_selection** import **GridSearchCV**

from **sklearn**.**tree** import **DecisionTreeRegressor**

from **sklearn**.**svm** import **SVR**

from **sklearn** import **metrics**

from **sklearn**.**metrics** import **accuracy\_score**

from **sklearn**.**metrics** import **confusion\_matrix**

from **sklearn**.**metrics** import **classification\_report**

from **sklearn**.**metrics** import **mean\_squared\_error**

from **sklearn**.**metrics** import **roc\_auc\_score**

from **sklearn**.**metrics** import **f1\_score**

from **sklearn**.**metrics** import **mean\_absolute\_error**

from **sklearn**.**metrics** import **r2\_score**

from **sklearn**.**metrics** import **precision\_score**, **recall\_score**

**Part 1 : Clean data and Web scrap**

1. **Cleaning Data – Step 1**
2. First step, read the csv file

df0 = **pd**.**read\_csv**('books.csv', sep=';')

1. Then visualize the data frame unique values :

Une image contenant texte, reçu

Description générée automatiquement

Here we observe that an unnamed column has been added at the end of the data frame containing 4 lines.

In order to localize them in the data frame, I display the column ‘language\_code’ which contains the least number of unique values. This will reduce the research.

Now, we can clearly point out the issue :

Une image contenant table

Description générée automatiquement

4 lines were shifted to the right by 1 column. The isbn13 of those 4 columns appears in the ‘language\_code’ column.

1. Locate shifted rows

df0.loc[df0['language\_code'].**isin** (['9,78085E+12', '9,78156E+12', '9,78159E+12', '9,78067E+12'])]

Une image contenant table

Description générée automatiquement

The column ‘authors’ has been split into 2, this may have been caused by a comma in the cells.

1. Shift rows left

df0.iloc[[3348, 4702, 5877, 8979], 3:13] = df0.iloc[[3348, 4702, 5877, 8979], 3:13].**shift**(periods=-1, axis="columns")

I check that the action has been completed correctly :

Une image contenant table

Description générée automatiquement

1. Data frame Information

Une image contenant table

Description générée automatiquement

I can observe 3 issues :

* The header ‘num\_pages’ is shifted to the right so I need to remove the space in front, through renaming the column :

df0 = df0.**rename**(columns={"  num\_pages": "num\_pages"})

* The ‘unnamed:12’ column needs to be dropped.

df0.**drop**('Unnamed: 12', axis=1, inplace=True)

* Several columns have the same number of null values (51), before making any action, I visualize them :

Une image contenant table

Description générée automatiquement

As all columns have missing values, none of these rows will be kept. So I drop them.

1. Data types

As for some columns, the data types are not adapted to the types of data, I convert the data types to the right types :

df0 = df0.**astype**({"average\_rating": **float**, "num\_pages": **int**, "text\_reviews\_count": i**nt**,"ratings\_count": **int**, })

The isbn13 conversion raises an error, so I will deal with this column manually, the other column now

1. **Web scrap the books categories (genres)**

Background : Goodreads does not deliver any new access to their API. Therefore, the best alternative is to scrap the information from their website.

options = Options()

*#options.add\_argument('headless')*

website='https://www.goodreads.com'

path=r"C:\Users\sophi\OneDrive\Bureau\python\_project\chromedriver.exe"

driver=webdriver.Chrome(path,chrome\_options=options)

*#acceptedcookies=False*

part="\_part9.csv"

def **autosearch**(isbn13):

*#global acceptedcookies*

    driver.get(website)

    driver.implicitly\_wait(5)

    searchbar=driver.find\_elements(By.ID,'sitesearch\_field')[0]

    searchbuttonposition=driver.find\_elements(By.CLASS\_NAME,'submitLink')[0]

    searchbutton=searchbuttonposition.find\_elements(By.TAG\_NAME,'img')[0]

    searchbar.send\_keys(isbn13[:-1])

    searchbutton.click()

    driver.implicitly\_wait(8)

    try:

        genrelist=driver.find\_elements(By.CLASS\_NAME, 'elementList')

        li\_list=genrelist[0].find\_elements(By.CLASS\_NAME, 'left')[0]

        categories=li\_list.find\_elements(By.TAG\_NAME, 'a')[0].text

*#deduplicated=max(set(categories), key = categories.count)*

**print**(categories)

        return(categories)

    except:

**print**('NA')

        return('NA')

bookDF=**pd**.**read\_csv**(r"C:\Users\sophi\DSTI\Project Python Lab\Project DA\Books\_for\_analysis.csv +part",sep=";")

**print**(bookDF.columns)

*#create a new column + iterate for each line with the above function*

df0['categories']=df0['isbn13'].**apply**(lambda x:**autosearch**(**str**(x)))

As I needed to scrap more than 11000 information, so many calls in one go on their website could create some bottlenecks. I split the dataset into 12 parts in order to scrap by batch.

To control that Goodreads website returned the information effectively, I inserted a print function. This allowed me so control any potential rejection from their server.

**Part 2 Data cleaning and analysis**

Load the file\_4\_prediction dataset and visualize the 2 first and 2 last rows

Une image contenant texte

Description générée automatiquement

Check the data frame information

Une image contenant table

Description générée automatiquement

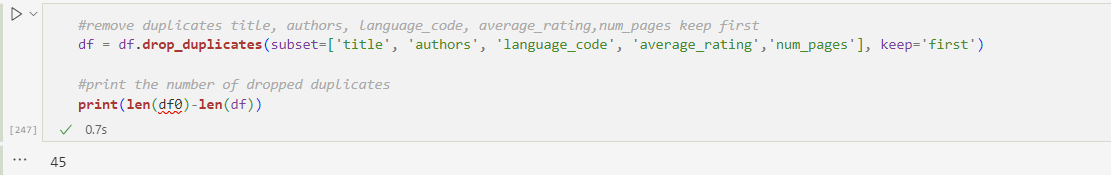
All columns have values except ‘categories’ which has missing values (397), missing information that could not be provided by Goodreads.

