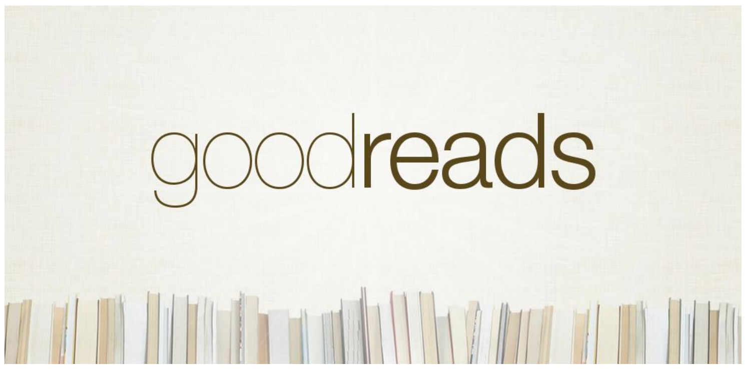
Albin Brogialdi
Florian Irrien
Luc Sagnes
Nader Thabet
Vincent Weber

Project Report Python Lab



Summary :

Link to the github repository: https://github.com/albinbrogialdi/book_rating_prediction

1. Data Analysis :

a. Data Cleaning :

The first step of a Machine learning project is to get a clean dataset, meaning the values are correct without any missing attributes. We did some check and transformation to get a clean dataset:

The first thing that we tried was to read the csv with pandas dataFrame. We directly received an error: "*ParserError: Error tokenizing data. C error: Expected 12 fields in line 3350, saw 13*". Indeed, when we look at the csv in closer detail with Excel, we notice that there are some lines with a comma (,) on the authors. Since the delimiter of the csv is also a comma, these lines will have 13 attributes instead of the 12 supposed. Since there are only 4 lines of error with the same error, we regroup the 2 values by adding a "/" to the author's name.

We deleted the space of the columns " num_pages" at the beginning of the name to prevent later being annoyed by this detail.

We verified that there was no missing value in our dataset and that was the case indeed.

Some books have an **average_rating** not equal to 0 but without any rating (i.e **ratings_count = 0**). We don't know if the value of the **ratings_count** is missing or maybe books need to have an average_rating so someone does it manually when there is no rating. In any case, we couldn't keep this data because it would have led our model to make some mistakes, so we decided to delete these lines.

One last thing to notice, on this dataset books are present. Indeed, there are some lines with for example an author named "*NOT A BOOK*". In fact, some audios are also available on the website but since it has an **average_rating** with non-null **ratings_count**, we decided to keep the data. Audio may be rated in the same way as a book.

After all the changement, our dataset was finally clean. The next step now is to start digging and take a closer look at the data itself!

b. Data Exploration :

We decided to be more flexible to work during this project separately and to make some weekly meetings to share our progress to other members of the team. By doing this, everyone could explore the dataset as he wanted to and try some analysis. We all searched in different ways and developed very different analyses sometimes. Here's some data exploration that we made. We only kept information that we found the most useful for the next part of the project.

i. Exploration shelved

Let's begin with the date. The format year-month-date isn't really the best because having values with a day granularity for this project is no use. This is why we decided to split the date in 2 columns, *publication_year* and *publication_month*. Maybe people are reading more in holidays, hence in summer on the beach. But with a little digging, the **average_rating** of a book has nothing to do with the month he was published. This is because here it is not about when people have rated a book, but when the book was published. People are not reading a book only when it's getting published, so this is why we decided to drop the publication date that has nothing to do with the average rating.

We also tried the same approach with the **language_code.** We noticed that we could regroup some languages together, especially for the English language. Indeed, it is true that depending on the country people won't speak English with the same accent or the same expression, but still, they all talk English. In the dataset they make a difference between the USA english and the British english for example. This is why we grouped into one group called '*eng*' the languages '*en-US*','*en-GB*','*en-CA*'.

We then tried to do something with the language of the book, but we didn't find any correlation towards the **average_rating,** so we decided to drop this feature.

