Capstone Project 2

1. Problem statement

The goal of this project is to build a book recommendation system for users, based on the **Goodreads** data set, which can be found here. I additionally included via Goodreads API two columns, relative to the book description and the book popular shelves.

Goodreads is a social cataloging website that allows individuals to freely search its database of books, annotations, and reviews. Users can sign up and register books to generate library catalogs and reading lists. They can also create their own groups of book suggestions, surveys, polls, blogs, and discussions. The company is owned by the online retailer Amazon.

The problem is very relevant for Amazon or any other related company since it can potentially increase its profits as it gives recommendations of interest to their clients. Some of the key statistics about recommender systems are the following:

- At Netflix, 2/3 of the movies watched are recommended.
- At Google, news recommendations improved click-through rate (CTR) by 38%.
- For Amazon, 35% of sales come from recommendations.

The way I intend to solve the problem is to build a collaborative-filtering system. First, use an algorithm as ALS to predict implicit behavior and, on top of that, use it as a feature to predict explicit behavior (the rating).

Deliverables: code, a written report, and a slide deck.

2. Data inspection and Cleaning

2.1) Tags

book_tags.csv contains tags/shelves/genres assigned by users to books. Tags in this file are represented by their IDs. They are sorted by goodreads_book_id ascending and count descending. ***tags.csv***, on the other hand, translates tag IDs to names. I then naturally joined these two tables, creating ***tags_df.*** There were 6 negative values in the count column which I did not understand, so I discarded them.

2.2) Ratings

ratings.csv contains almost 6 million book ratings by users. Ratings can go from one to five stars, where:

- **5 stars**: "it was amazing"
- **4 stars**: "really liked it"
- **3 stars**: "liked it"
- **2 stars**: "it was ok"
- **1 star**: "did not like it"

There was everything ok with the data set.

2.3) To read

to_read.csv provides IDs of the books marked "to read" by each user. There was everything ok with the data set.

2.4) Books

books.csv has metadata for each book (goodreads IDs, authors, title, average rating, etc.).

goodreads IDs:

Each book may have many editions. goodreads_book_id and best_book_id generally point to the most popular edition of a given book, while goodreads work_id refers to the book in the abstract sense.

It's possible to use the goodreads book and work IDs to create URLs as follows:

- https://www.goodreads.com/book/show/17397466
- https://www.goodreads.com/work/editions/24219959

Note that book_id in *ratings.csv* and *to_read.csv* maps to work_id, not to goodreads_book_id, meaning that ratings for different editions are aggregated. 'books_count' is the number of editions for a given work. I changed this name, for clarity, to book_editions.

book_id, goodreads_book_id, best_book_id, authors, average_rating and **work_id** seem ok. There were some rows where the **ISBN** and **isbn13** number are missing, but for now we will leave it like that and return later if necessary. The same happened with the columns '**original publication year**', '**language code**' and '**original title**', but since there are not that many I will leave it like that for now and return later if necessary.

When comparing the title and author columns on these cases we see that there is nothing wrong with the data. They just wrote a book with the same title than other author did, and not twice.

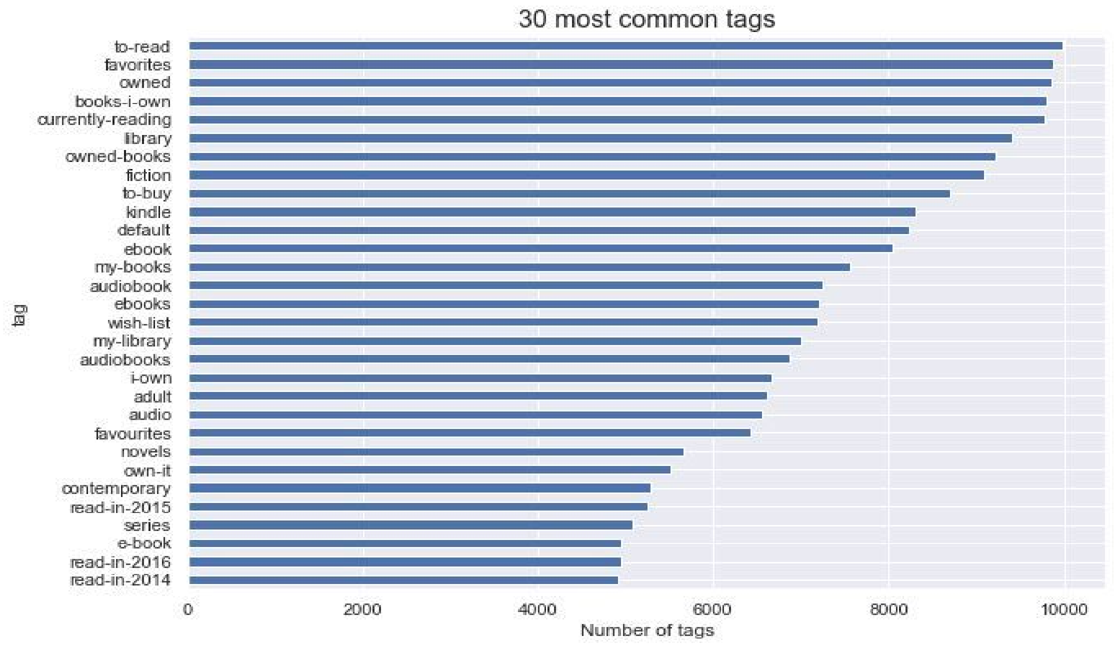
It is the column **work_ratings_count** that corresponds to the total number of ratings per row/book.

- The columns representing the **ratings** from 1 to 5 seem ok, as **work_text_reviews_count** do.
- Finally, the last two columns - **image_url** and **small_image_url** - represent the image of the cover of the book.

3. Exploratory Data Analysis

3.1. Tags

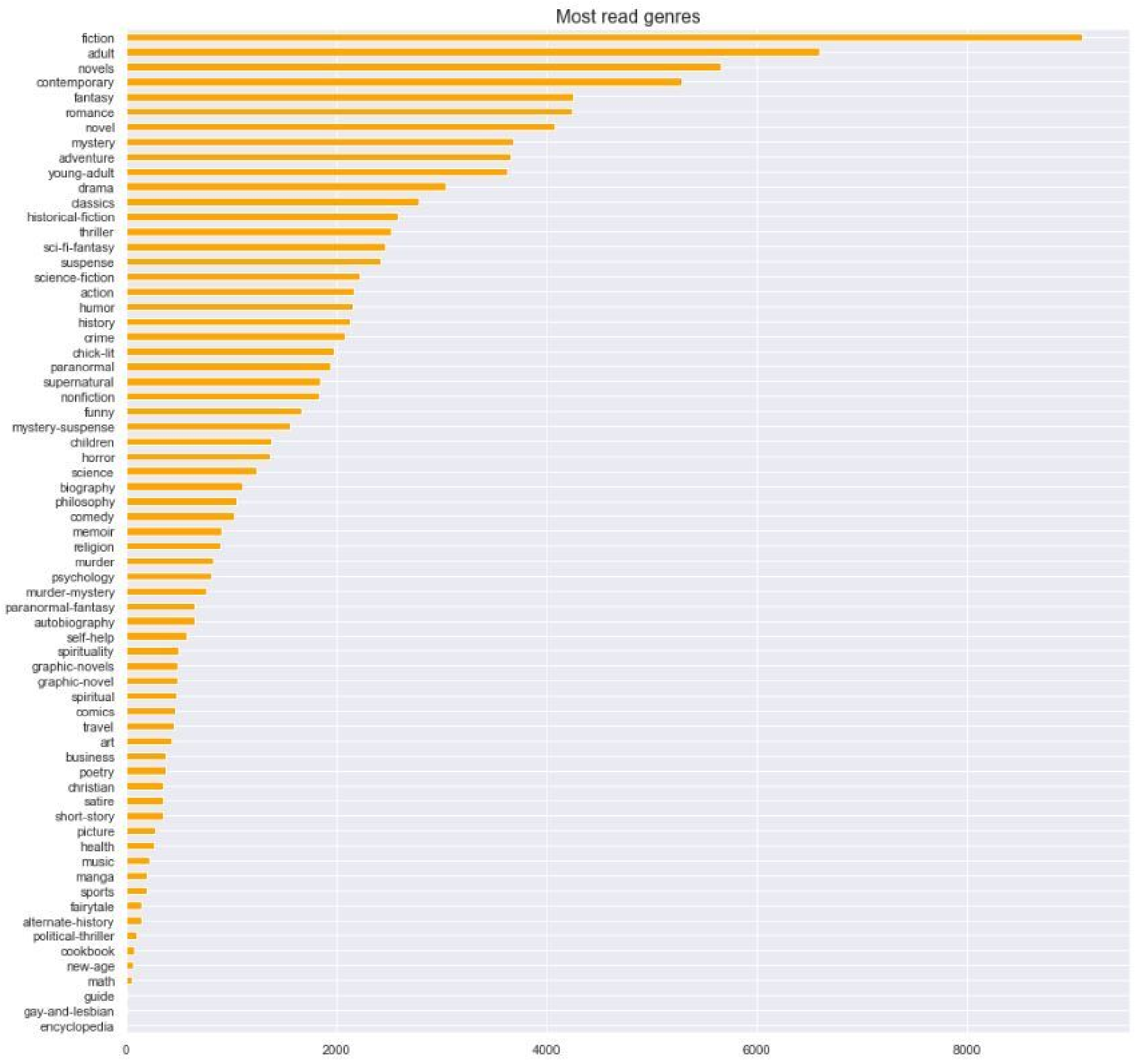
- What are the most common tags?



Although "to-read" appears at the top of the list, some tags that have the same meaning easily surpass it when aggregated. For example, "owned", "books-i-own", owned-books" and "my-books" means the same.

- What are the most common genres?

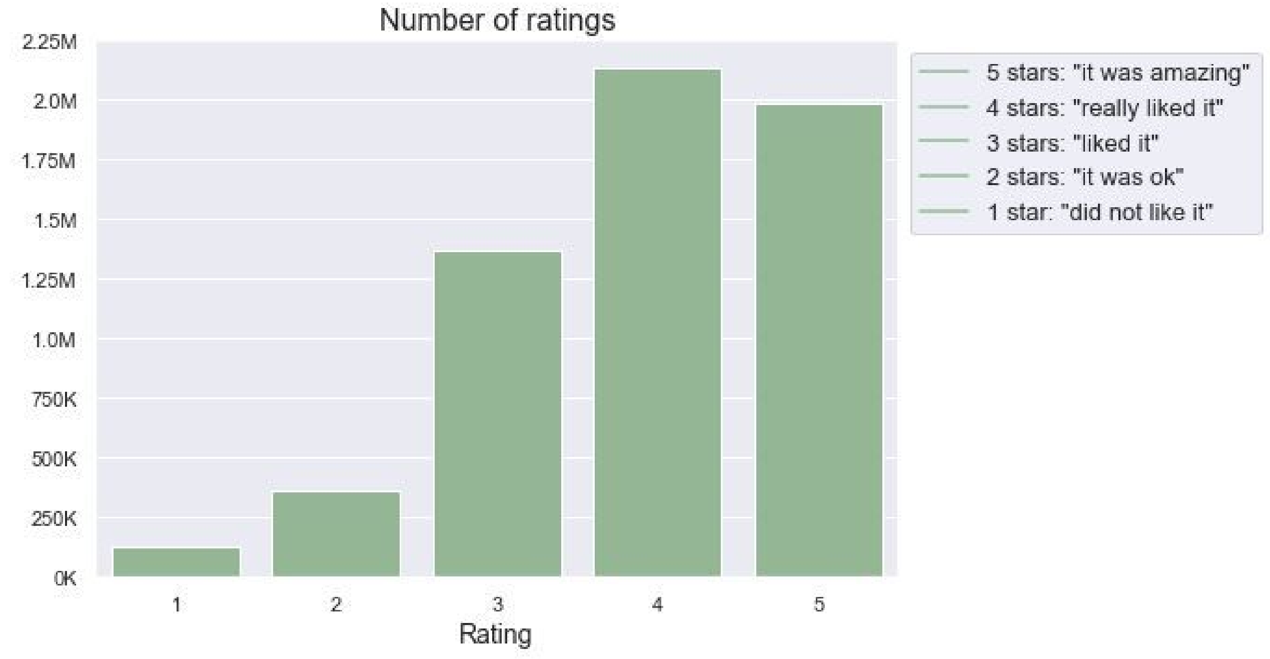
Let us take a look at the main genres. To do that I picked a list with genres from the web and added some relevant names. I will base the analysis on the previously built tag_df.



