Machine Learning with Python Labs – A22

Date: March 26th, 2023

Professor: Hanna Abi Akl

Project contributors:

Diego Carriel Lopez : diego.carriel@edu.dsti.institute

Romain Duhomme : romain.duhomme@edu.dsti.institute

Lan Le : le.lan@edu.dsti.institute

Teresa Graffi : teresa.graffi@edu.dsti.institute

PROJECT REPORT

Contents

1. Summary.....1

2. Overview of the targeted problem.....2

3. Exploratory Data Analysis and Data Preprocessing.....2

4. Feature engineering and Model selection.....4

5. Model training and evaluation.....**Error! Bookmark not defined.**

6. Deployment.....**Error! Bookmark not defined.**

7. Results and recommendations.....**Error! Bookmark not defined.**

*****

1. Summary

Takeaways:

- Adding data: From open book, retrieve the subject of all books.

- Cleaning data: fill missing values, remove blank spaces, change format date, remove duplicates, delete useless data, rename columns.

- Securing the data: cross validation with Excel workbook.

- Analyzing data: plot all variables with statistical information, correlation analysis.

- Transforming data: create season and centuries from date, fusion of all English languages, transform into categories (from 1 to 5) the variables average_rating, rating_count, max_pages, transform the subject from open book into a genre category.

- Select the data: elimination of biased columns.

- Model used: Random Forest & Adaboost algorithms.

- Results: Recall 74,4% and Precision 75,7%.

2. Overview of the targeted problem

The aim of this machine learning project is to build a model that can help Goodreads users get book recommendations based on authors, book types, and other relevant factors.

We believe that the problem at hand has an average complexity and will require a manageable number of features (independent variables) for the machine learning model.

Questions to consider before starting the project?

- missing information: subject, genre and cost of the book.

How we intend to address the problem?

- find the missing data, or indicators for it, on Google API and Open Books API.

Methodology

The project is conducted with the following steps:

- **Exploratory data analysis** to understand the dataset and **data pre-processing** to clean and format the dataset to ensure it is in a suitable format for machine learning algorithms.
- **Feature engineering** by creating new features or extracting relevant features from the data to improve the model's predictive performance, and **model selection** based on the nature of the data and the required predictive accuracy for predicting book ratings.
- **Model training and evaluation** by dividing the dataset into train and test set to train the model, improve its performance, and evaluate the performance of the model.
- **Deployment** by deploying the trained and evaluated model to a production environment.

3. Exploratory Data Analysis and Data Preprocessing

About the dataset

To achieve our objectives, we used a dataset containing approximately 11,000 rows of data scraped from the Goodreads API. This dataset includes customers' ratings of books, as well as other information such as author, book code, number of pages, count of text reviews, publication date, and publisher obtained from the Goodreads website[1].

Here, questions regarding the volume and accuracy of the data arise. While we believe that the collected dataset may be sufficient to build an ML model, it is crucial to ensure that the dataset is representative of the target population and contains a diverse range of books and ratings. The first step in determining the sufficiency of a dataset is to experiment with exploratory data analysis and data preprocessing techniques to validate the dataset.

It's important to note that the dataset used for this project was last updated in May 2019. This may have a significant impact on the accuracy of the machine learning model, as it may not reflect the current distribution of the target variable or account for new patterns or trends that have emerged since the data was collected. To mitigate this risk, we have decided to limit the scope of the project by focusing only on the period before mid-2019 and analyzing long-term trends that can be detected from the available data.

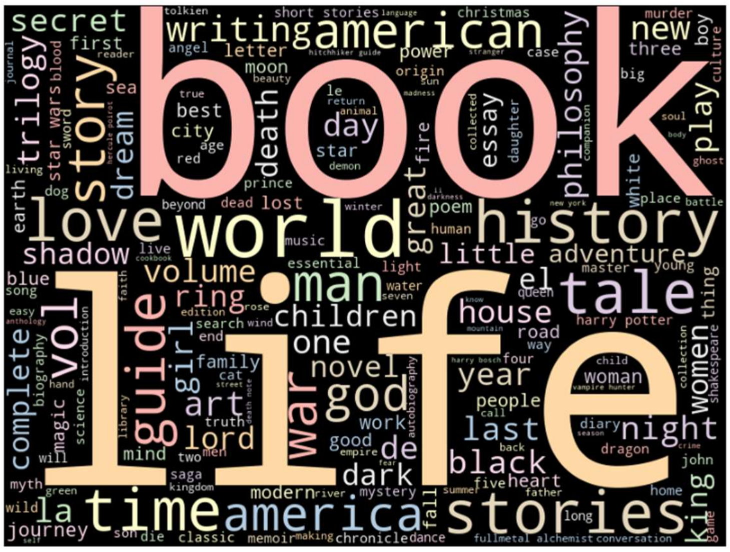


Figure 1: Wordcloud obtained from the titles of the books of the dataset, showing the most frequent words appearing in the titles.