This feature is highly correlated with the ISBN Number, on the 4th digit.

ii. Successful exploration

When we looked at the 2 attributes of **isbn10** and **isbn13**, we tried to figure out what these numbers really meant. ISBN (**International Standard Book Number**) is a codification number which integrates information. ISBN10 is dedicated to books but ran out of numbers, so it has been replaced by ISBN13 since 2007 (This is why there are 2 attributes on the dataset). Both of these codifications numbers follows this rule :

ISBN-10 :

- 1 digit: Distribution area
- 2-7 digits: Editor Number
- 1-6 digits: Publication Number given by the editor.
- 1 digit: ControlKey

ISBN-13 :

- 3 digits: Bookland, where the book is produced (978 ou 979 for books)
- 1 digit: Distribution area
- 2-7 digits: Editor Number
- 1-6 digits: Publication Number given by the editor.
- 1 digit: [ControlKeyEAN13]

Reference - See International Standard Book Number — Wikipédia.

We have in our dataset some similar information which can change by the time, for example:

- Publisher names can change but still represent the same company.
- Titles can vary due to different languages but can represent the same artwork.

For example: 978042521076 is linked to *G.P. Putnam's Sons* publisher, and on the wikipedia page of this publisher, we can see that *"In 1965, G. P. Putnam's Sons acquired Berkley Books, a mass market paperback publishing house"*.
So *G.P. Putnam's Sons* and *Berkley* have been the same publisher since 1965. We can confirm this regarding the ISBN code with the value *425* on digits 4,5 and 6 for the publishers *Berkley Books*, *G.P. Putnam's Sons, Berkley, Berkley Trade, Berkley Sensation, Underwood/Miller Berkley*.

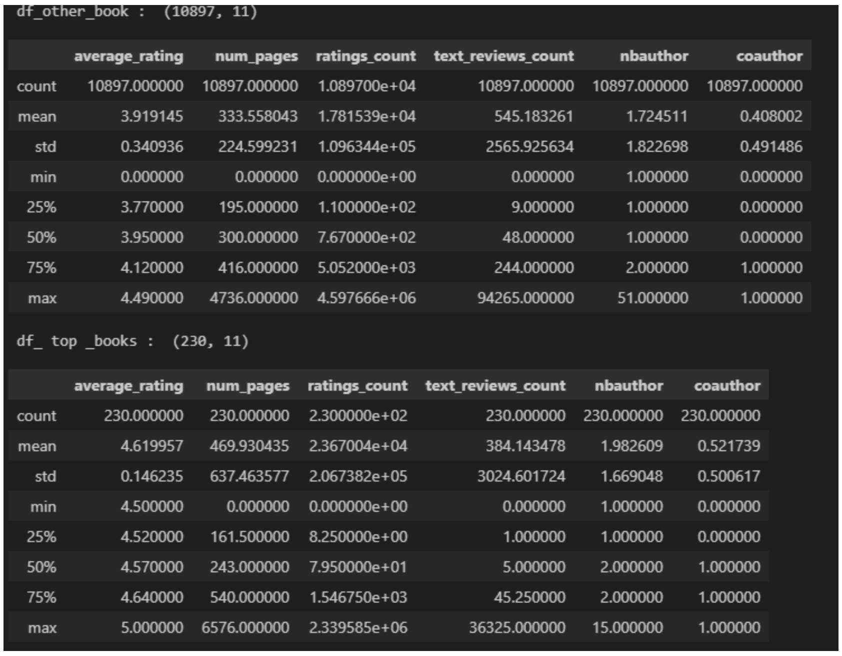
According to this information it may be interesting, instead of just using the publisher's name, to look for the owner of the publisher.
So, we created a new feature '**publisher_owner_ID_xml'** extracted from the ISBN number following several steps. USER MANUAL Draft sections Range File Generation | International ISBN Agency
We can download an xml file which describes the rules of defining ISBN publisher ID and Title IDs which is another way to get a '**publisher_owner_ID_xml'**. This powerful new column will allow us to encode and group together the same publisher.