It seems that fiction books, mainly of the genre "adult", "novel"/"romance", "contemporary" and "fantasy" are amongst the most read genres by users.

3.2. Ratings

- How are ratings distributed? Do people tend to give higher or lower ratings?

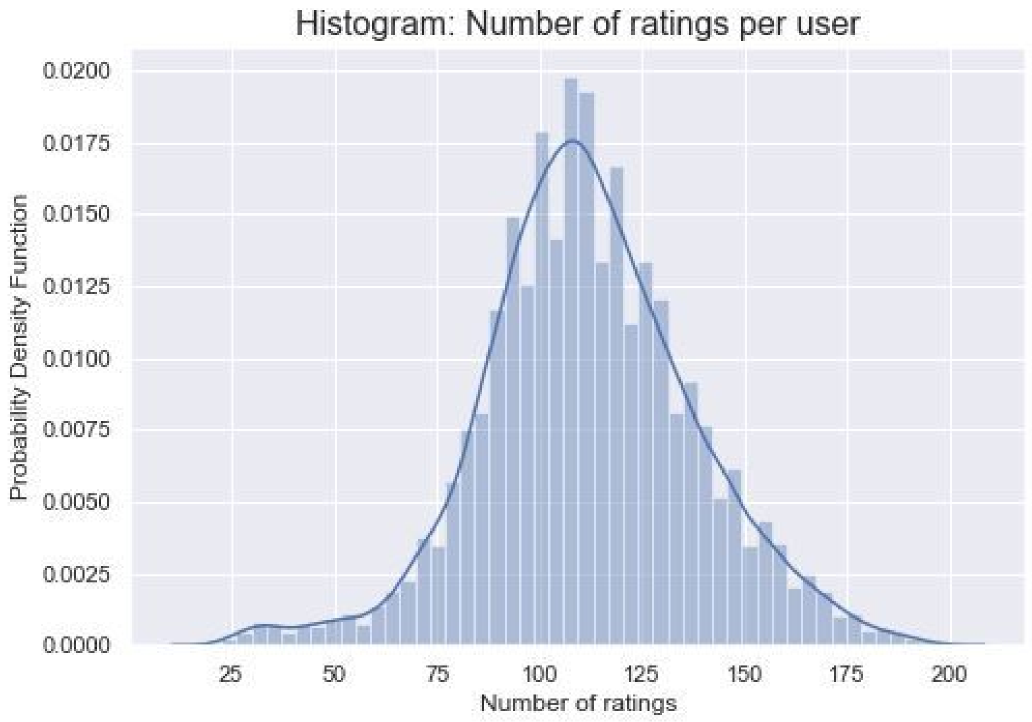


Regarding our users data set, we see that **almost 70%** of the ratings are classified as either **"it was amazing"** (5 stars) or **"really liked it"** (4 stars). We can say that in general users give good ratings. This is not surprising since this is a data set for only popular books and, among other reasons, the users are the ones who choose the books they are going to read and the topics they are going to pursue, according to their interests. If you add a **"liked it"** rating (3 stars), the percentage goes up to **91%**.

The average rating is in fact 3.9, nearly close to a "really liked it" evaluation, which is very close to the mean of all books (596873216) data frame as we've seen above (4.002191000000001). The median is 4.

- Does the distribution of the number of ratings given follow a normal distribution?

There are a total of 53424 different users in the data set. The Kernel Density Estimate which can be seen in the plot below does not suggest that, but we will check with a normality test: D'Agostino's K-squared test. The null hypothesis assumes normality of the distribution.



D'Agostino's K-squared test: statistic=460.696; p-value=0.000

Since the p-value is practically zero, we have the statistical evidence to **reject** the hypothesis that the number of ratings given follows a normal distribution. Looking at the distribution, we see that it is somewhat **left-skewed**.

- Can we have a 95% confidence interval for the mean of the ratings a user gives?

I'll assume that the data is representative of the population. The diversity in book genre is pretty widespread and users that rate the most popular books and have this many ratings given on average is good enough.

We can make for now some inferences about the population of Goodreads users using the data in 'ratings.csv' (6 million ratings) - it may be useful when making recommendations later.

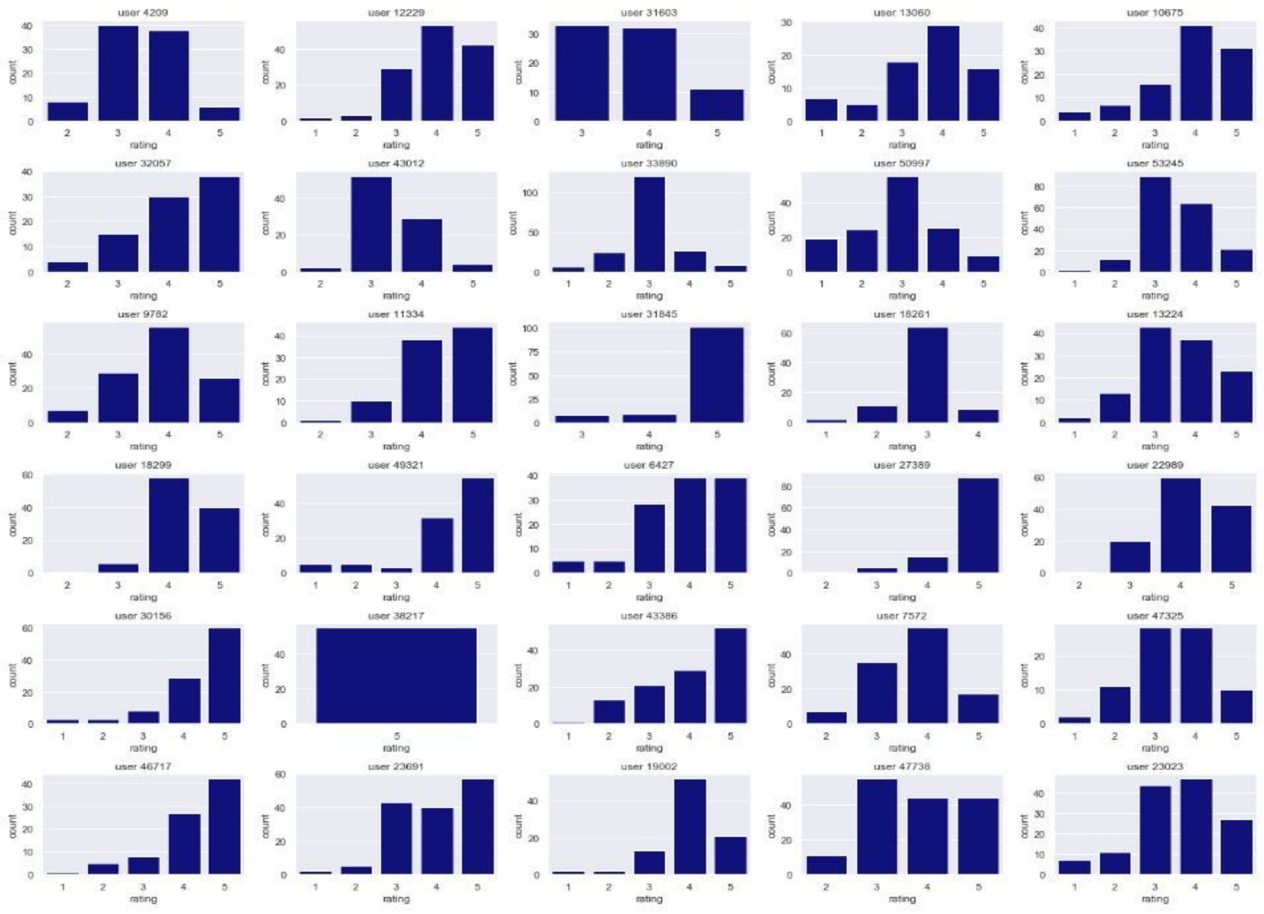
The conditions for assuming the Central Limit Theorem implications are met here, since:

- the sample is more or less symmetric and bigger than 30, the value normally assumed for the validity of the Central Limit Theorem.
- there was no replacement when retrieving the sample, but the observations are considered to be independent, since they constitute less than 10% of the population size (the 10% rule).
- the sample was taken randomly (I assumed that it was).

I created a bootstrap confidence interval for the mean between [111.64700883, 112.09271395]. This is a pretty close range. If we repeated measurements over and over again, 95% of the observed values for the mean of the number of ratings given by users would be very close to 112.

- Do distributions of ratings among users vary significantly?

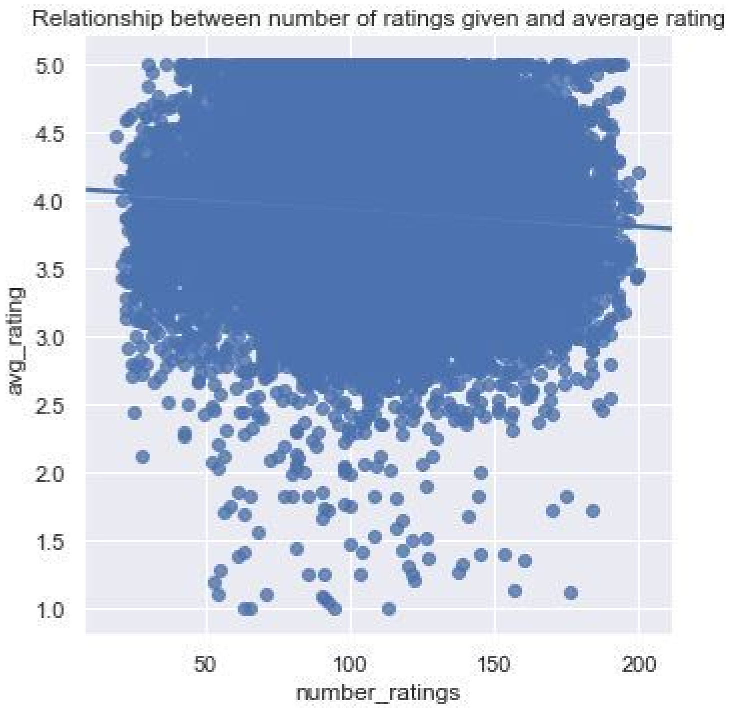
For a sample of 30 users:



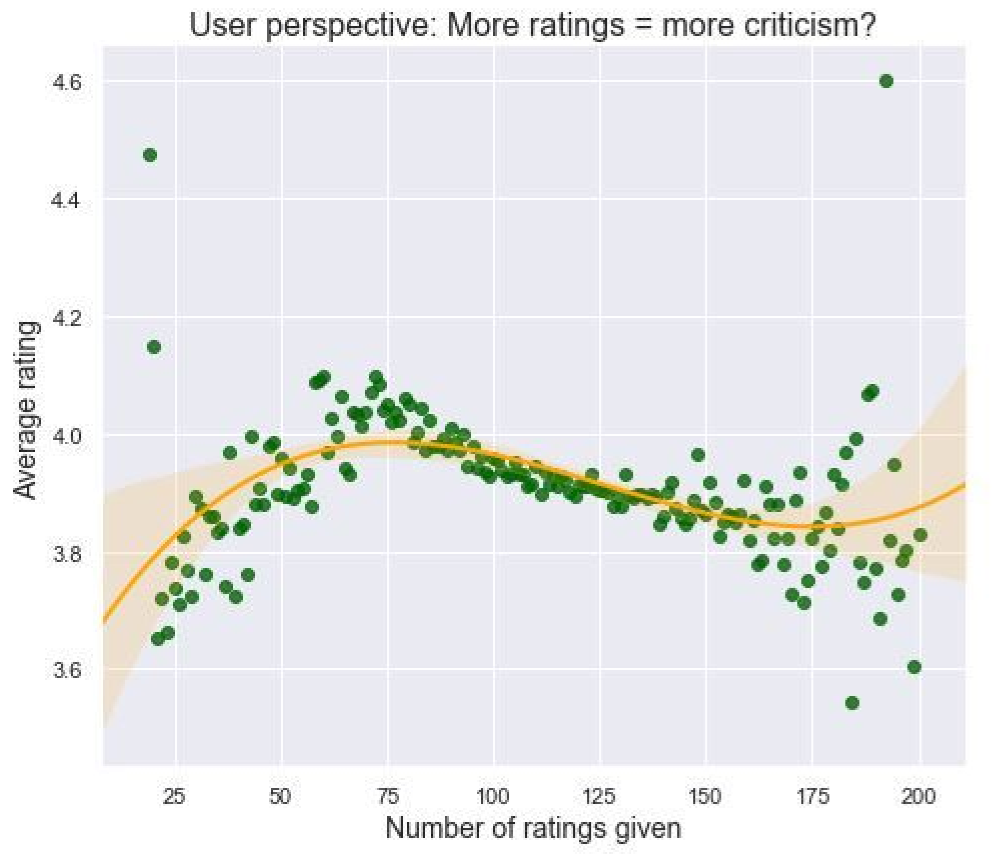
It is straightforward to see that the distributions of ratings vary significantly among users. We can see right away that the average rating by users are also significantly different.