[1] Goodreads is an American social cataloging website and a subsidiary of Amazon that allows individuals to search its database of books, annotations, quotes, and reviews.

<u>*Observations about data*</u>

In this stage, we will be using Excel and Python to conduct EDA and DP processes.

(1) Excel

We decided to use excel for two reasons:

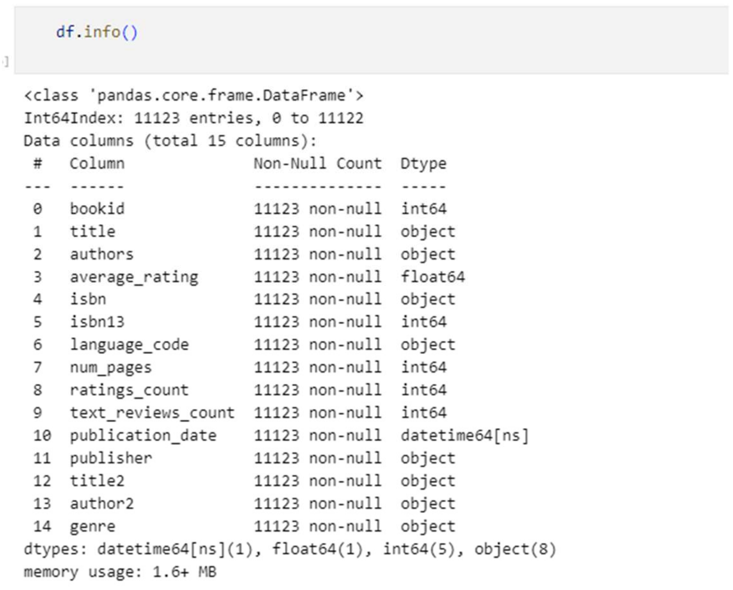
- If the dataset is under 100k rows, it's faster to get a quick overview of the dataset.

- We can doublecheck the transformation of the data on Python and Excel and detect potential mistakes.

(2) Python

In Python, we installed all necessary libraries and imported the dataset using Pandas.