As we can imagine, one of the most interesting features to explore is the **authors**. It is normal to think that depending on who has written the book, its quality and hence its rating should differ. But here's one problem, there can be several authors in a book and not for one other. For instance, we have some Harry Potter's book that have author J.K Rowling but some others have J.K Rowling /Mary GrandPré. For a human, we understand that J.K Rowling is the writer and Mary GrandPré just the illustrator, but the computer will consider these two as different authors, so we must be careful about it.
So, we tried some kind of Encoding with 1 column for each author in a book (J.K Rowling / Mary GrandPré would be split into 2 columns J.K Rowling and Mary GrandPré). Unfortunately, we quickly met a problem because one book has 51 different authors! It is a book of poetry, and we can't for every line just add 51 columns with most of them being set to none.
To figure this out, we decided to create 2 new columns: **nbauthors and coauthors**. **nbauthors** will be the number of authors for a book and **coauthors** a value 0 if there is only one author else 1. We will talk later on how we did encode the **authors**.

One other interesting attribute to dig is the **publisher**. We first looked at the number of books published by each publisher. We found an interesting repartition if we split the publisher with less and 3 books published and the others. If we group and split the data in 2, we can see a beautiful gaussian repartition on the average_rating. We know for some models such as Linear Regression, having a Gaussian repartition helps a lot the model because it is based and has the hypothesis that input values are following a normal law. This is why we created a new column, **factor**, that will be 0 if the **publisher** has published less than 3 books else value will be the number of books published.

Last but not the least, we found some correlation between the number of ratings and the number of text reviews (0.71), so we decided to create a new column **reviews_by_rating** which is the division of **ratings_count** by **text_reviews_count**. We can see with a correlation matrix that this combination of these 2 columns is more correlated to average_rating than the 2 splitted. (from 0.11 to -0.15)

Good book vs other books: What's the difference?
Hard to define at what point a book is "good", but if we consider a really good book as having at least a rate greater or equal than 4.5/5, then we would only consider 2.06% of all the books of the dataset. In fact, 90% of the books on the website has an **average_rating** of 4.3 /5 at maximum. So, if we look at the best book, we can see interesting stuff:
First, the mean of **num_pages** go up to 133 pages, but the median goes down from 60 pages. This means that the big books usually have a better rating.
Also, we can see that the best books have almost 600 more **ratings_count** than the normal books. So, the more we get rates, the better you will be rated. We can explain it and find this statistic also in google Maps for example with restaurants. People usually rate and comment on something when they enjoy something rather than when they find the content medium or bad. A proof that peoples are nice and prefer being positive!



```
df_other_book :  (10897, 11)
```

	average_rating	num_pages	ratings_count	text_reviews_count	nbauthor	coauthor
count	10897.000000	10897.000000	1.089700e+04	10897.000000	10897.000000	10897.000000
mean	3.919145	333.558043	1.781539e+04	545.183261	1.724511	0.408002
std	0.340936	224.599231	1.096344e+05	2565.925634	1.822698	0.491486
min	0.000000	0.000000	0.000000e+00	0.000000	1.000000	0.000000
25%	3.770000	195.000000	1.100000e+02	9.000000	1.000000	0.000000
50%	3.950000	300.000000	7.670000e+02	48.000000	1.000000	0.000000
75%	4.120000	416.000000	5.052000e+03	244.000000	2.000000	1.000000
max	4.490000	4736.000000	4.597666e+06	94265.000000	51.000000	1.000000

```
df_ top _books :  (230, 11)
```

	average_rating	num_pages	ratings_count	text_reviews_count	nbauthor	coauthor
count	230.000000	230.000000	2.300000e+02	230.000000	230.000000	230.000000
mean	4.619957	469.930435	2.367004e+04	384.143478	1.982609	0.521739
std	0.146235	637.463577	2.067382e+05	3024.601724	1.669048	0.500617
min	4.500000	0.000000	0.000000e+00	0.000000	1.000000	0.000000
25%	4.520000	161.500000	8.250000e+00	1.000000	1.000000	0.000000
50%	4.570000	243.000000	7.950000e+01	5.000000	2.000000	1.000000
75%	4.640000	540.000000	1.546750e+03	45.250000	2.000000	1.000000
max	5.000000	6576.000000	2.339585e+06	36325.000000	15.000000	1.000000

Another field to explore is the title, even if each title is unique, we can extract some information linked to special characters particularly: #. This character shows that the book belongs to a series of books (for example Harry Potter), the position of the book in the Serie and sometimes the total number of books in the series.

c. Feature engineering and selection :

Now that we got deep into the dataset, it is time to make some choices on what to keep, and how to keep it.
First, we made some modifications on the attributes **text_reviews_count, ratings_count, num_pages and factor.** To perform better with Linear Regression, we applied the log function to **text_reviews_count** and **rating_count** to get a distribution that looks more like a Gaussian distribution. We also normalized the standard deviation of **num_pages** and **factor**.