- How does the AVERAGE rating given by a user evolves as the number of his/her number of given ratings increases, on average? Do people tend to become more critic and give lower ratings?

I first created a new data frame with user_id, number of ratings given and average rating. When considering all the points, the scatter plot looks like this:



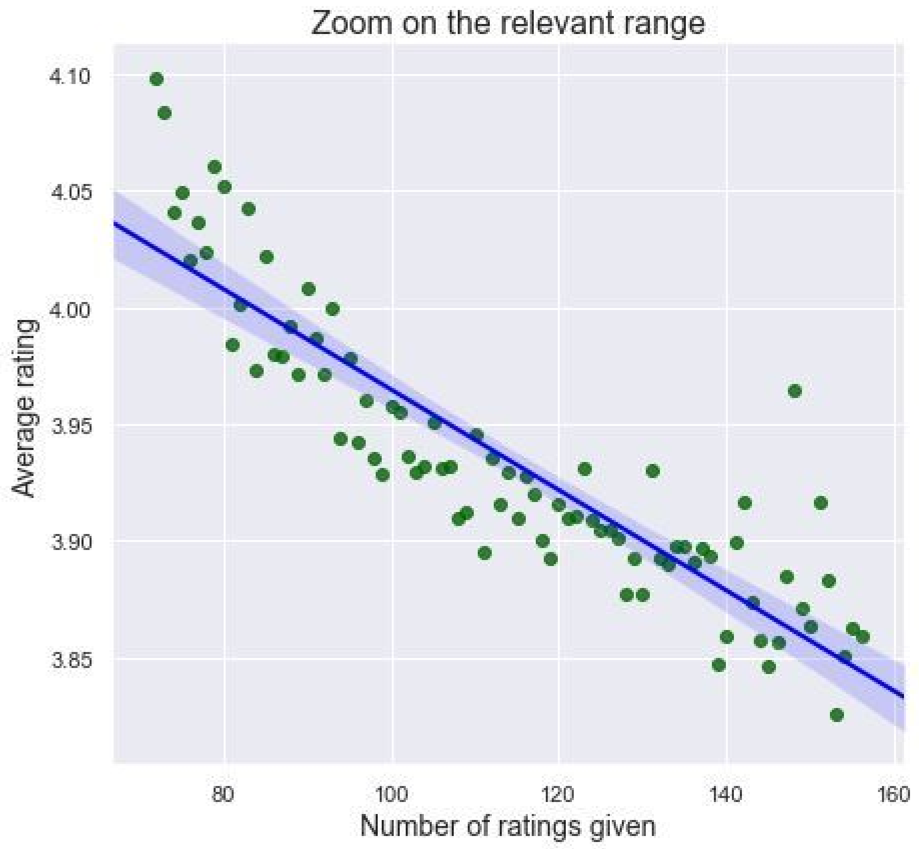
In order to properly see the relationship I am going to agglomerate the users with the same number of given ratings and compute the average of their average rating given. Otherwise I would have almost 6 million points on the scatter plot as seen above. In any case, it's a negative relationship.



Each point encompasses a different number of users. As we have seen in the previous histogram, most of the data reside between 75 and 150 ratings. In fact, almost 90%, as computed below. Curiously, we see here that up until 75 ratings given by a "user" the average rating tends to increase and probably the enthusiasm for books and reading as well. After that, and until 150, which is exactly the range where most of the data is, the average rating linearly decreases as the number of ratings increase. It then spreads across the range between 3.5 and 4.6 without a clear tendency.

This may suggest that, on average, the user tends to become more critical and give lower ratings as it reads more books.

Taking a closer look at the relevant range, we see more clearly this negative relationship:

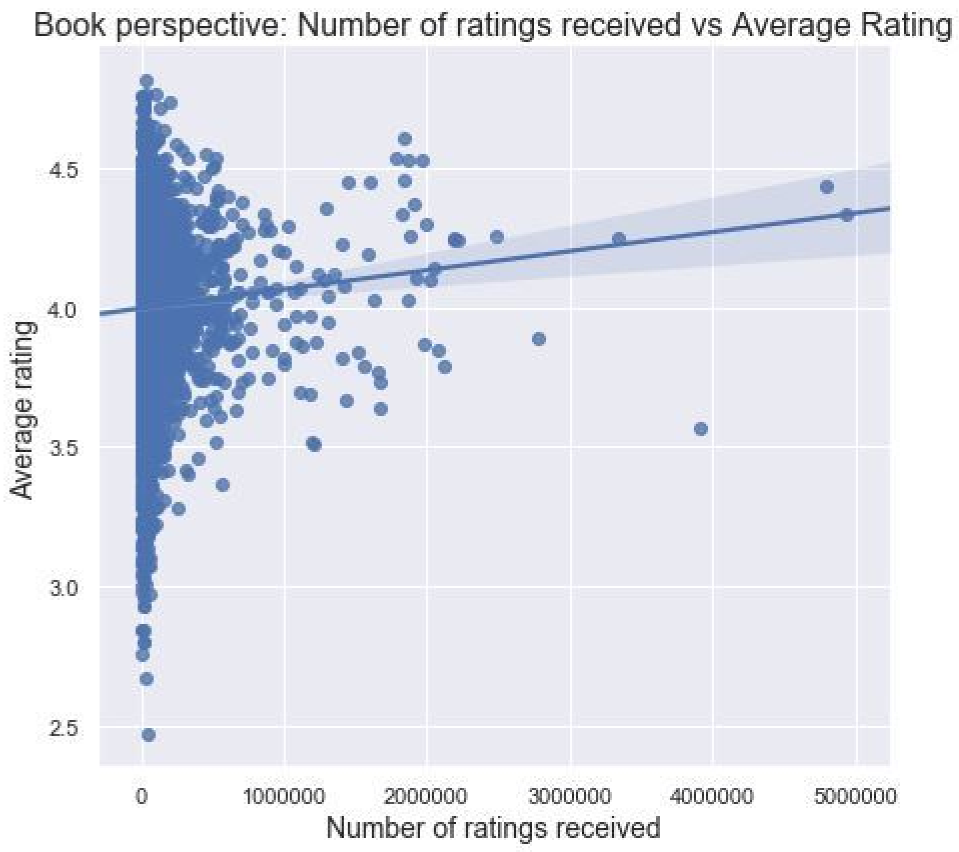


There is a very strong negative correlation (-0.89) between the number of ratings a user has given and the average rating, suggesting, again, that the user tends to become less enthusiastic, on average, as he/she reads more books.

- How does the average rating of a book relates to the number of ratings it has received?

Now from the book perspective, we now see a slight positive relationship between the number of ratings (or fame) a book has received and its average rating. We see indeed a small positive correlation coefficient equal to 4.5%. Can we be sure this correlation is not due by chance? To know that I did a test of correlation, where

- **Null hypothesis**: "the two variables are completely uncorrelated"
- **Alternate hypothesis**: "the two variables are correlated"
- **Test statistic**: "pearson correlation coefficient"



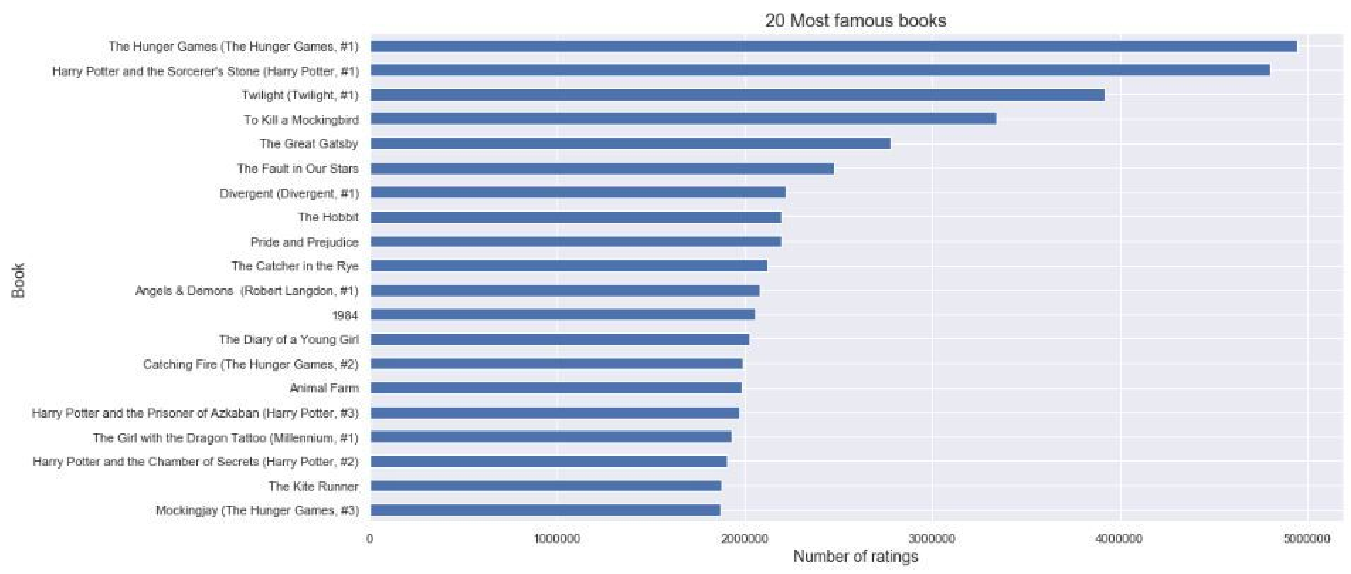
With a p-value equal to 0.0053, we **r**eject the hypothesis that the two variables are not correlated, considering the level of significance of 1%. As the number of ratings a book gets increases, the higher the chance that it will have a higher average rating. In any case, there is too much noise and the correlation is not clear.

- How does the distribution of ratings given by a user evolve as the number of ratings increases, on average?