<u>Subject data</u>: We obtained additional data through open books API, the code can be checked on the Jupyter Notebook "GoogleAPIservice". We tried to obtain information from google API, but the data was not exhaustive and the number of interrogation of the database was limited.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11123 entries, 0 to 11122
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   bookid              11123 non-null  int64
 1   title               11123 non-null  object
 2   authors             11123 non-null  object
 3   average_rating      11123 non-null  float64
 4   isbn                11123 non-null  object
 5   isbn13              11123 non-null  int64
 6   language_code       11123 non-null  object
 7   num_pages           11123 non-null  int64
 8   ratings_count       11123 non-null  int64
 9   text_reviews_count  11123 non-null  int64
 10  publication_date    11123 non-null  datetime64[ns]
 11  publisher           11123 non-null  object
 12  title2              11123 non-null  object
 13  author2             11123 non-null  object
 14  genre               11123 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(5), object(8)
memory usage: 1.6+ MB
```

Figure 3: Output of info() function.

We checked the data types using the code *"data.info()"* to get an overview of the dataset's dimensions and description. We identified two types of variables: categorical and numerical variables. In the same time we checked for the presence of missing values.

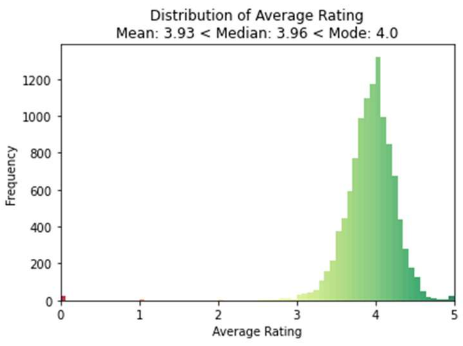


Figure 2: Distribution of 'average_rating'.

Some observations: We found that the dataset contains 11 123 records with 12 columns. There were no missing values, and the data types seemed coherent. We observed some <u>redundancy in the "*titles*"</u> (10 348 unique values) <u>and "*authors*"</u> (6 639 unique values), with Stephen King having the most works.

There were 27 unique "*languages*", with English books dominating the dataset (8 908 values). There were 2 290 unique "*publishers*", with Vintage being the most common (318 works).

The <u>mean "*average_rating*" was 3.93</u>, with 25% at 3.77, the median at 3.96, and 75% at 4.14. Ratings were mostly high, and <u>the spread was small (0.35)</u>. Skewness seemed low. The mean of "*num_pages*" was 336, with the median at 299, and it exhibited some skewness because of outliers (with the top value of 6 576).

The mean "*ratings_count*" was 1.79e+04, with the median at 7.45e+02, exhibiting strong skewness as the top value was > 1e+6. The mean "*text_reviews_count*" was 542, with the median at 47, exhibiting high skewness with a top value of 94 265.

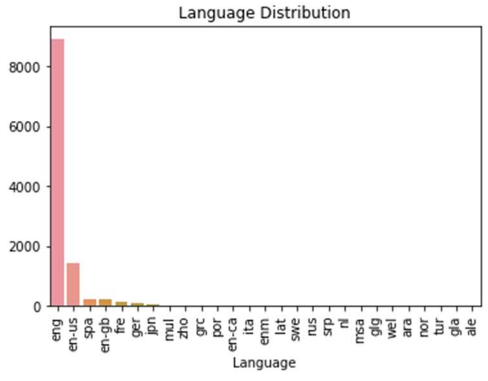
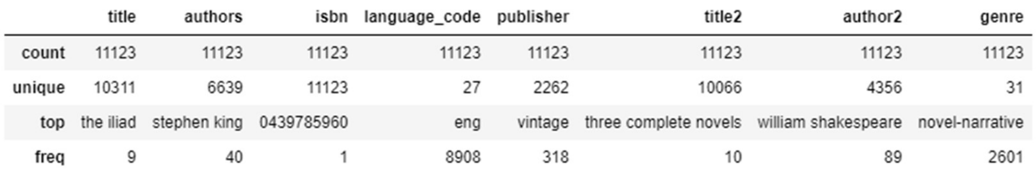
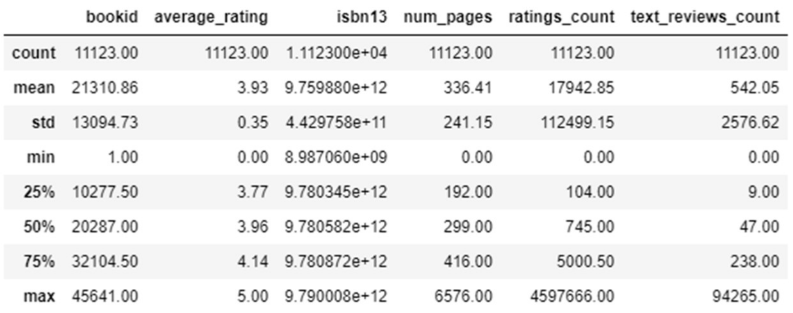


Figure 4: Distribution of the 'language' variable.



	title	authors	isbn	language_code	publisher	title2	author2	genre
count	11123	11123	11123	11123	11123	11123	11123	11123
unique	10311	6639	11123	27	2262	10066	4356	31