A Data cleaning

1. Drop isbn as not needed for further analysis, isbn13 is kept for the moment.
2. Duplicates :

There are some duplicates in the books ‘ column. 2 choices are possible :

* Keep them : however this may create fake statistics.
* Remove them from the data frame : I chose this alternative, selecting duplicates having the same book’s title, authors, language\_code, average rating number of pages.



45 books were removed from the data frame.

1. Categories

As previously stated, 397 categories are missing. Before removing them, I check how many have a total number of ratings count above the minimum of the column ratings count. This in order to keep accuracy in the dataset as the number of ratings has an important weight :

Une image contenant texte

Description générée automatiquement

The dataset has now 10634 rows.

I check the unique values of each columns :

Une image contenant texte, reçu, capture d’écran

Description générée automatiquement

120 unique values in the ‘categories’ column will not be helpful in the further steps. I will therefore factorize them, grouping them in order to reduce the number categories. This is done through checking on Goodreads website and ebooks.com to get the head categories and sub categories.

I could reduce to 16 categories:

Une image contenant texte

Description générée automatiquement

1. Language\_code

I group all the English language code under one umbrella : ‘en’.

Une image contenant texte

Description générée automatiquement

1. Replace the null values (0) in the dataset.

The describe function allows me to point out that the minimum for the columns average rating, num\_pages, ratings count and text reviews count is 0. After checking, this value is spread in the dataset and does not correspond to specific rows. Therefore I replace those null values by the mean value of each variable.

Une image contenant texte

Description générée automatiquement

29 null-values were replaced.

f) Format the dates

The publication date column has a type ‘object’, I convert the column to a date format ('%m/%d/%Y')

And check if any rows have missing values. I find 2 rows :



I treat them manually looking for their publication date on internet. This does not take long.

As the day and month of the publication will not be useful, I create a variable ‘publi\_year’ in place of ‘publication\_date’.

This new column will allow me to calculate the age of the books, this information will be useful in the analysis phase.

Une image contenant texte

Description générée automatiquement

Now that the dataset has been cleaned, removing NaN values, replacing null values and converting the necessary data to the right type, I can visualize the statistics of the dataset :

Une image contenant table

Description générée automatiquement

Some observations of the above statistics :

Average rating : the mean is quite high 3.94 out of a maximum of 5. And the standard deviation is very low (0.28). the median is very close to the mean, here the distribution is symmetric.

Num\_pages, ratings\_count and text\_reviews\_count are right positive skewed. The standard deviation is high, the medians are lower than the mean so we can anticipate some outliers in this columns.

1. Data transformation
2. Categories

In order to be able to make some calculations in the analysis phase on the ‘categories’ column, I transform the categorical values into numerical :

Une image contenant texte

Description générée automatiquement

1. Authors

Assuming that the author corresponds to the first part of the column (left side), and that the co-author is not an author but rather a designer or translator… (I did some random checks manually)

I split the author and co-author and create a column ‘author’, dropping the columns ‘authors’ and ‘co-authors’.

Une image contenant texte

Description générée automatiquement

1. Occurrences

As per the below summary, we can obverbe that authors and publishers can occur several times in the dataset. It could be interesting to point out the influence of the occurrences over the books rating.

Une image contenant table

Description générée automatiquement

William Shakespeare is the most prolific writer, and the publisher ‘Vintage’ seems to have published many books.

I create 2 calculated columns : author and publisher occurrence

Une image contenant texte

Description générée automatiquement

1. Analysis

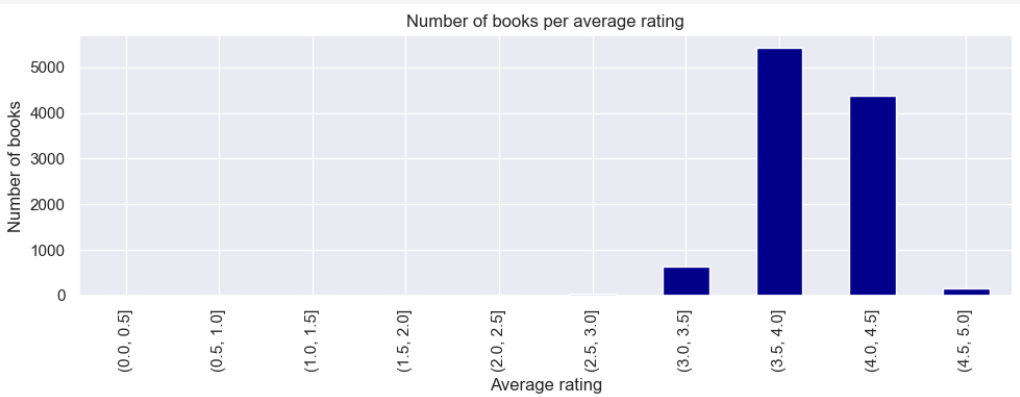
**Graph 1** : visualize the number of books published per year versus the ratings counts per year :