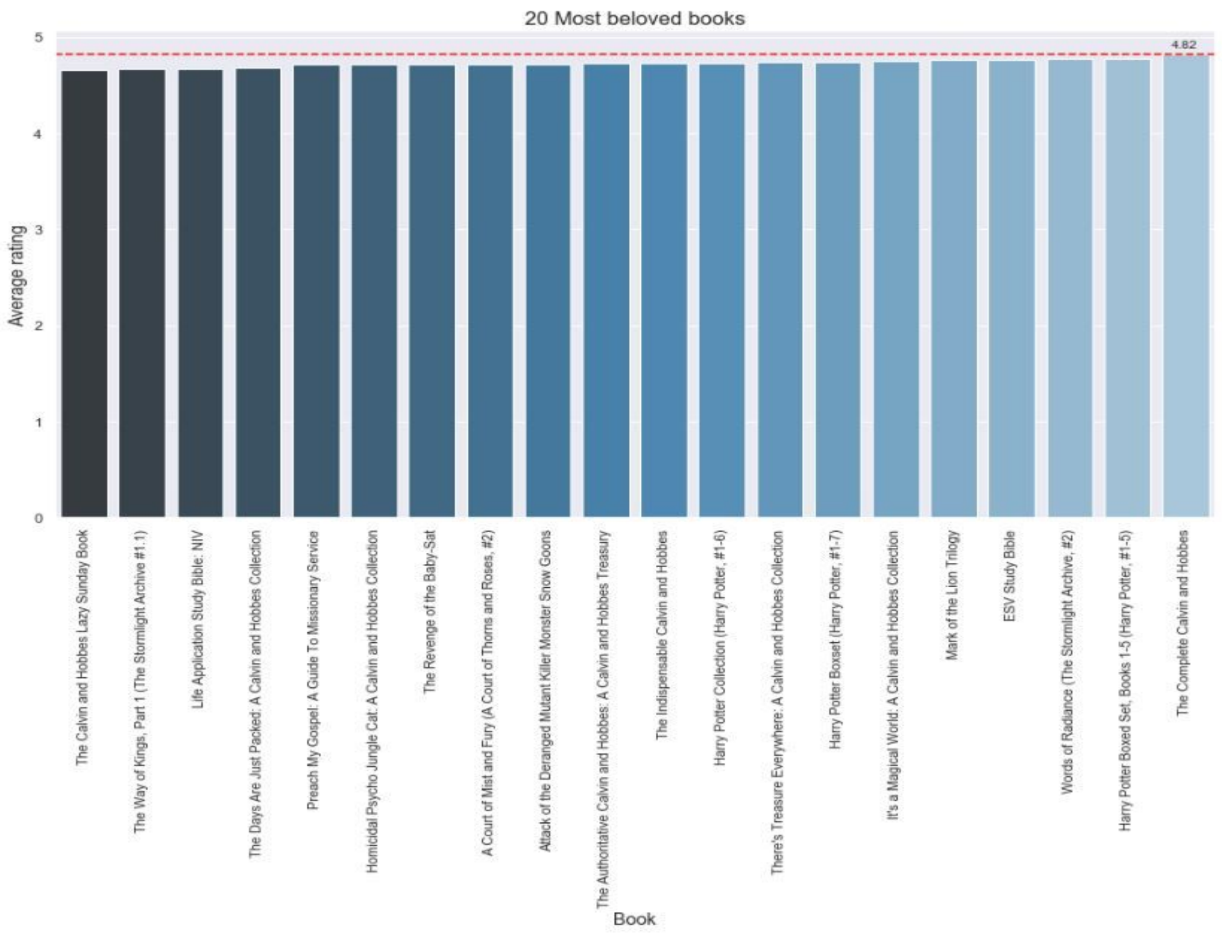
To do this I built a plot using bokeh (please see the jupyter notebook) where we can clearly see that there is a tendency on the distribution to become rightly-skewed as the number of given ratings increase.

3.3. Books

- What are the most famous books?



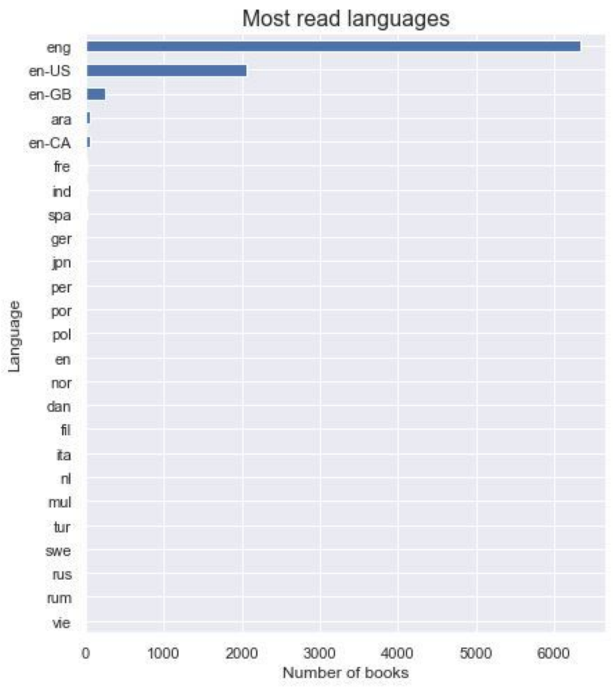
- What are the most beloved books?

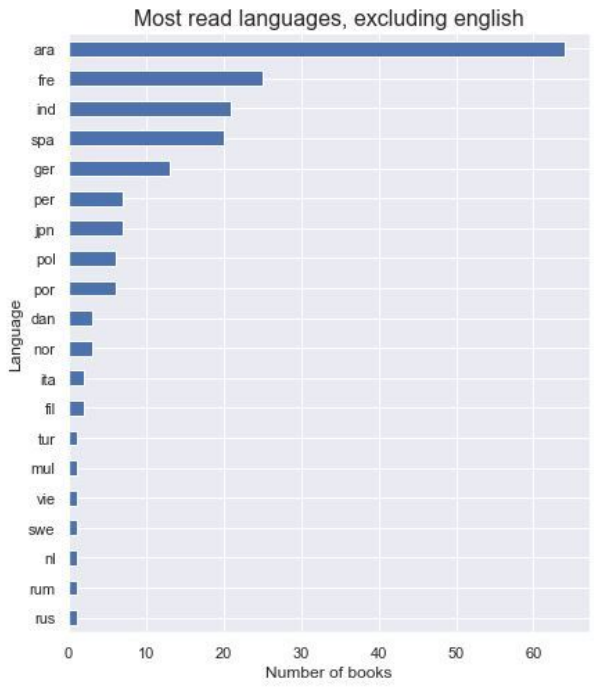


By famous I meant the books with the highest number of given ratings. By beloved, the books with the highest average ratings

- What are the most read languages?

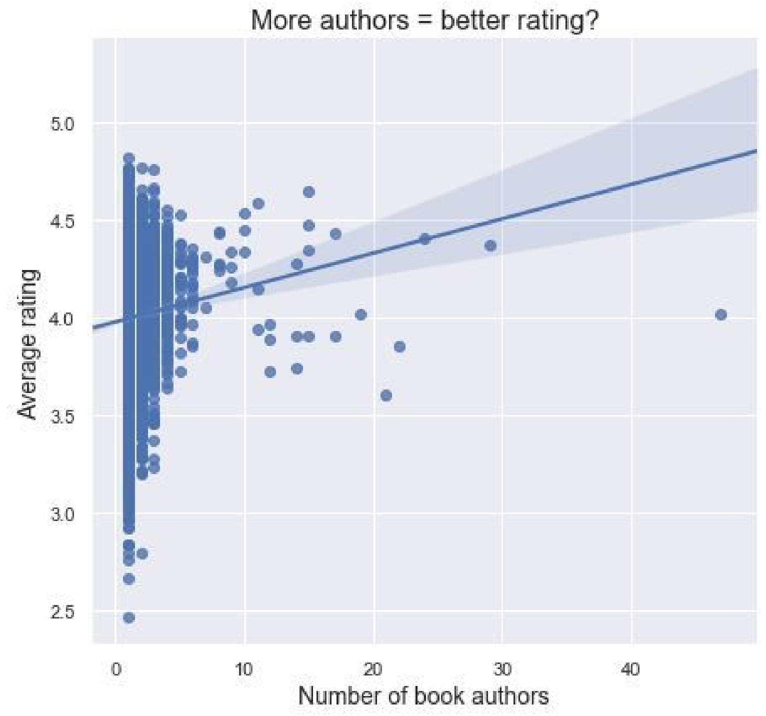
English is by far the most read language, and almost the only one in this data set, in relative terms.



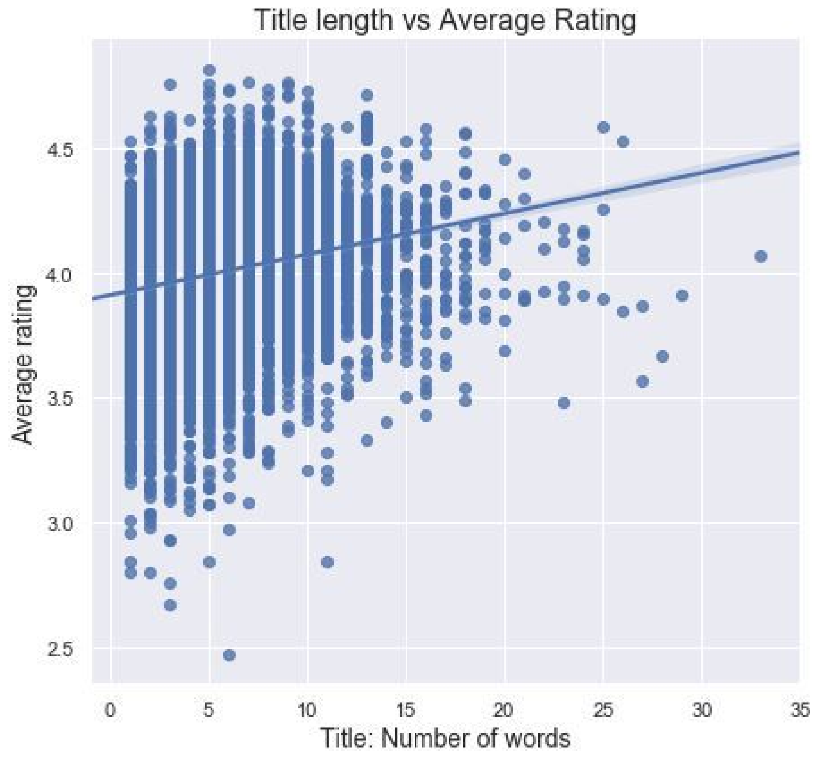


- Is having more authors better?

Here the correlation is slightly more positive (7.5%): as the number of authors increases, so does the average rating. There is great variability, so this result should not be taken with too much weight.



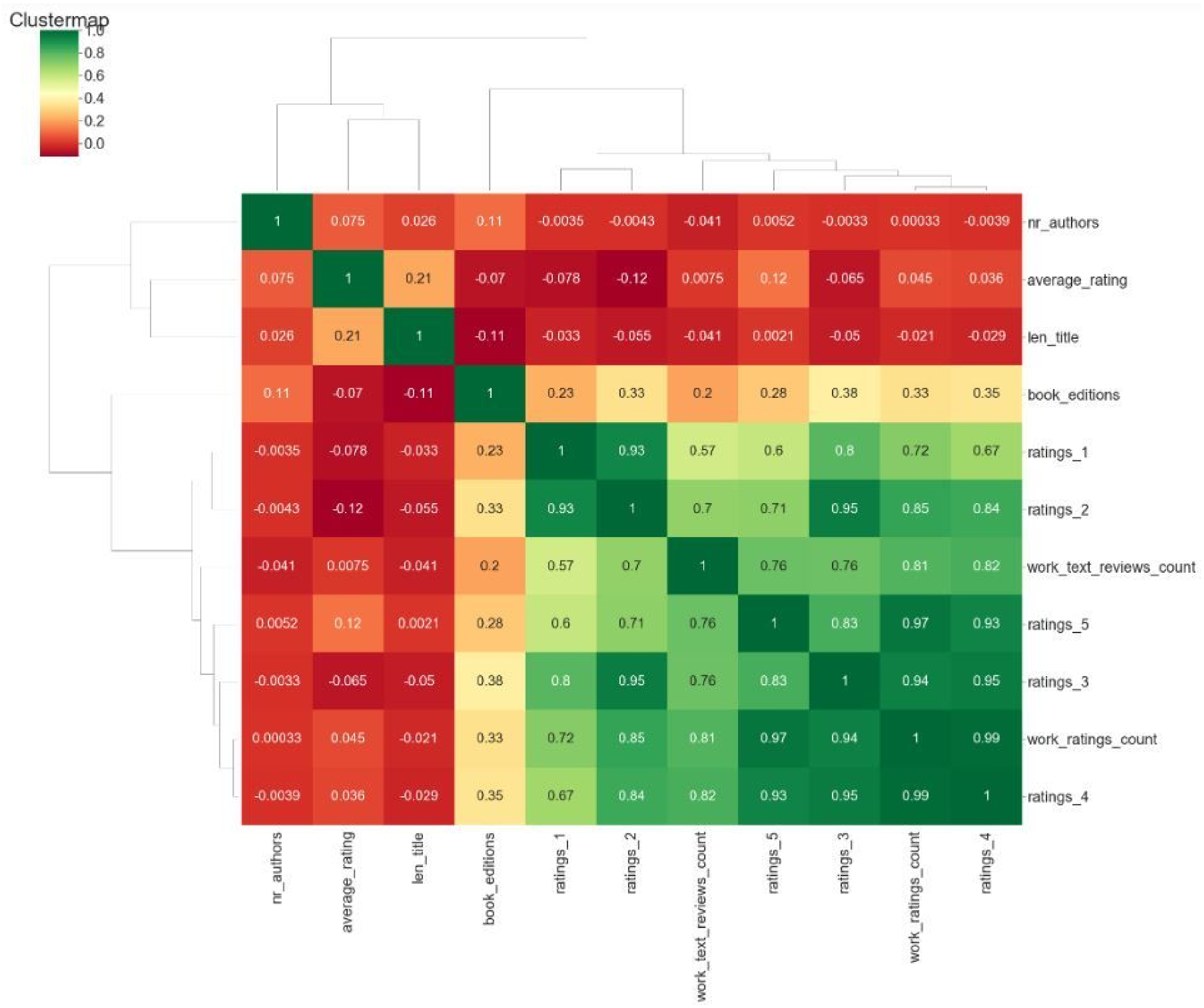
- **Does the length of the title influence the average rating?**



There is also a small positive correlation (21%) between the length of the title and the average rating.