top	the iliad	stephen king	0439785960	eng	vintage	three complete novels	william shakespeare	novel-narrative
freq	9	40	1	8908	318	10	89	2601

Figure 6: Output of the describe method applied on categorical variables.



	bookid	average_rating	isbn13	num_pages	ratings_count	text_reviews_count
count	11123.00	11123.00	1.112300e+04	11123.00	11123.00	11123.00
mean	21310.86	3.93	9.759880e+12	336.41	17942.85	542.05
std	13094.73	0.35	4.429758e+11	241.15	112499.15	2576.62
min	1.00	0.00	8.987060e+09	0.00	0.00	0.00
25%	10277.50	3.77	9.780345e+12	192.00	104.00	9.00
50%	20287.00	3.96	9.780582e+12	299.00	745.00	47.00
75%	32104.50	4.14	9.780872e+12	416.00	5000.50	238.00
max	45641.00	5.00	9.790008e+12	6576.00	4597666.00	94265.00

Figure 5: Output of the describe method applied on numerical variables.

Actions:

- We renamed the columns correctly, converted everything to lowercase and removed blank spaces in the "num_pages" column name.
- We converted every string to lowercase except for the ISBN since it is an identifier. We also corrected some data types, such as changing the "*publication_date*" to MM-DD-YYYY format date (since it was assimilated as an object).
- We also inspected duplicates in the dataset and found 812 titles in duplicates. We decided to keep only the observation with the max "*average_rating*" and the less recent "publication_date" for each title.
- After identifying many observations with a number of pages equal to zero, we decided to deal with it in 4 steps. First step is looking for the highest value of number of pages in observations with the same book title and storing it in "max_pages". Second and third step is computing the average number of pages by author and by publisher and filling "*num_pages*" with the computed averages. Last step is look for the remaining data online.
- We identified some rows with "*ratings_count*", "*text_reviews_count*" equal to zero, and in the same row "*average_rating*" either zero or five. We decided to drop these rows.

Completing this step helped us gain a deeper understanding of the dataset and prepared it for the next steps of feature engineering and model selection.

4. Feature Engineering and Model Selection

The publication date can provide insights on the context of the books, e.g., books published in the winter season could have different ratings than those published in the summer season. The number of pages can also potentially affect the book rating, as readers may have different preferences for longer or shorter books. Additionally, the number of ratings a book receives can be an indication of its popularity and can help to identify books that are widely read, and so on.

Identifying such patterns or trends can be useful in creating predictive models to identify which books are likely to receive high or low ratings.

Therefore, we created some new features to include in our predictive model:

- For ***"publication_date"***, we opted to split the data by season[2] and by century with corresponding numerical indications. The *'season'* and *'century'* columns could provide valuable insights into temporal patterns in book publishing and help identifying which periods and seasons are most popular for book releases.

[2] The seasons were defined as follows: Winter (1), Spring (2), Summer (3) and Autumn (4)

- We found '***languages'*** with the values of "*en-us*", "*en-gb*", and "*en-ca*", and grouped them into the "*language_code_ENgroup*" category. This column will be useful to see which English-speaking countries are most active in publishing books.
- For the ***"rating_count"*** column, we summed together the value of rows with the same book title and transformed this into categories with a log function. We defined the bins using a logarithmic scale since the distribution of the "*ratings_count_updt*" column was highly skewed, making logarithmic bins a good choice for creating more evenly distributed categories. We then defined the labels for each category as ['very low', 'low', 'medium', 'medium-high', 'high', 'very high'], assigned a number (from 1 to 6) to each category, and created a new column with these category labels.
- Another feature has been created transforming ***"max_pages"*** into categories by using a classic splitting method. We decided to define bin ranges as [0, 50, 200, 400, and >= 1000], with corresponding labels of ['very low', 'low', 'medium', 'high', 'very high'].

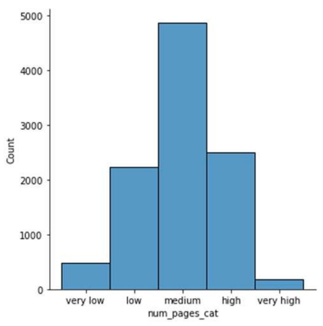


Figure 7: Distribution of observations according to num_pages categories.

- We transformed also the ***"average_rating***" into discrete and more meaningful categories of ['very bad', 'bad', 'medium', 'good', 'very good'] - using a logarithmic function and tried a second division in 3 bins in '*average_rating_3cat'*.
- We identified the '***genres'*** using str.contains() methods and keywords found within the value of *'Subject_list'* column. Initially we filled NA values with *'not specified*', than we repeated the str.contains() method on "*title*" column. The rows that didn't fall into one of the 28 genres have been grouped into the '*mixed'* value. The *'genre'* column seems interesting as it was created by extracting subjects and topics dealt in the book and using them to identify the genres using some key words. This could help grouping and comparing books based on their genres and see which genres are most popular and successful. The genre has been transformed into numerical values assigning a random unique integer to each genre.

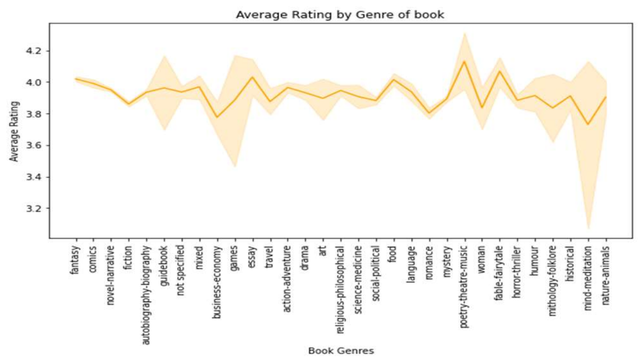


Figure 8: Ratings distribution according to the genre of the books.

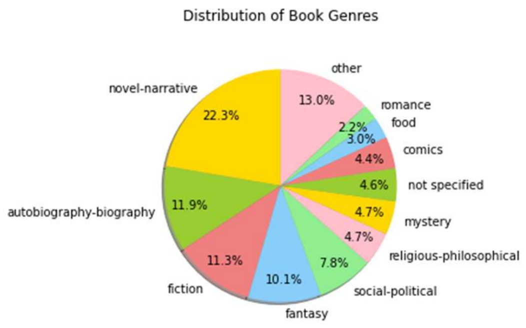


Figure 9: Pie Chart showing how genres are distributed.

We re-verified the dataset. As a result, we obtained a well-prepared dataframe with 10283 observations and 60 columns with no missing values. The dataset contains a lot of useful information about books and can be used to analyze various aspects of the book industry such as the prediction of book ratings.