Une image contenant table

Description générée automatiquement

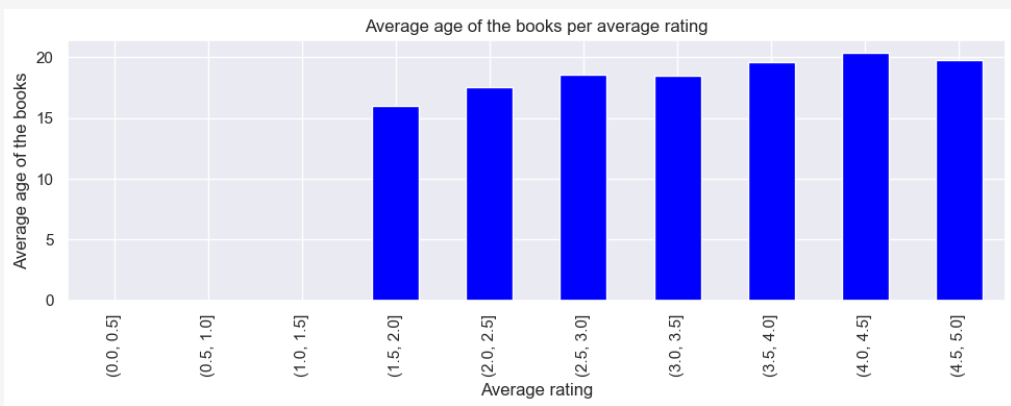
The ratings count curve follows the trends of the number of published books per year. It is impressive to visualize the prolific production of books in the years around 2000. Is the dataset built around this period? Or is it representative of the books production globally?

**Graph 2** : Number of books per average rating :



As already figured out in the descriptive statistics, the majority of the books are rated between 3.5 and 4.5. This will impact the machine learning results.

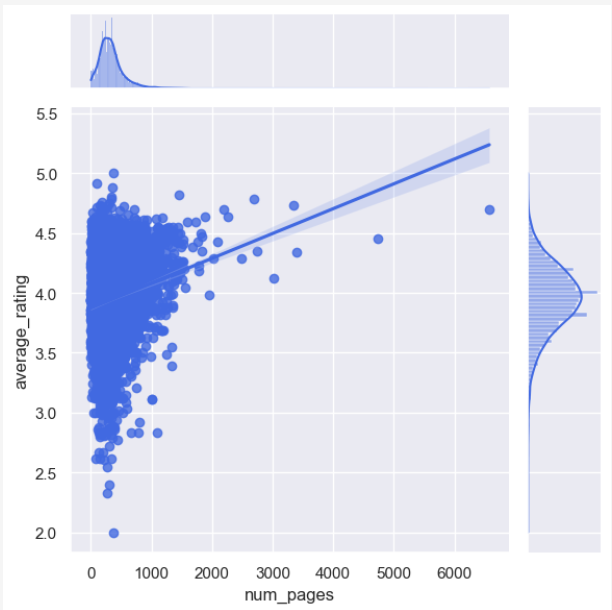
**Graph 3** : Age of the books per average rating :



The average rating is higher when the books is older. However this is not sufficient to make any conclusion. In the 20+ books, some were published more than 100 years ago and are part of the ‘Classics’ category.

**Graph 4** : the correlation between the number of pages in the books and the average rating.

Here I try to answer to question : Do people prefer ‘light books’ or ‘heavy’ books ?

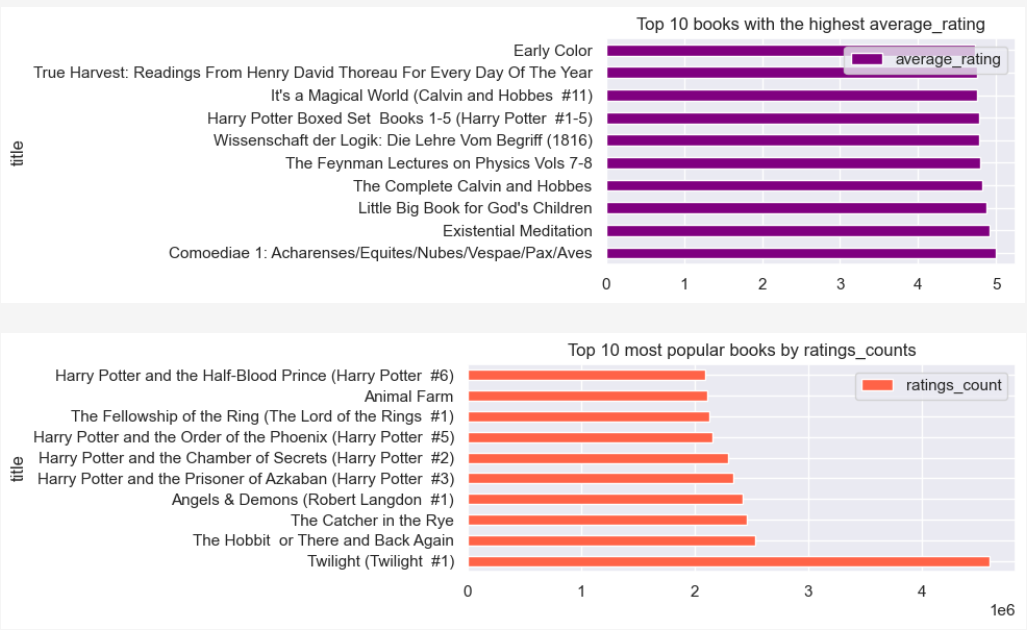


We can observe on the above graph that the majority of the books are graded in the neighborough of 4 and have less than 1000 pages, mostly less than 500 pages on average.

As previously stated in the description statistics, we can now visualize the outliers in the num\_pages column.

The average rating graph is symmetrically distributed with a concentration around 4; mostly between 3.8 and 4.2. we can observe an increase of the average rating correlated with the number of pages.