- Correlations

Clustermap with some relevant variables:



	nr_authors	average_rating	len_title	book_editions	ratings_1	ratings_2	work_text_reviews_count	ratings_5	ratings_3	work_ratings_count	ratings_4
nr_authors	1	0.075	0.026	0.11	-0.0035	-0.0043	-0.041	0.0052	-0.0033	0.00033	-0.0039
average_rating	0.075	1	0.21	-0.07	-0.078	-0.12	0.0075	0.12	-0.065	0.045	0.036
len_title	0.026	0.21	1	-0.11	-0.033	-0.055	-0.041	0.0021	-0.05	-0.021	-0.029
book_editions	0.11	-0.07	-0.11	1	0.23	0.33	0.2	0.28	0.38	0.33	0.35
ratings_1	-0.0035	-0.078	-0.033	0.23	1	0.93	0.57	0.6	0.8	0.72	0.67
ratings_2	-0.0043	-0.12	-0.055	0.33	0.93	1	0.7	0.71	0.95	0.85	0.84
work_text_reviews_count	-0.041	0.0075	-0.041	0.2	0.57	0.7	1	0.76	0.76	0.81	0.82
ratings_5	0.0052	0.12	0.0021	0.28	0.6	0.71	0.76	1	0.83	0.97	0.93
ratings_3	-0.0033	-0.065	-0.05	0.38	0.8	0.95	0.76	0.83	1	0.94	0.95
work_ratings_count	0.00033	0.045	-0.021	0.33	0.72	0.85	0.81	0.97	0.94	1	0.99
ratings_4	-0.0039	0.036	-0.029	0.35	0.67	0.84	0.82	0.93	0.95	0.99	1

Variables as the length of the title, the number of 2 and 5 star ratings have some correlation with the average rating a book has in this dataset.

4) Recommendation System

One of the things that Recommendation System experts suggest that typically work in these kinds of problems is, besides doing appropriate data preprocessing and dimensionality reduction using matrix factorization, is combining methods through ensembles. In order to construct the recommendation system for Goodreads users, I will start by building an implicit matrix factorization algorithm using the Alternating Least Squares (ALS) algorithm. Many have acknowledged that implicit feedback is more useful, as it is usually a better representation of user behavior and, for example, also better correlated with AB test results. It also has some drawbacks, as it is not always the case that implicit feedback correlates well with user retention (think of clickbait). With that in mind, I will combine implicit and explicit models to not only tackle these questions but also build a more efficient final model.

On top of the ALS model, that is, using it as a feature, I train an efficient Gradient Boosting Decision Tree ensemble using scikit-learn's Light Gradient Boosting (LightGBM), after testing on several models with different specifications.

4.1) Build a matrix factorization algorithm using ALS (Alternating Least Squares) for implicit feedback

- Method:

Taking the ratings dataframe, containing user id’s, book id’s and almost 6 million ratings, I created a sparse matrix with **53,424 rows** (number of unique users) and **10,000 columns** (number of unique books), which, when considering the almost 6 million ratings, gives us a **sparsity value of 98.9%**, that is, filled with 0’s. This value makes sense since, although we are dealing with the most popular books, there are 10,000 of them.

I then defined a function, ‘make_train’, which takes that original user-book matrix and "mask" a percentage of the original ratings where a user-book interaction has taken place. **The test set contains all of the original ratings (but as a binary choice), while the training set replaces the specified percentage of them with zero in the original ratings matrix**.

Then, I created an **ALS model** using the library ***implicit***. For this, I considered a number of **latent factors** equal to 30 and an **alpha** value (which is then multiplied with the training set) of 15, after testing several hypotheses for both variables - namely, [10, 15, 20, 30] for number of latent factors and [1, 15] for alpha.

- ALS Results:

Considering this model with these parameters, we were able to calculate the **mean AUC** of all users which were randomly chosen for the training set and get a value equal to **95,8%** when compared to the test set. I also calculated the mean AUC of these users when compared with a popularity benchmark (where a sum of book interactions was considered to find the most popular or read, since the test set was stored as a binary preference matrix), where the result was not so strong but still good, equal to **80,1%**.

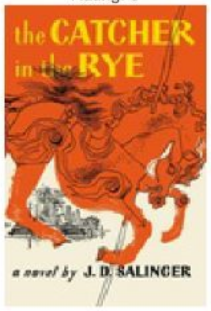
Recommendation example

It was then natural to take a user as an example and see if the recommendations of the implicit model made sense. For instance, with user nº 15:

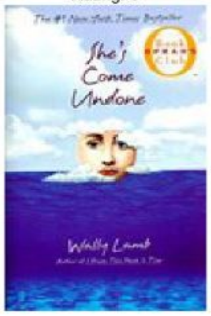
User id: 15

Top 8 favorite books:

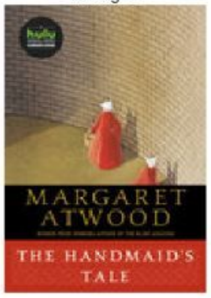
The Catcher in the Rye
Rating: 5



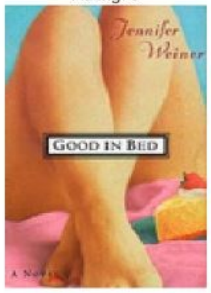
She's Come Undone
Rating: 5



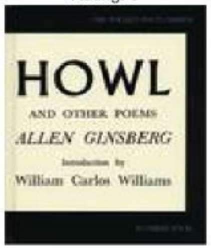
The Handmaid's Tale
Rating: 5



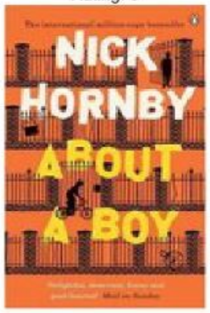
Good in Bed (Cannie Shapiro, #1)
Rating: 5



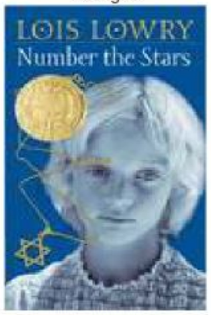
Howl and Other Poems
Rating: 5



About a Boy
Rating: 5



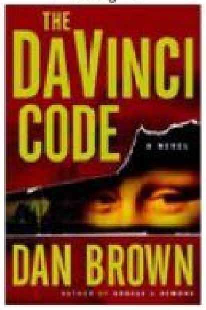
Number the Stars
Rating: 5



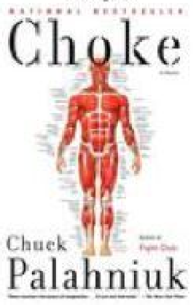
The Perks of Being a Wallflower
Rating: 5

8 least favorite books:

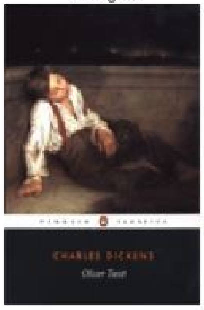
The Da Vinci Code (Robert Langdon, #2)
Rating: 3



Choke
Rating: 3

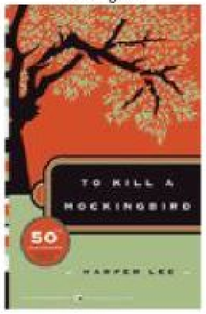


Oliver Twist
Rating: 3

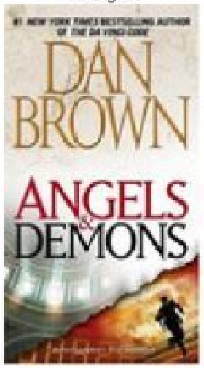


The Realm of Possibility
Rating: 3

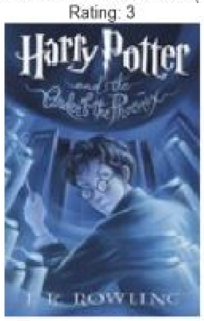
To Kill a Mockingbird
Rating: 3



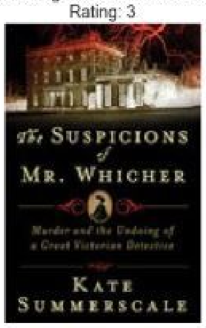
Angels & Demons (Robert Langdon, #1)
Rating: 3



Harry Potter and the Order of the Phoenix (Harry Potter, #5)
Rating: 3



The Suspicions of Mr. Whicher: A Shocking Murder and the Undoing o
Rating: 3


The SUSPICIONS of MR. WHICHER
Murder and the Undoing of a Great Victorian Detective
KATE SUMMERSCALE

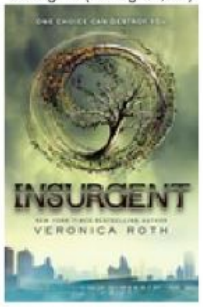
Recommended books for this user:

Blink: The Power of Thinking Without Thinking

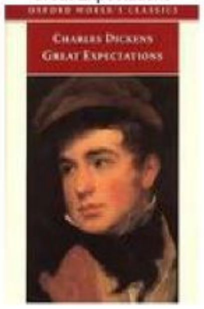
The Nightingale

Harry Potter and the Goblet of Fire (Harry Potter, #4)

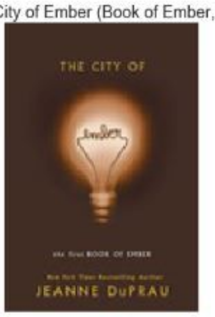
Insurgent (Divergent, #2)



Great Expectations

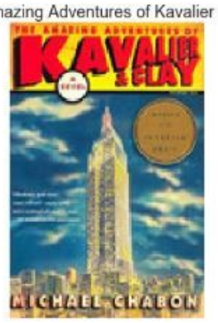


The City of Ember (Book of Ember, #1)

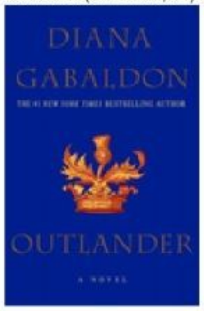


Torment (Fallen, #2)

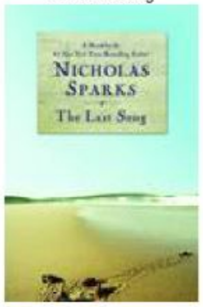
The Amazing Adventures of Kavalier & Clay



Outlander (Outlander, #1)



The Last Song



Based on the content of the recommended books and the books the user read, it seems they are somewhat accurate based on an **implicit** model. Although a book of the series “Harry Potter”, classic popular books like “To Kill a Mockingbird” and “Oliver Twist” and a popular book like “The da Vinci Code” or “Angels & Demons” are among the least favorite (which, however, the user stills attributes 3 (like it) stars), the recommendations - based on an implicit model - seem accurate, in the sense that books like “Harry Potter #4” and “Great Expectations”, and the popular “Insurgent #2) are suggested. On a more subjective tone, books like “Outlander #1” and “The Last Song” could also be a good fit.