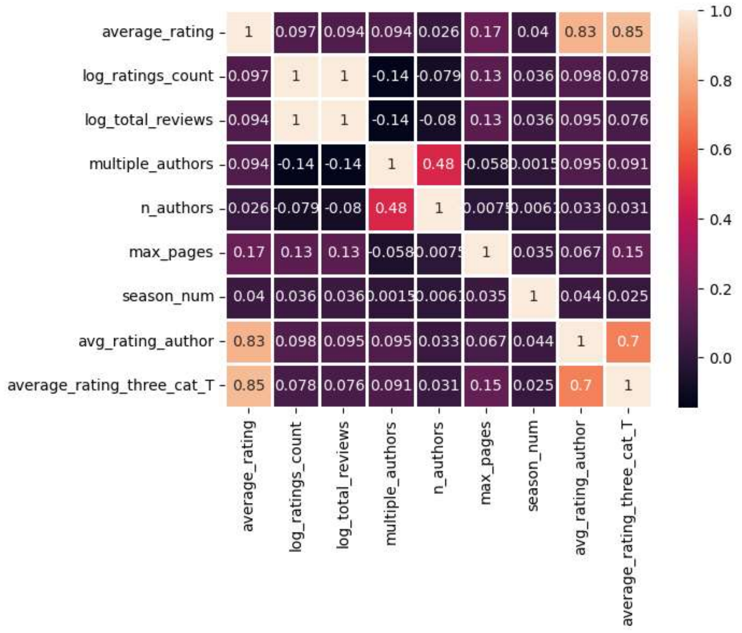
Some observations at this stage:

- After removing duplicates, we have 10 283 unique '*titles'*;
- In terms of *'authors'*: we now have 6 252 unique values in the '*authors'* column and 4164 unique values in the '*main_author'* column. Stephen King was also the author with the most books in the dataset;
- We observed 27 unique '*languages'* in the dataset, with English books dominating (8908 books). We created an additional column to group languages related to English, and now there are 9736 English books in the dataset;
- There were 2262 unique '*publishers*', with vintage being the most common (318 works). Even after processing the data, 'vintage' still appears to be the most frequent publisher with 293 observations;
- We found 31 unique *genres*, with *'novel-narrative'* being the most popular.

- Most books (3002) had a *'rating_count'* that fell under the 'medium high' category, whereas the most frequent category for *'text_reviews'* was 'low' with 4448 observations. If we consider the total number of reviews, the most popular category is 'medium high' with 3046 observations.
- The difference between *'num_pages'* and *'max_pages'* was very low. The most frequent category for both columns was *'medium*,' with 4 865 and 4 897 observations, respectively.
- Finally, with 9384 observations, *'good'* was the most frequently occurring *rating category* in the dataset.

We have selected the following parameters to address the challenge of predicting the average ratings of a book for regression:

Dataset use for simulations	**Example**	**Comment**
log_ratings_count	3.441239	log transform of ratings_count
log_total_reviews	3.473648	log transform of total_reviews
multiple_authors	0	1 is true , 0 is false
n_authors	1	number of authors
max_pages	192	max number of pages
season_num	4	published season derived from date
avg_rating_author	4	derived variable from ratings



	average_rating	log_ratings_count	log_total_reviews	multiple_authors	n_authors	max_pages	season_num	avg_rating_author	average_rating_three_cat_T
average_rating	1	0.097	0.094	0.094	0.026	0.17	0.04	0.83	0.85
log_ratings_count	0.097	1	1	-0.14	-0.079	0.13	0.036	0.098	0.078
log_total_reviews	0.094	1	1	-0.14	-0.08	0.13	0.036	0.095	0.076
multiple_authors	0.094	-0.14	-0.14	1	0.48	-0.058	0.0015	0.095	0.091
n_authors	0.026	-0.079	-0.08	0.48	1	0.0075	0.0061	0.033	0.031
max_pages	0.17	0.13	0.13	-0.058	0.0075	1	0.035	0.067	0.15
season_num	0.04	0.036	0.036	0.0015	0.0061	0.035	1	0.044	0.025
avg_rating_author	0.83	0.098	0.095	0.095	0.033	0.067	0.044	1	0.7
average_rating_three_cat_T	0.85	0.078	0.076	0.091	0.031	0.15	0.025	0.7	1

Figure 10: Heat map of correlation matrix of variables used for the regression models. Targets of the regression were average_rating and average_rating_three_cat_T.