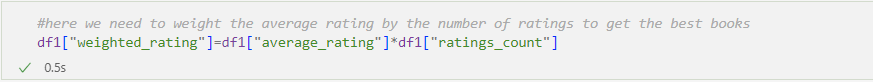
**Graph 5 and 6** : Books ratings and popularity

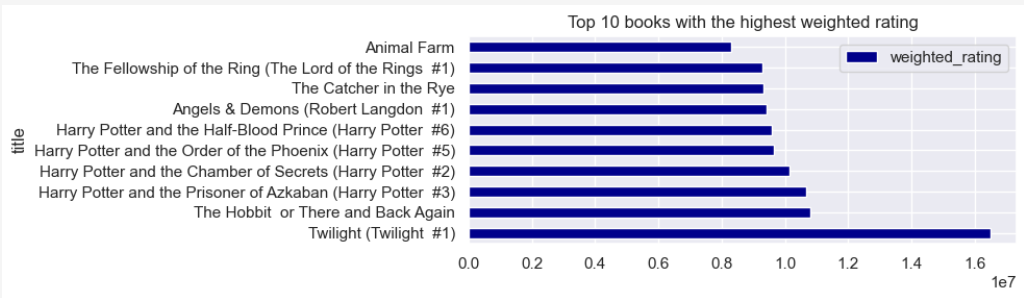


The most popular books based on the ratings counts, are not the ones rated the highest. We clearly figure out that although for instance, Twilight gained the highest number of ratings, the top 1 book is Comoediae. Except Harry Potter(#1-5), none of the most popular books are ranked in the top 10 per average rating.

The most popular book is of category Fiction, while the top1 average rating is of category Classics. A part some exceptions, the best ranked in terms of average rating are ‘more confidential’ books, Classics, philosophy...while the most popular books are of category Fiction (Twilight, Harry Potter, Lord of the ring…).

In order to weigh the average rating, I created a column named ‘weighted rating’ to visualize how the ratings count impacts the ranking**, graph 7**:

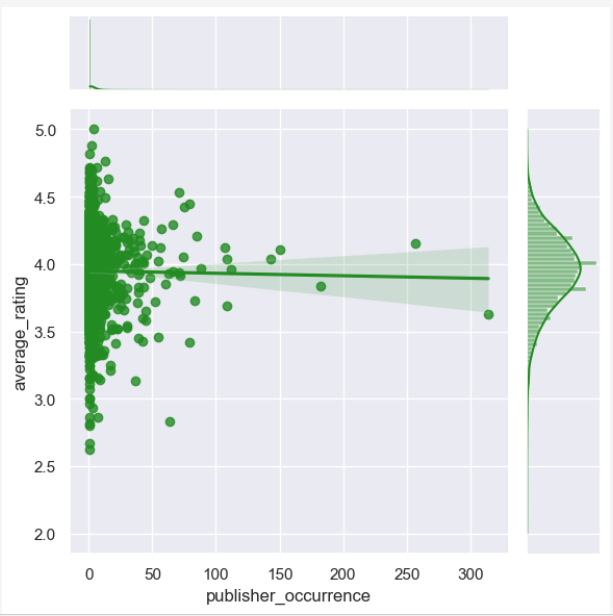
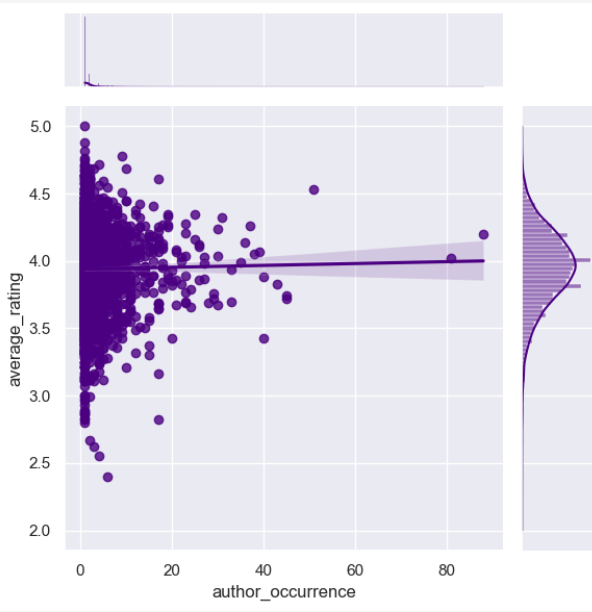




The ranking order is slightly changed, for instance ‘Angels and demons’ slipped from the 3rd position to the 7th position, however logically, the same books appear in the ranking per weighted rating.

The following graphs are related to the authors, publishers based one the same metrics than the books, and logically follow the same rankings (graphs 8 to 14)