4.2) Preparing the dataset

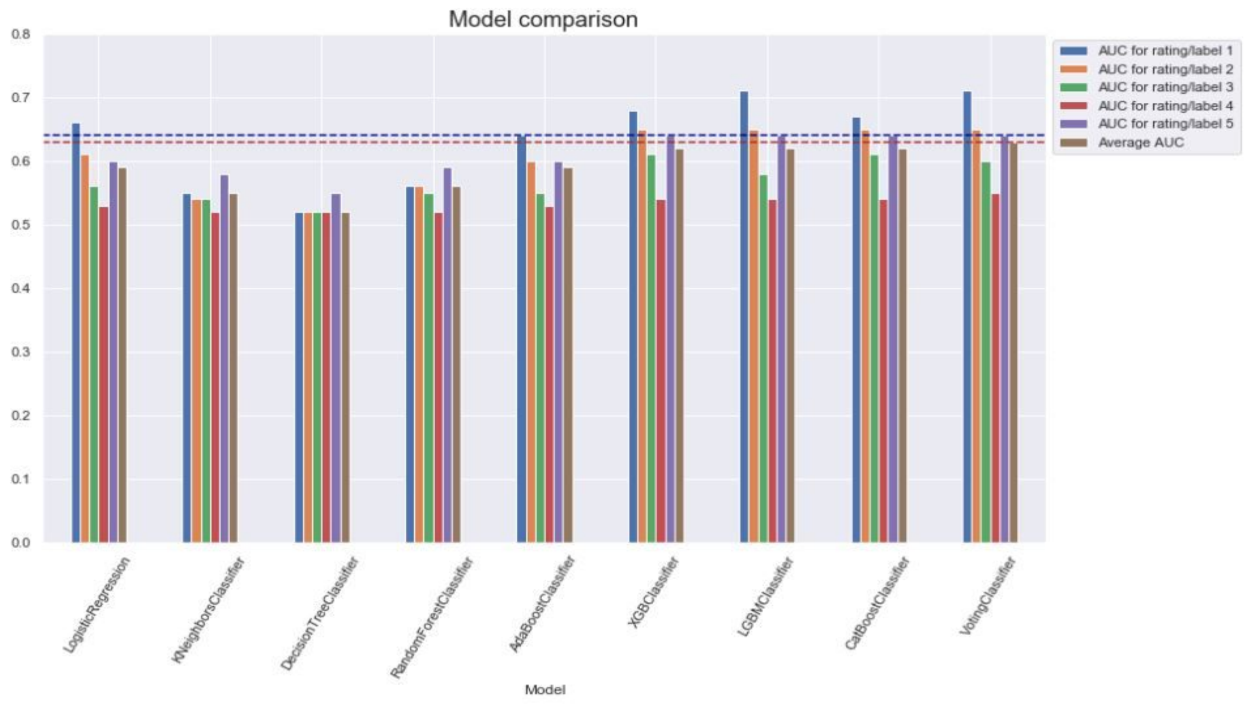
I then create a new dataframe - a copy of the ratings dataframe, but with a new column, ALS, with the normalized prediction values of the previously constructed implicit model. Based mainly on the previous exploratory phase, as well, I add the following variables:

- *Number of ratings given by the user; Average rating of the book; Number of 5-star ratings; Number of 4-star ratings; Number of 3-star ratings; Number of 2-star ratings; Number of 1-star ratings; Number of book editions*

I then split into training and test set, again with a 80-20% rule. Since we are dealing with unbalanced data set, as seen before, we will need to compensate for this during the modeling phase.

4.3) Explicit Rating Classifiers

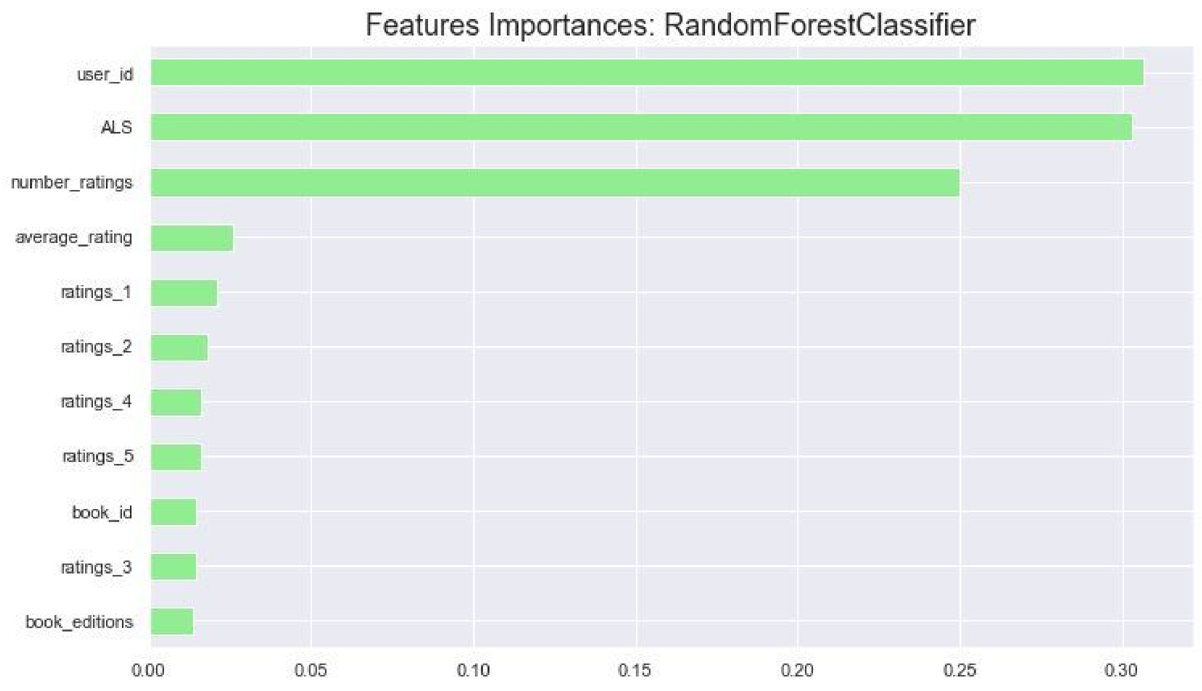
I then proceeded to test models with the purpose of predicting user's explicit behavior, that is, his/her rating. For that, I tested the following models and got the following results:



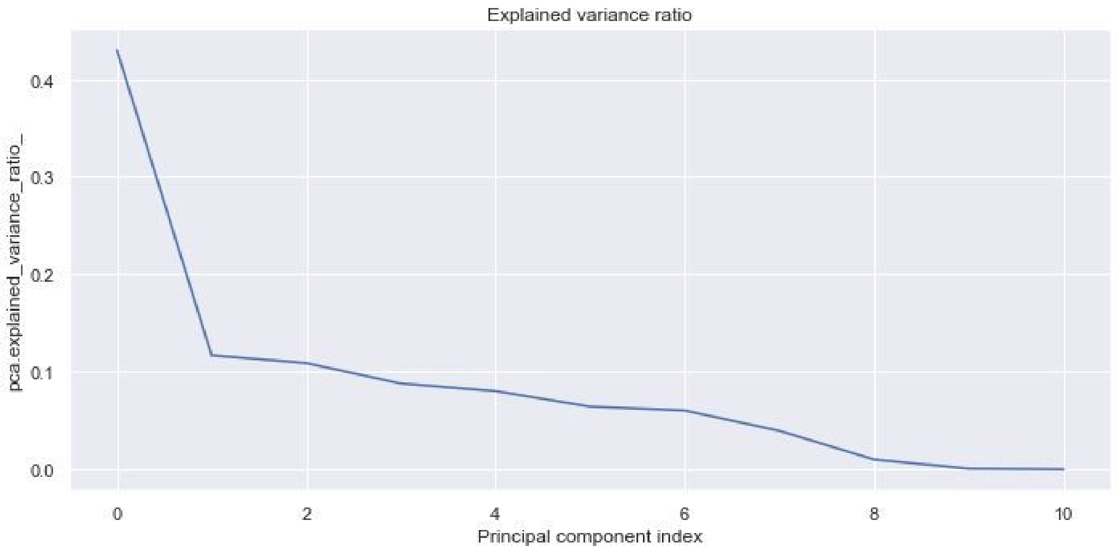
Due to the performance and results obtained I chose to go with the LGBMClassifier. It gave almost the same average AUC score of the best model (the Voting Classifier, which was based on 3 models: Logistic Regression, XGBTClassifier and LGBMClassifier) but was substantially faster to train.

4.4) Dimensionality reduction & Feature selection

As it was hinted when seeing the feature importances of the Random Forest Classifier, there were mainly 3 crucial features for the model, namely the user id, the ALS computed column, the number of ratings of the user and, in a smaller scale, the average rating of the book.

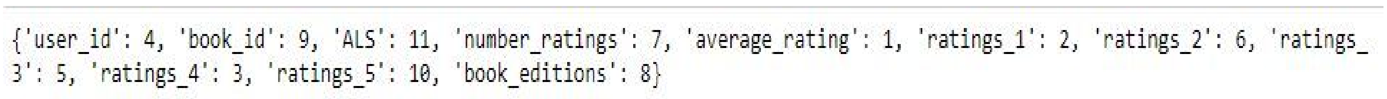


In order to see what would be the optimal number of dimensions, I started to check, using Principal Component Analysis (PCA), how the explained variance ratio evolved as the principal component index increases, and see where does the variance ratio significantly decreases, which is known as the “Elbow method”.



We see that we can reduce the dataset to 3 or 4 dimensions without compromising too much on explained variance.

Next, performing feature selection with Recursive Feature Elimination (RFE), I got the following ranking among the chosen features:



```
{'user_id': 4, 'book_id': 9, 'ALS': 11, 'number_ratings': 7, 'average_rating': 1, 'ratings_1': 2, 'ratings_2': 6, 'ratings_
3': 5, 'ratings_4': 3, 'ratings_5': 10, 'book_editions': 8}
```

I then proceeded to compute the average AUC score while iteratively consider the training variables according to a ranking range between 1 and 10. Since the average AUC score does not improve substantially beyond a ranking of 4, and considering also the important features of the Random Forest Classifier previously seen, **I chose to keep only the following features**:

- user_id
- ALS
- number_ratings
- average_rating
- ratings_1
- ratings_4
- book_id

4.5) Hyperparameter Tuning

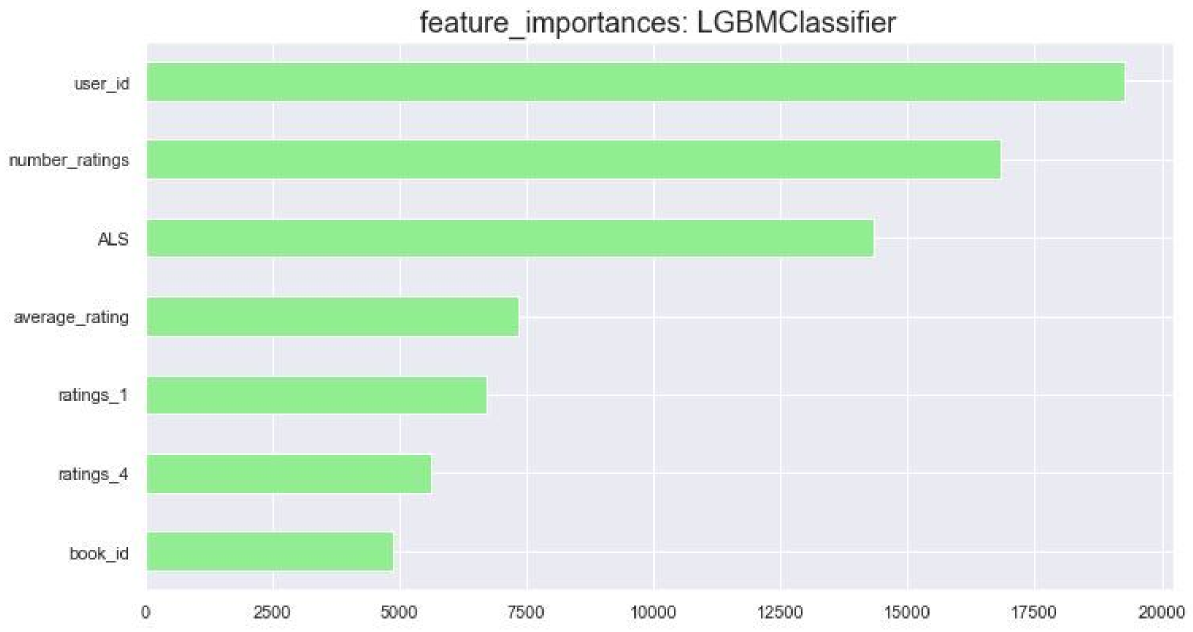
I then tuned some hyperparameters of the LGBMClassifier. The parameter grid was set as:

```
{'boosting_type': ['gbdt', 'dart'],
 'num_leaves': [21, 31, 41, 51, 61],
 'max_depth': [-1, 9],
 'learning_rate': np.arange(0.1, 1, 0.1)}
```

Scoring the model with ‘balanced_accuracy’, the best model got the following parameters, different than the default ones which were tested as well, as seen above:

{n_estimators=300, learning_rate=0.8, num_leaves=51}

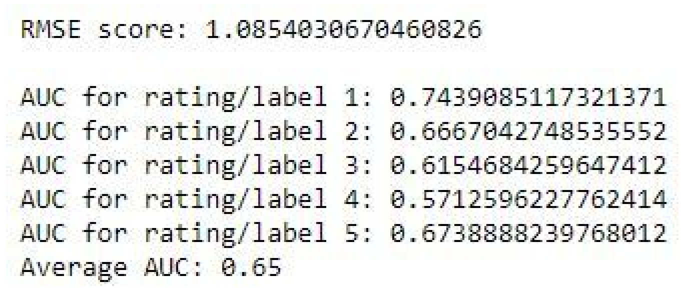
This model was able to improve the RMSE score to 1.64, each of the classes AUC scores and the average AUC score to 0.65. Just to check, the feature importances given by it were



Again, the user in question, the number of ratings given, the ALS prediction and the average rating of the book seem to be the most important features when predicting the rating a user will give.