The transformations of the ratings count, and total reviews were performed in order to have a less cluttered distribution of data points. In the same way, the season was used from the publishing date instead of other variables. The idea behind these selections was to keep numerical and categorical variables that were interesting for the regression model. In the case of categorical ones, since their distributions were highly skewed, we took only season numbers and not language or genre. However even though in both languages and genre categories were interesting variables, their utilization added a high number of columns, we some cases showing classes with very small support. Some experiments were used with these variables, but they only added complexity and noise in the regression.

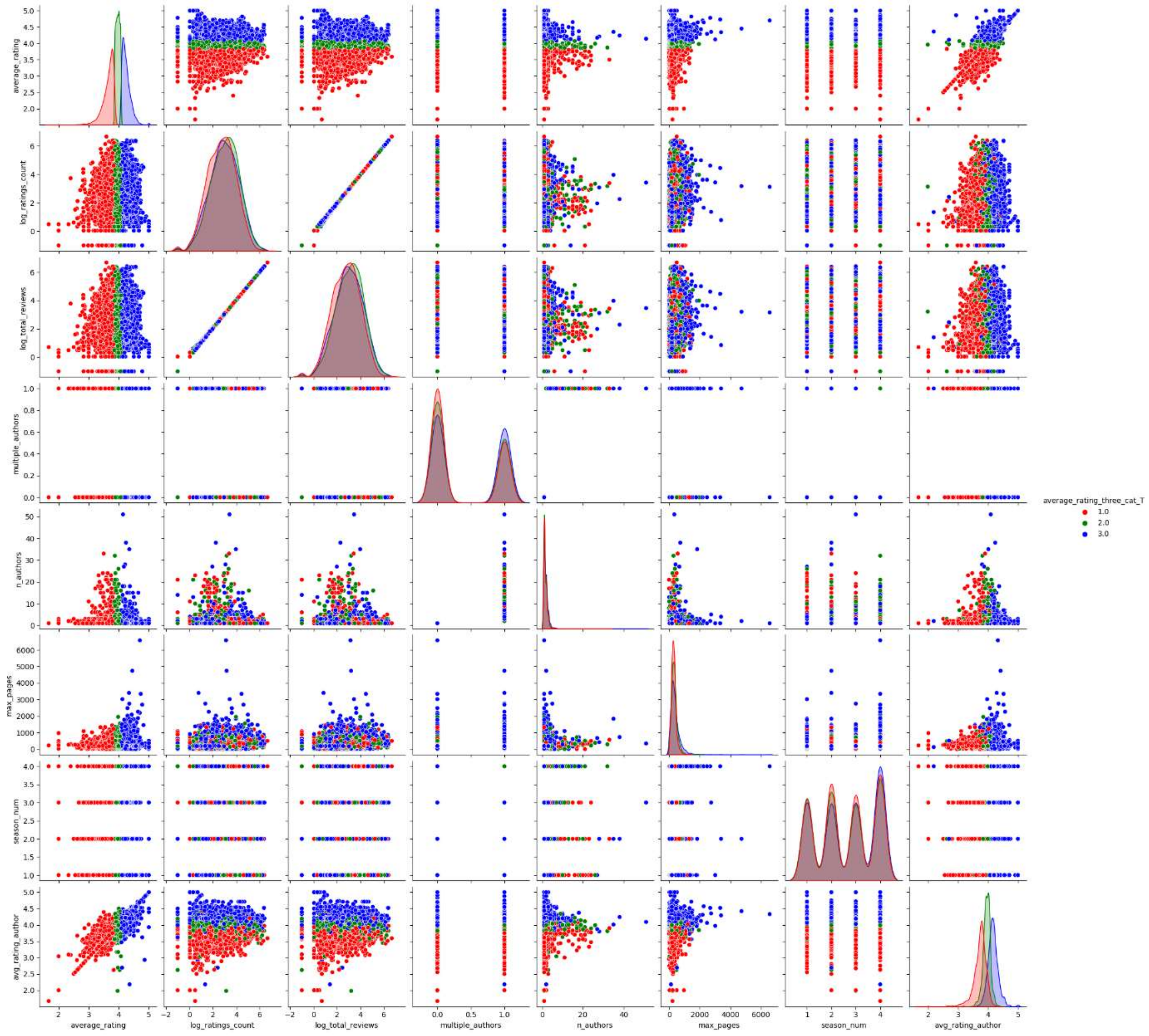


Figure 11: Plot showing the pairwise distribution of the variables.

In the upper panel, we find a pairwise distribution of the variables selected for the regression modeling. We observe some correlated distributions but most of the variables seem to be orthogonal in a pair-to-pair comparison.

From our data analysis, we decided to take parallel strategy to prepare data for performing classification. Since the target column was converted from a continuous to a categorical variable by log transformation and CDF derivation.

Compared to the regression variables we decided to take into account language_code and genre, but also more information about dates and categorical data from the ratings count.

Dataset use for simulations	**Example**	**Comment**
language_code	en-us	language information
genre	food	genre
multiple_authors	0	1 is true , 0 is false
n_authors	1	number of authors
max_pages	192	max number of pages
avg_rating_author	4,22	derived from ratings
pub_month	9	publication month
pub_year	2003	publication year
season_num	4	season
log_ratings_count	3,44	log transform of ratings_count
ratings_count_category	medium high	rating_count_category
log_total_reviews	3,4736	log transform of total_reviews

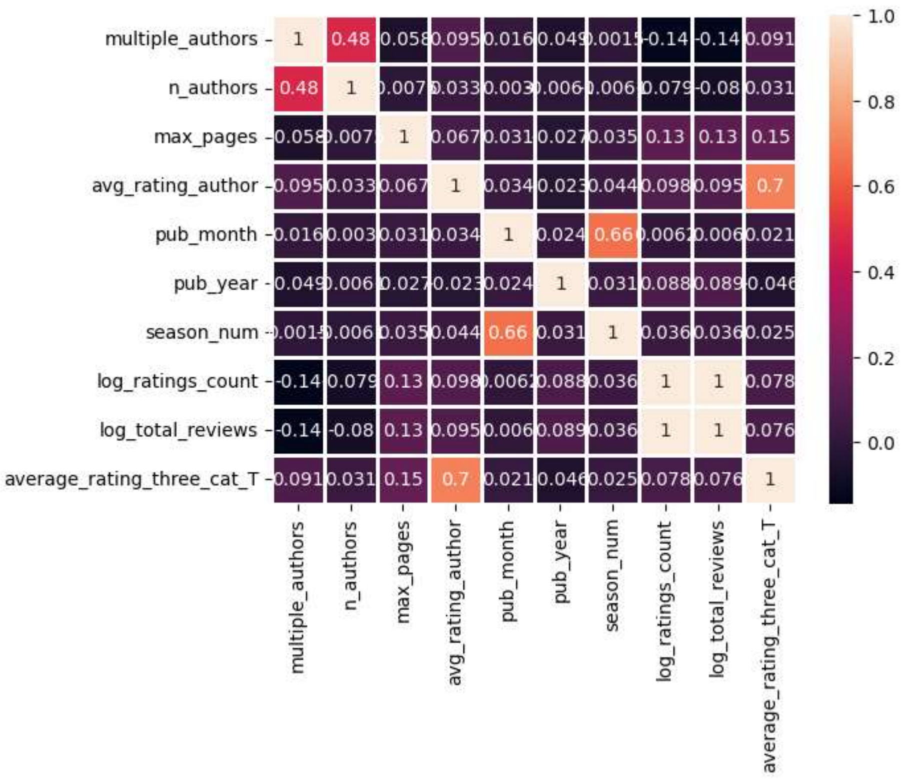


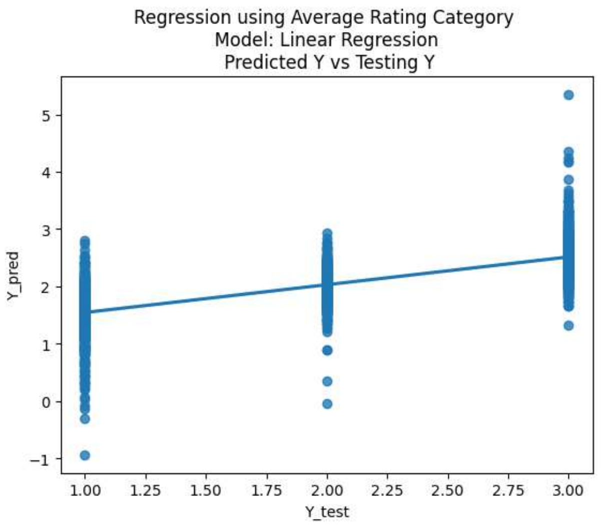
Figure 12: Heat map of the variables used for the classification.

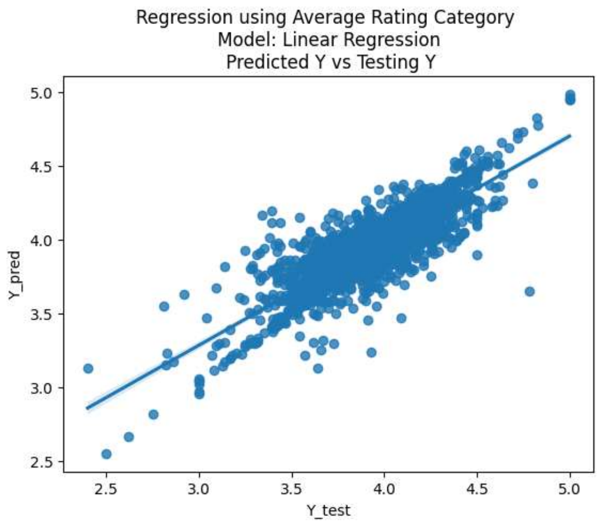
In the upper panel, heat map of correlation matrix of variables used for the classification models. Targets of the regression was average_rating_three_cat_T.

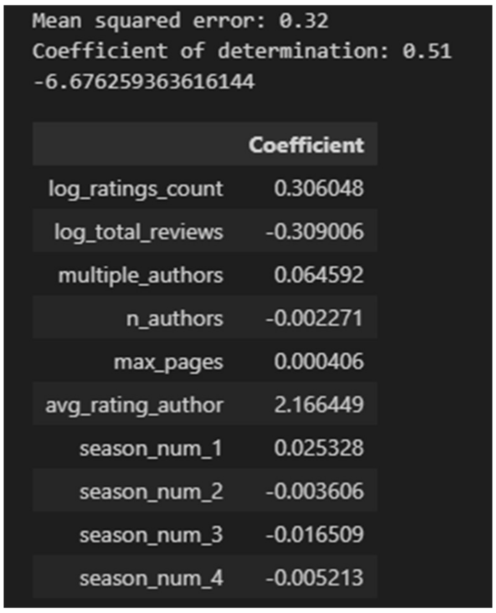
1. **Model training and evaluation**

5.1 Regression algorithms

After doing the data cleaning and feature selection. We have used the 7 selected variables to see if we could predict the average_rating of the books, but also to see the impact of the categorization of the target variable into 3 different categories. The training method involved the splitting of the data as 80% train dataset and 20% test dataset. Then classical linear regression from the sklearn library was used with standard parameters. Concerning the metrics to evaluate the performance of the models we used the mean squared error and the R^2 coefficient of correlation.

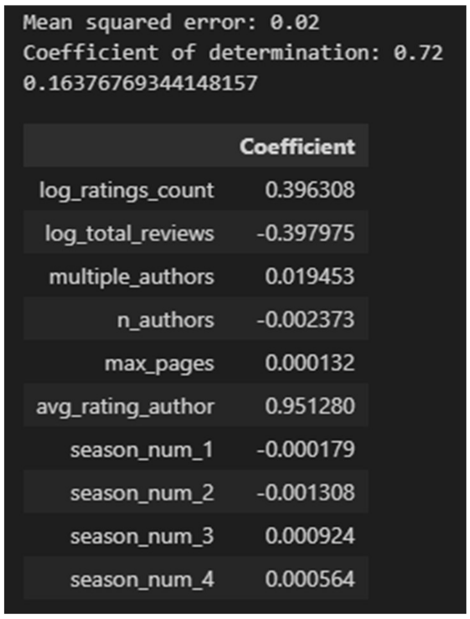