**Graph 15 and 16 :** Influence of the author and publisher occurrence on the average rating

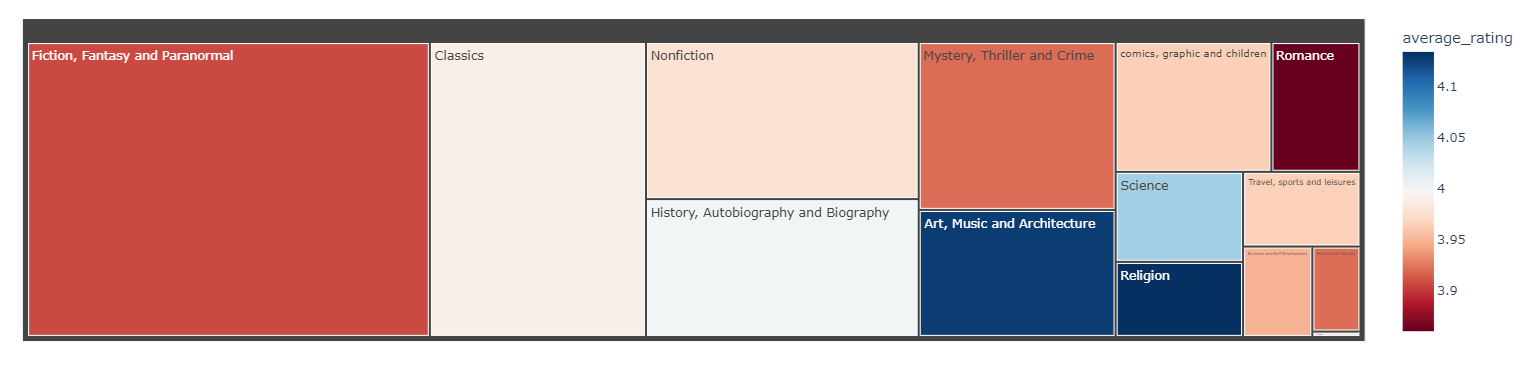


First observation, the number of different publishers is inferior to the number of authors. Whereas in many cases an author has 1 publisher, publishers have contracts with several authors.

Both publishers and authors are right skewed, this highlights outliers. A general trend, the occurrence for authors does not affect the average rating, which remains stable around 4, except for some outliers. For publishers, the average rating tends to slightly decrease and remains below 4.

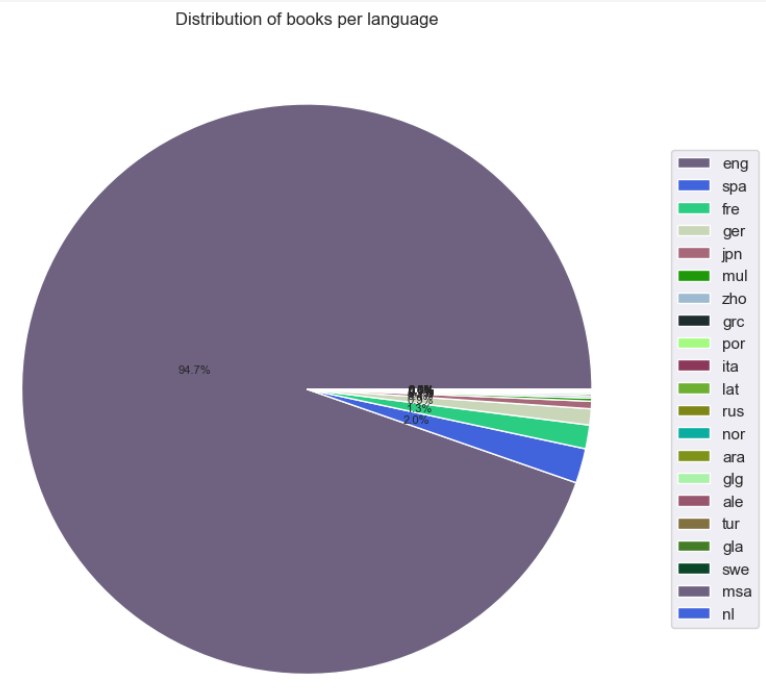
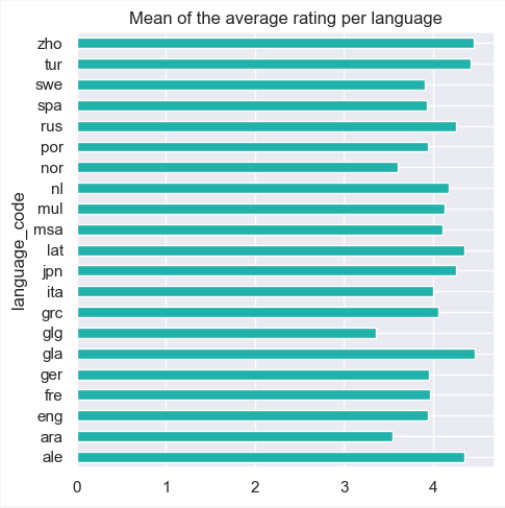
We cannot notice any positive influence of these occurrences over the average rating.

**Graph 17 :** Tree map of categories per average ratings



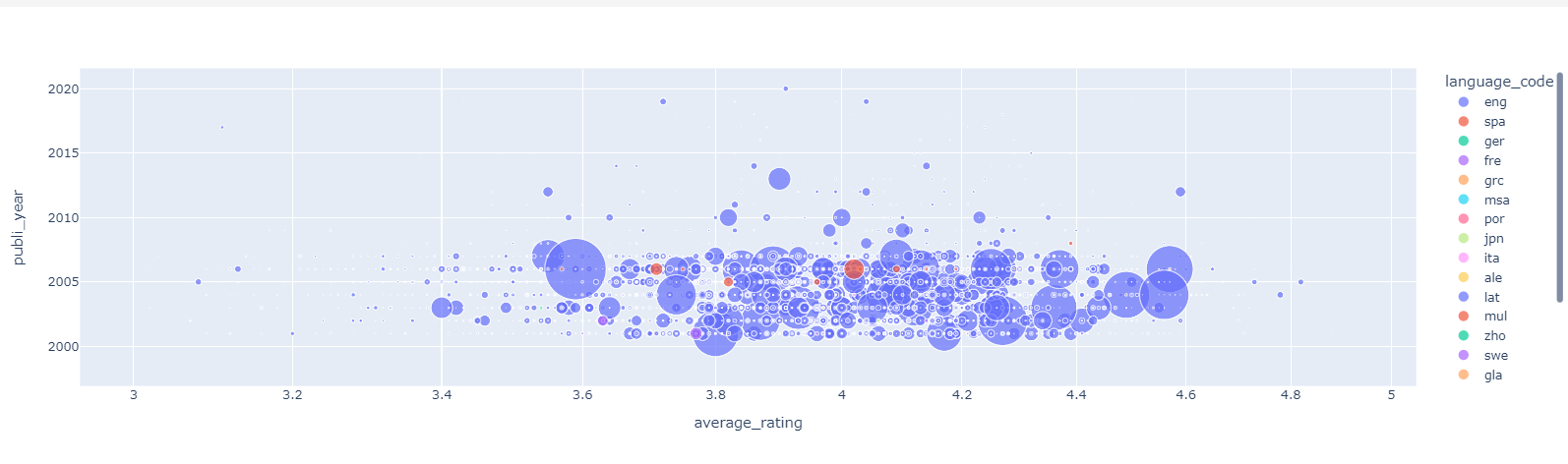
This graph is in line with the graph 5, indeed, the best graded books are not the most popular categories. The categories Religion, Art Music and Architecture and science are the best graded(above 4). Their audience looks for specific books and targets meticulously the books they want to read. Their chance of being satisfied is therefore higher than a large none insider audience who may pick books with no specific goal in mind.