4.6) Probability calibration

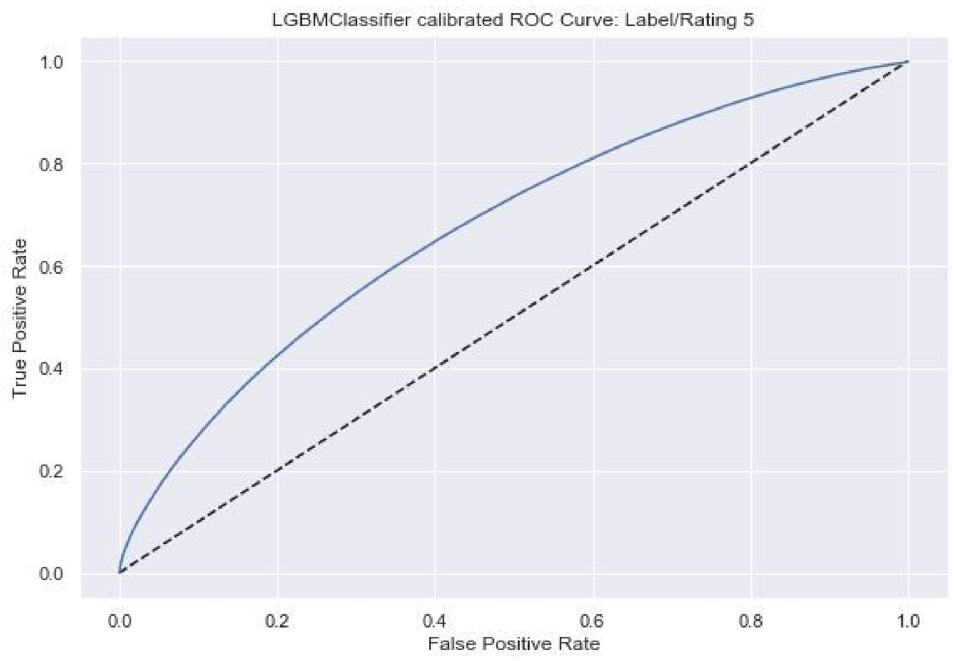
Note that the usage of all these parameters result in poor estimates of the individual class probabilities. This probability gives some kind of confidence on the prediction. Hence, I will perform probability calibration of the model, since these probabilities will be important for the next step. Using sklearn's CalibratedClassifierCV, the results were further improved to



```
RMSE score: 1.0854030670460826

AUC for rating/label 1: 0.7439085117321371
AUC for rating/label 2: 0.6667042748535552
AUC for rating/label 3: 0.6154684259647412
AUC for rating/label 4: 0.5712596227762414
AUC for rating/label 5: 0.6738888239768012
Average AUC: 0.65
```

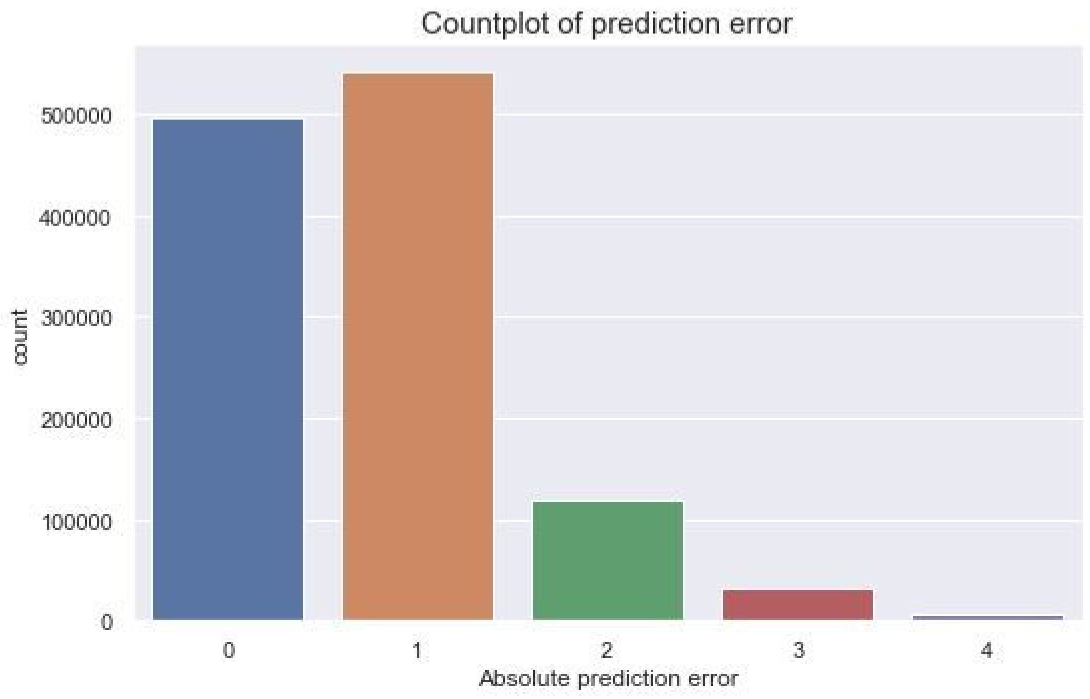
The ROC curve for the highest possible rating, 5, was especially important for the next step:



5) Predictions and recommendations

Now, to actually give recommendations, I used the **highest scoring probabilities of having a rating equal to 5**.

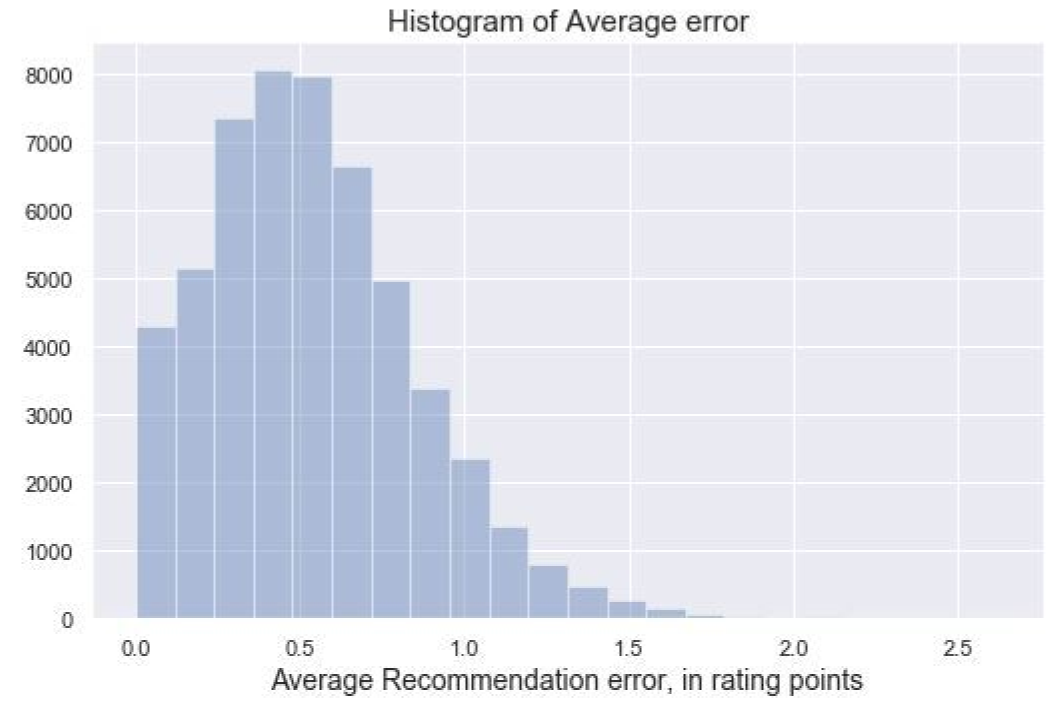
Here is, first, a countplot of the prediction error of the calibrated model - the difference between the predicted rating and the rating itself, in absolute terms.



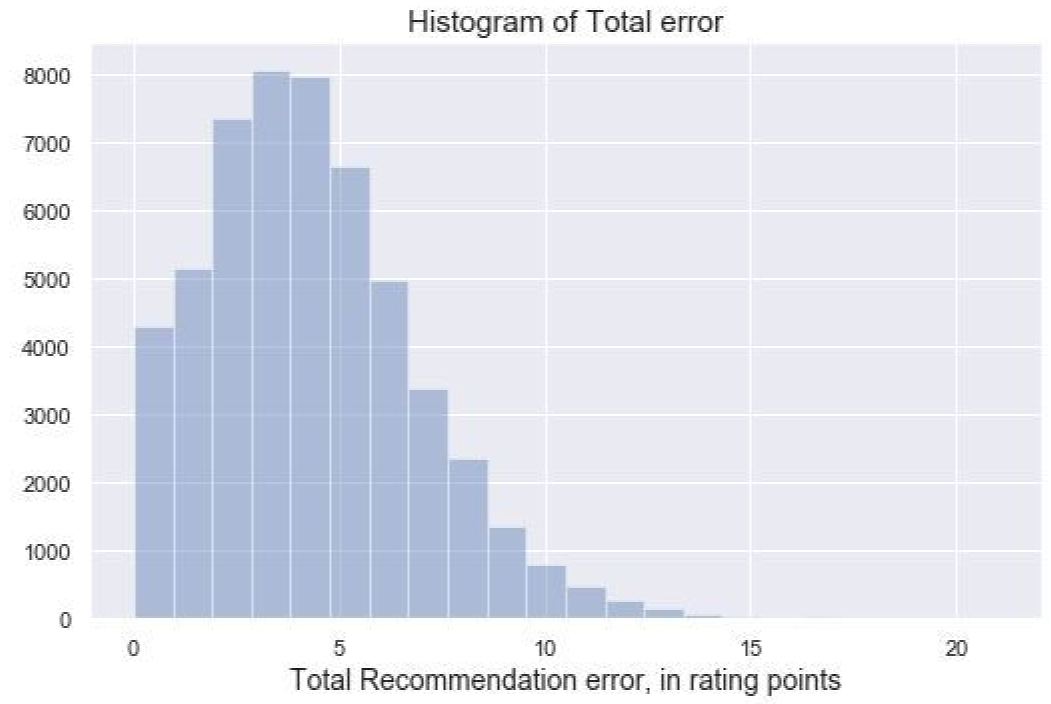
To check the given recommendations (highest scoring probabilities of having a rating equal to 5), I compare them to the highest rating books of the user and take the difference on the rating as a measure of failed recommendation.

Considering all users and 8 as the number of recommendations to give, **the average error of the recommendations was, in rating points, equal to about 0.50,** that is, half a rating point**. The total error averages around 4** (again, considering 8 recommendations to each user).