We next decided to encode the **authors** in 2 different ways:
We first decided to use the Ordinal Encoder from scikit-learn which allows us to Encode categorical features as an integer array.
If you remember Well, we sometimes have problems where we can have J.K Rowling and J.K Rowling, Mary Grandpré and with Ordinal Encoder we will not be able to join the 2 books with the same encoding

number. This is why we tried another method which takes as a parameter the number of new columns that we will create **size_buket**.
For each author, we convert it into its hash value, and we divide it % the **size_bucket**. As a result, we will get numbers between 0 and **size_bucket** which will be the new value of the author. For instance, J.K Rowling will be number 2 and Mary GrandPré number 7. So, we know that every Harry Potter's book will be now Encoded with a value 1 on the 2nd column.
Nevertheless, there are 2 cons to this method: we first create a lot more columns than the original dataset (we get **size_bucket** new column), and it is not because you have a 2 that you will be a J K Rowling book, so we group values. It is only the linear combination of all the columns that will identify an author.

Choosing columns:

To decide which columns to keep, we tried a method called PCA, Principal component analysis. Since we added a lot of columns with the encoding of the title, the publisher, and the ISBN (+33 columns with only these 3 hot encoding), PCA is a popular technique for analyzing large datasets containing a high number of dimensions/features per observation, and it is useful to reduce the dimensionality of a dataset.

We considered the 2 ways, choosing columns manually and with the PCA for the model training and evaluation.

2. Model Training and Evaluation :

a. Model used for the project

We tried several models for this project, mostly the ones that we saw during this course and the most known:
For the implementation, we first splitted our data into a train set and a test set and we verified that the distribution was well respected.

Then we implemented several regression models such as Decision Tree Regressor, Random Forest Regressor, Linear Regression, Gradient Boosting Regressor, XGBoost Regressor.

Then, all of these models have hyperparameters that can be changed from their default values. For example, a random forest has a parameter *n_estimators* with a default value of 100 and it determines the number of trees in the forest. But we can also decide to change and set the value to 50 to get less trees inside our forest, or on the other hand increase it. Optimizing these values are important because playing with them permits you to get a better model.
Instead of trying every model manually, one way to find the optimal parameters of a model and running our code only once is to use GridSearchCV from *scikit-learn* library. We only need to define a dictionary with as keys the name of the hyperparameters and the values they can take, and it will try to give us the optimal model that fits best with our dataset.

b. Comparison and evaluation of the models

Once our models are fitted, we can now predict values on our testing dataset. To measure and determine the best models we must measure the error made by each model. To do so, we used metrics like mean squared errors or R2 squared errors which allows us to determine on average what the error of our model is.
We also made some plotting to see the evolution of the error regarding the real value of **average_rating** and the predicted one. As you can see below, the error is descending and 80% of our errors are between 0 and 0.3.

Also, we saw in linear regression, your residuals should be normally distributed and normally distributed and independent. This is why we did some plotting on the error of the linear regression model and verified it was the case.

As a result, the best model we get is Gradient Boosting with a Mean Squared Error of 0.0803 and a R2 squared of 0.1306 ! Even if our R2 isn’t optimal as the best would be to reach 1, we get a good MSE!

3. Conclusion :

The aim of this project was to help us with developing an end-to-end pipeline to answer a given problem or use case. It allowed us to get a better understanding of the different steps during a Machine Learning project, and we have learned a lot by sharing the knowledge of each other inside our team. We tried to go as deep as possible into the dataset to find what was the important information that gave the actual rate of a book, for example our feature **factor** that was concluded to be a very important feature.

We would like to thank Hanna for his help during this really interesting project!