**Graph 18 and 19 :** Influence of the language code on the average rating



Although over-represented, the books in English language do not get the best average rating. The larger is the audience the most mitigating is the rating.

**Graph 20 :** Interactive map of the books average rating published between 2000 and 2020 per category, language, sized per weighted rating. Hover your mousse on the plot to discover the information (in the Jupyter Notebook).



**Part 3 : Features engineering and Machine Learning**

As the variable average rating is the variable that I need to predict, the regression will be used. I chose 3 models (the most common) :

* Linear regression
* Random Forest Regressor
* Ridge

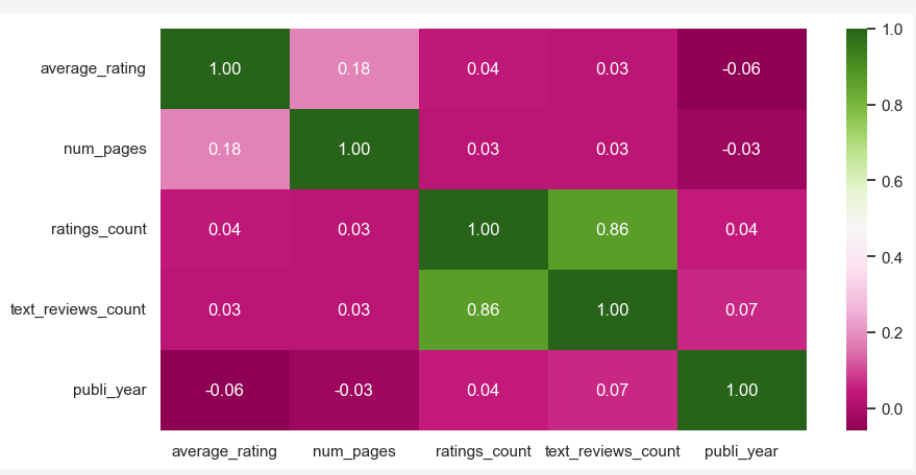
**A) Features selection and engineering**

1. Create features lists to separate the categorical from the numerical variables

Une image contenant texte

Description générée automatiquement

1. Then visualize the correlation between the numerical variables with a heatmap



Except text\_reviews\_count and ratings\_count which are strongly correlated, the level of correlation is low. Num\_pages and average rating are the most correlated compared to the other variables.

1. Remove variables

I remove isbn13 from the categorical\_features as not necessary for training the model. I remove author\_occurrence and publisher\_occurrence from the numerical features as the author and the publisher will be transform into dummies for the models. As text\_reviews count is strongly correlated to ratings\_count, I choose to keep ratings count and remove text\_reviews count. Indeed, a person who wrote an comment about a book will surely give a rating to the book, however the contrary is not true. That is why I prefer keeping ratings\_count.

1. Remove outliers

As previously observed during the analysis phase, ratings counts and num\_pages have outliers. They may lead to wrong results of the models. So I remove the outliers, using IQR, looking at eliminating the first and 3rd quartiles lower and upper bound values.