It is then possible to see how the distribution of these errors behave. First, the histogram of the average error:

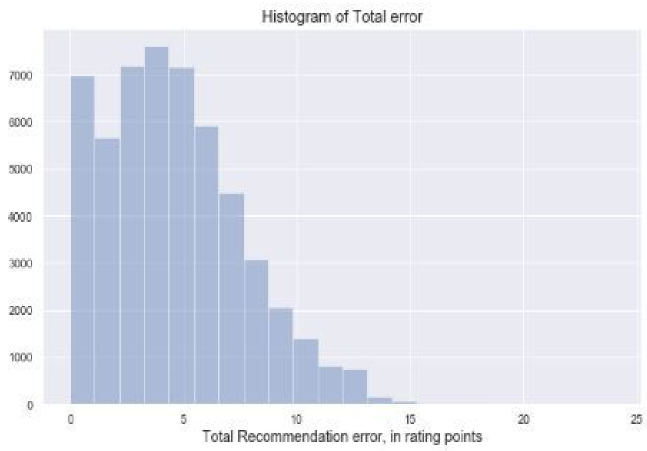


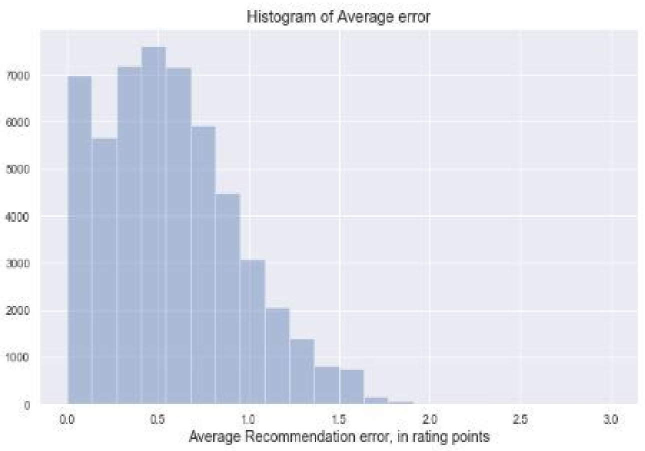
Then, relatively to the “total error”:



When analyzing the distribution of either the total error or of the average error, we see that it exponentially decays when it arrives the mentioned mean values.

I then also analysed the recommendation error when sorting the recommendations not by 5-star highest probability values, but by **highest predicted rating**. In this case, however, the results were not so good, as the average error was equal to 0.58 and the "total error" to 4.7. It also behaved differently, as we can see on their histograms:



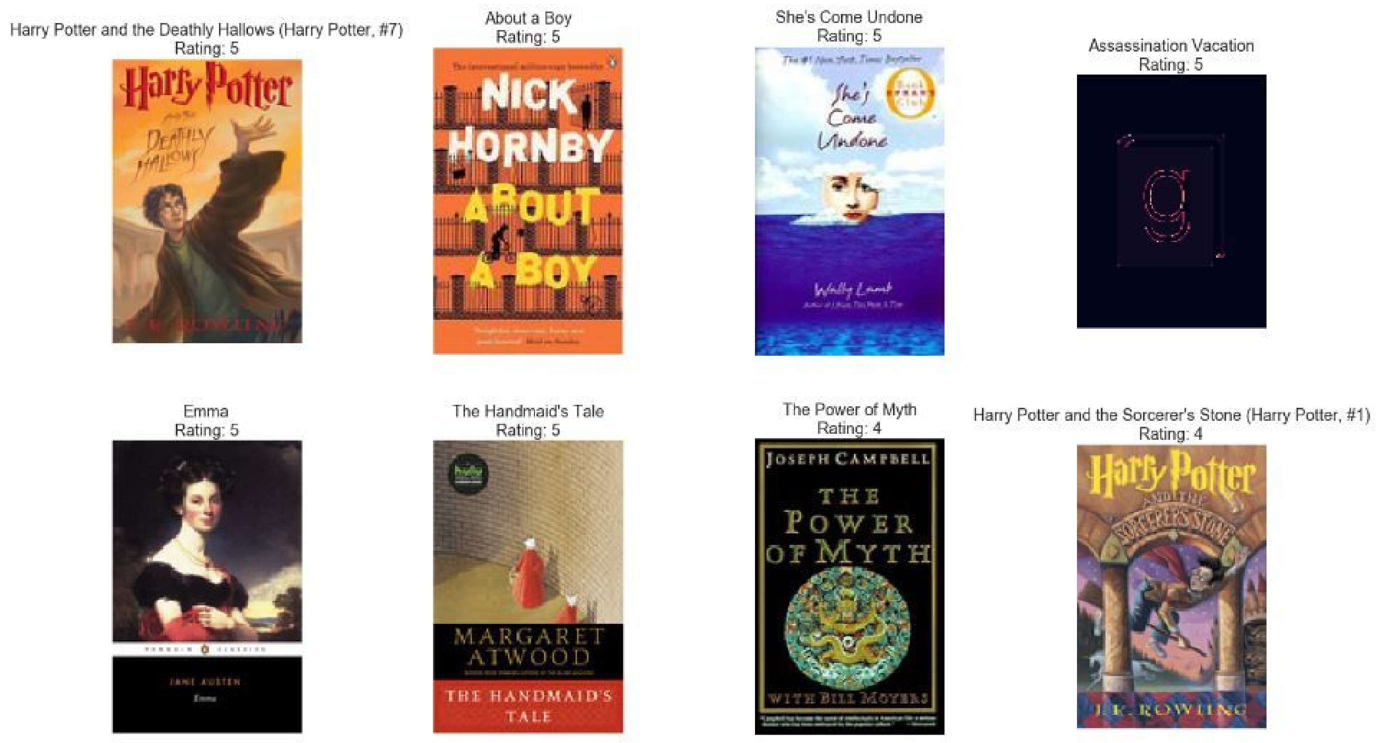


This way, the recommendations was considered as the n highest probabilities of thee chosen model attributing 5 stars.

- Example: for a single user

Taking user nº 15 again as an example, and now considering only the test set, these were the **highest rating books** by him/her:

```
User_15 8 favorite books:
```



Harry Potter and the Deathly Hallows (Harry Potter, #7)
Rating: 5

About a Boy
Rating: 5

She's Come Undone
Rating: 5

Assassination Vacation
Rating: 5

Emma
Rating: 5

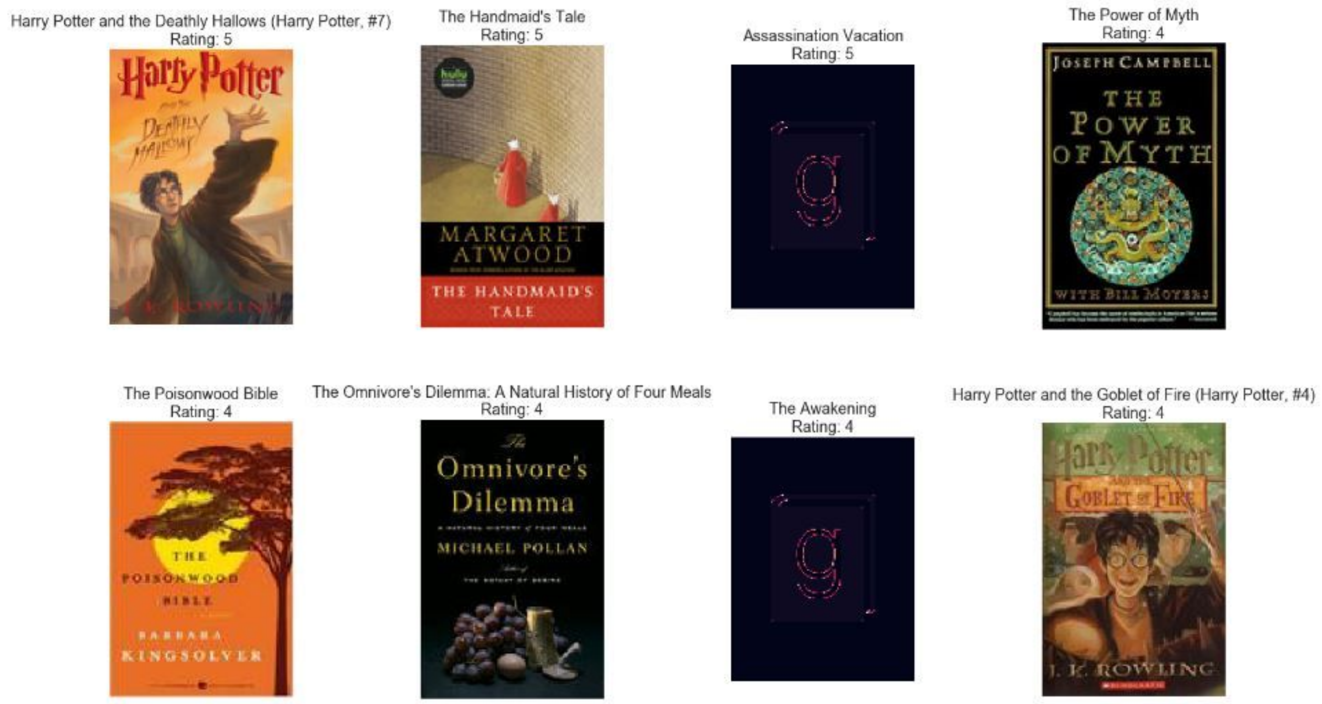
The Handmaid's Tale
Rating: 5

The Power of Myth
Rating: 4

Harry Potter and the Sorcerer's Stone (Harry Potter, #1)
Rating: 4

The **recommendations**, on the other hand, were:

```
User_15 8 recommended books:
```



Harry Potter and the Deathly Hallows (Harry Potter, #7)
Rating: 5

The Handmaid's Tale
Rating: 5

Assassination Vacation
Rating: 5

The Power of Myth
Rating: 4

The Poisonwood Bible
Rating: 4

The Omnivore's Dilemma: A Natural History of Four Meals
Rating: 4

The Awakening
Rating: 4

Harry Potter and the Goblet of Fire (Harry Potter, #4)
Rating: 4

6) Conclusion

The goal of this project was to build a good book recommendation system for users, using Goodreads.com dataset with almost 6 million ratings done by 50K+ users for 10K+ books. We started to explore some variables and in what way they could influence a user rating. We found some patterns, as, for example, the almost linear negative relationship between the number of books a user has and his/her average rating.

The built recommendation system works as follows:

- First, we built a matrix factorization algorithm using ALS (Alternating Least Squares) in order to get implicit feedback. In other words, an algorithm that could predict if the user read a certain book or not, regardless of his opinion of the book. This could be seen as an attempt to study user behavior, in the sense that it tries to predict general/implicit preferences. The test score was pretty good, with a mean AUC of 0.96.
- Using that implicit behavior as an extra feature, we then started to build the recommendation system by testing several algorithms and predict the user rating (explicit behavior). This was made after doing an initial feature selection, which was mainly based on the previous exploratory data analysis phase. Although the one with better results was a Voting Classifier ensemble using 3 of the models, I chose to go with the Light Gradient Boosting (LightGBM) algorithm, which got similar but much faster results.
- Then, through principal component analysis (PCA), Recursive Feature Elimination (RFE) and also considering the feature importances obtained when testing the Random Forest Classifier, further feature selection was made, reducing the chosen features to 7.
- After that, the chosen model was tuned on 4 major parameters. Then, since the usage of all these parameters can usually result in poor estimates of the individual class (rating) probabilities, we went further to perform probability calibration using sklearn's CalibratedClassifierCV. This was important for the next step. The final model got an RMSE score of 1.08 and an average AUC of 0.66.
- The recommendations were based using the highest scoring probabilities of having a rating equal to 5 - the highest score a book can have. After testing with other methods, this one got better results. Against the test dataset it got an average error of 0.5, in terms of rating points. This means that the 8

recommended books were, on average, 0.5 rating points below the 8 highest rating books of the user, that is, on what would be a perfect score (error 0).

I proceeded to check some individual examples. In one instance, 3 of the 8 recommendations matched the user's favorite books, and for example the 4th series of the Harry Potter series was suggested (Harry Potter and the Goblet of Fire), instead of the 1st (Harry Potter and the Sorcerer's Stone), which can be considered close enough - beyond the difference in terms of rating. Hence, we see that although we have a small average error rating (of 0.5), it can happen that these errors encompass books that are not very far away, in terms of content, from the user's favorite ones. It can happen the inverse as well. Ultimately, the objective way to measure is through the attributed rating.