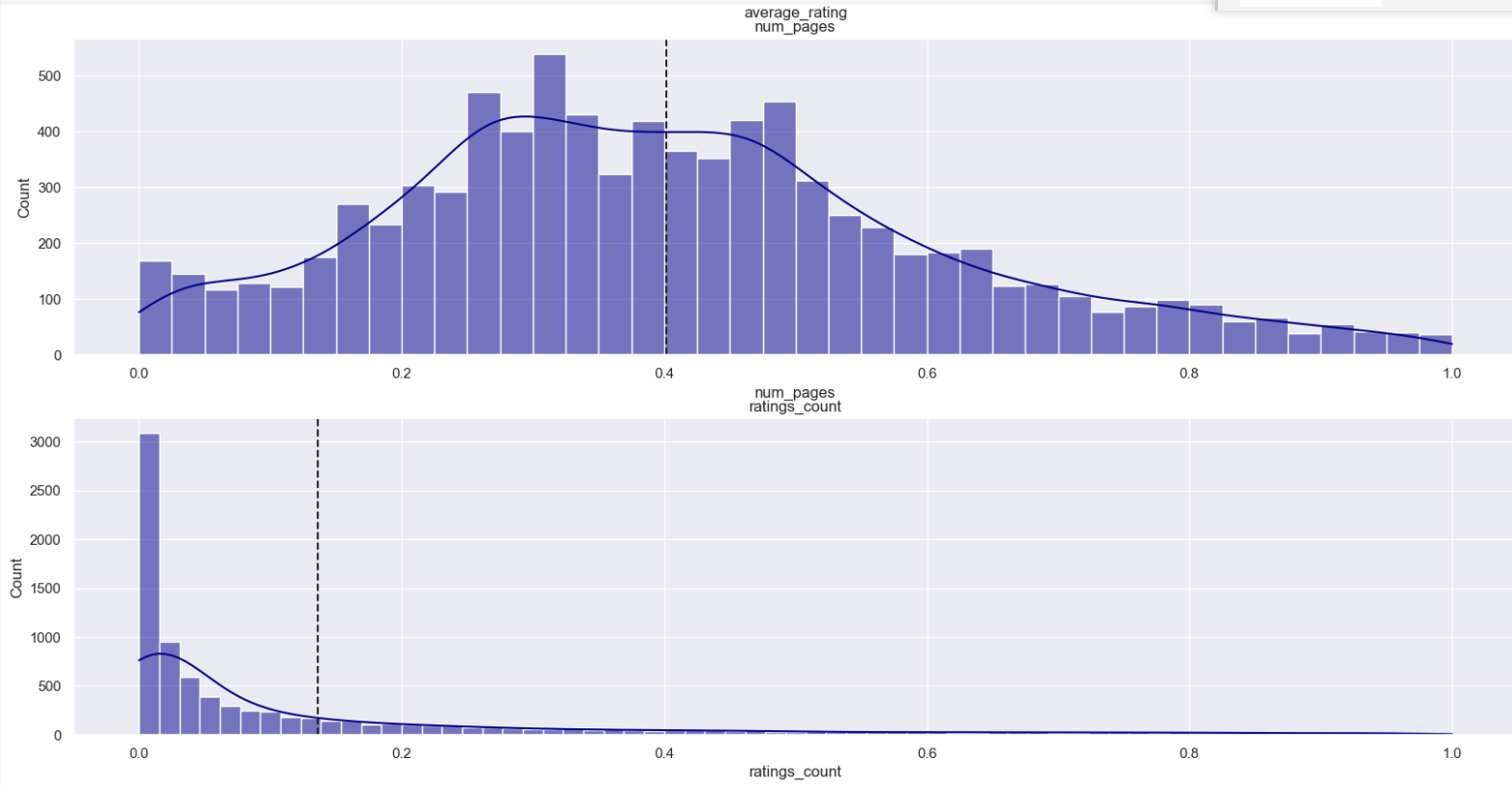
Une image contenant texte

Description générée automatiquement

1. Normalization

I then normalize the data of the same variables in order to get a common scale for both variables.

Une image contenant texte

Description générée automatiquement

Now that the feature engineering is done, I can proceed to transforming the categorical features into dummies and split the dataset into train, test set for training the models (20% test, 80% train) :

Une image contenant texte

Description générée automatiquement

1. **Machine Learning**

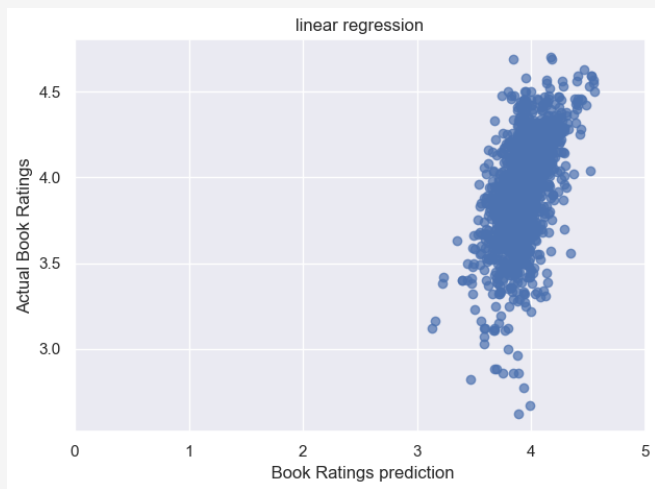
Results comparison of the 3 models :

1. **ML including the ‘categories’**

(notebook ‘Project\_part3\_ML\_model(withcategories)’

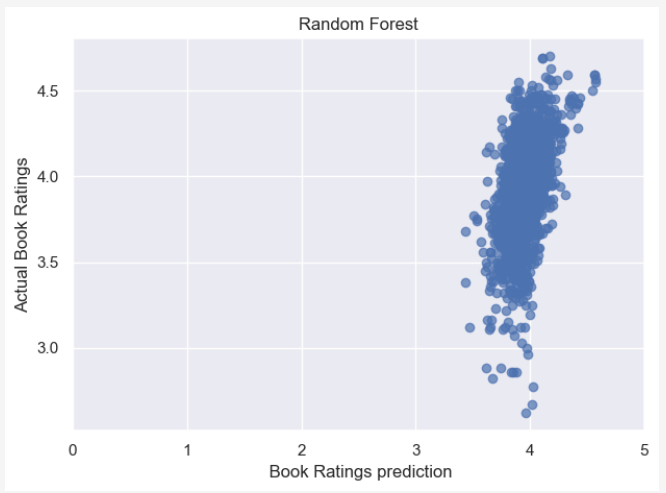
**Linear Regression**

**Une image contenant texte

Description générée automatiquement**

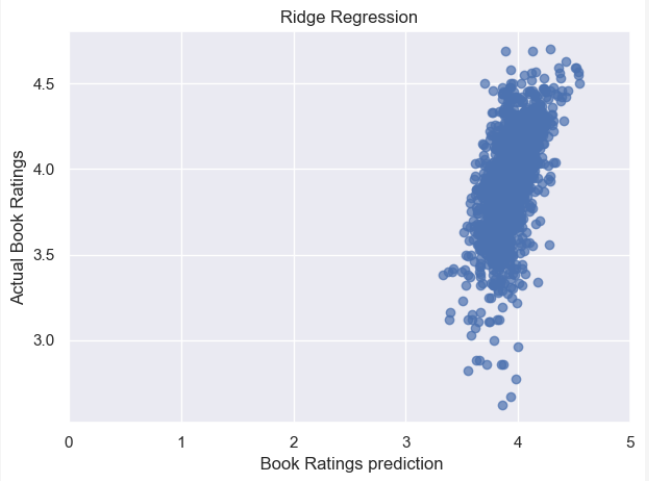
**Random Forest**

**Une image contenant texte

Description générée automatiquement**

**Ridge**

**Une image contenant texte

Description générée automatiquement**

The model which shows the best results in terms of R^2 score, MSE and RMSE is Ridge.

The R^2 score is 0.33 on the test set so only 33% of the variance of the target variable can be explained by the independent variables (predictors).

However, the MSE is very low : 5.3% on the test set, so the squared difference between the observed values and the predicted ones. The closest to 0 it is, the best. Here 5.3% is quite a good result.

The RMSE score, which represents the aggregated mean and subsequent square root of the errors between the actual and predicted, is 0.23. So the predicted has a potential gap of 0.23 versus the actual. Not perfect, but quite close to the actual.

The model is not absolutely perfect as it is able to predict ratings around 4. A bit more precisely than the Linear regression model. It is able to predict a rating in a range from 3.9 to 4.1. However, for the ratings below and above, the model is not accurate.

The reason may be because the majority of the dataset ratings seats around 4 to the model does not have enough data below 3.8 and above 4.1

In order to better visualize the actual versus predicted, the below graph represents 15 samples of the dataset showing how accurate is the model :



In order to check with the same models whereas the categories were creating noises in the models, I trained them without the categories. I processed the same outliers removal, normalization and kept the following features (for details, see notebook ‘Project\_part4\_ML\_model(nocategorie)’):

categorical column of the dataset are : ['title', 'authors', 'language\_code', 'publisher']

numerical column of the dataset are : ['average\_rating', 'num\_pages', 'ratings\_count']

Length of the dataset, test, train sets :

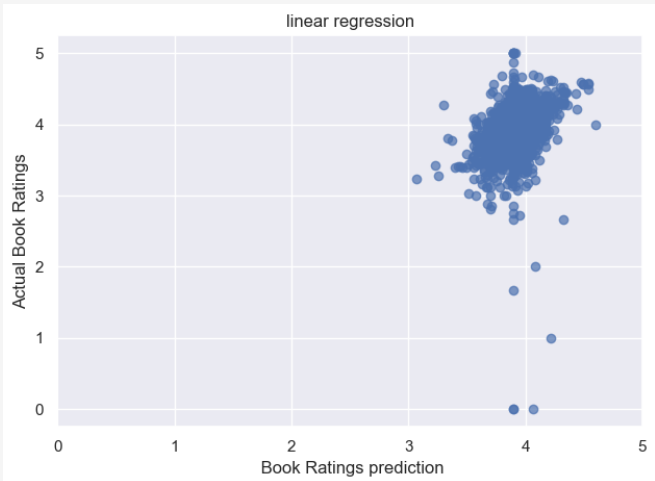
Une image contenant texte

Description générée automatiquement

1. ML without the ‘categories’

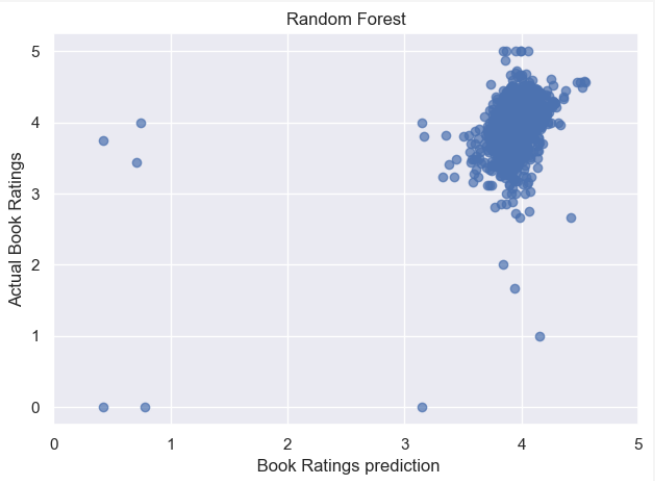
**Linear Regression**

Une image contenant texte

Description générée automatiquement

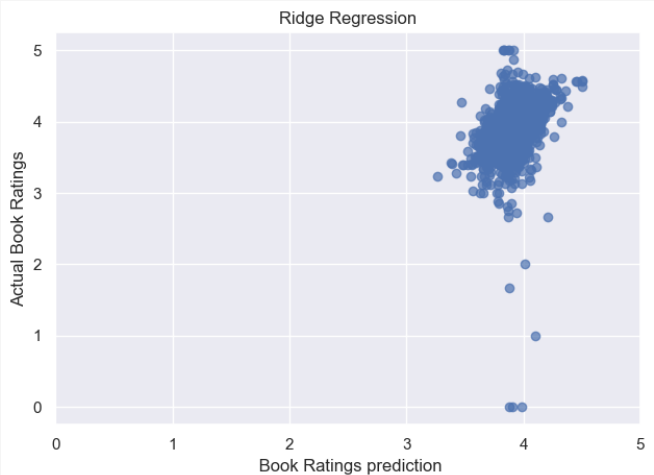
**Random Forest Regressor**

Une image contenant texte

Description générée automatiquement

**Ridge**

Une image contenant texte

Description générée automatiquement

Again, here we see that Ridge is the ‘most’ accurate among the 3 models. However, removing the categories did not improve the score.

Therefore, the independent variable ‘categories’ allows a ‘better’ rating prediction.