```
Mean squared error: 0.32
Coefficient of determination: 0.51
-6.676259363616144
```

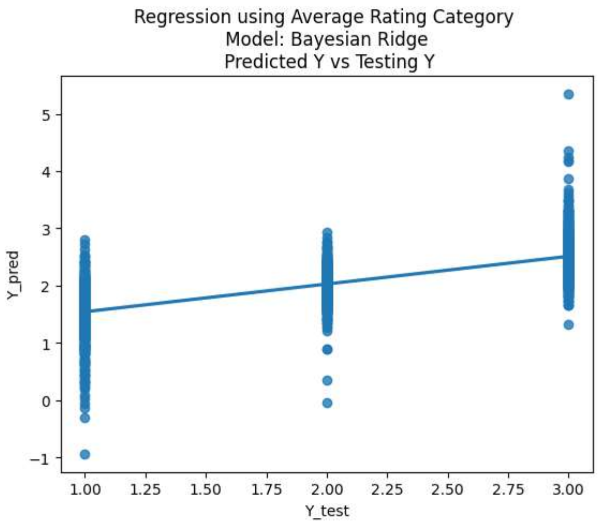
	Coefficient
log_ratings_count	0.306048
log_total_reviews	-0.309006
multiple_authors	0.064592
n_authors	-0.002271
max_pages	0.000406
avg_rating_author	2.166449
season_num_1	0.025328
season_num_2	-0.003606
season_num_3	-0.016509
season_num_4	-0.005213

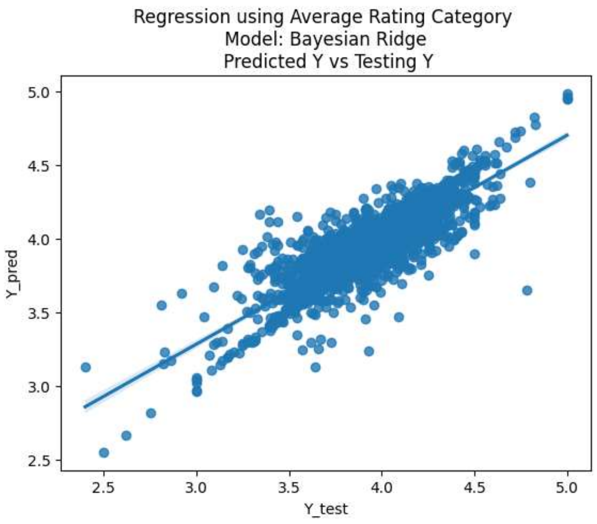


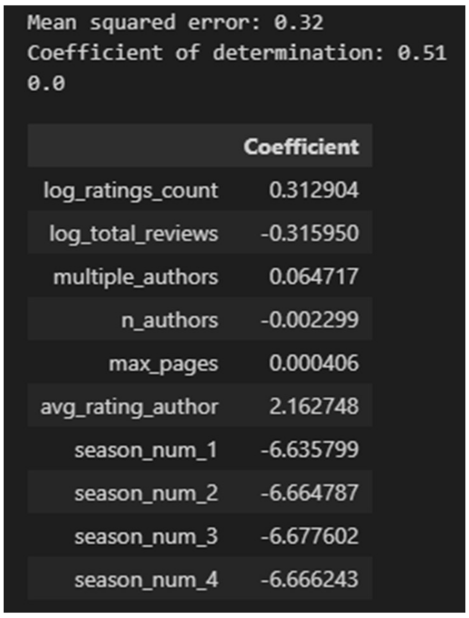
```
Mean squared error: 0.02
Coefficient of determination: 0.72
0.16376769344148157
```

	Coefficient
log_ratings_count	0.396308
log_total_reviews	-0.397975
multiple_authors	0.019453
n_authors	-0.002373
max_pages	0.000132
avg_rating_author	0.951280
season_num_1	-0.000179
season_num_2	-0.001308
season_num_3	0.000924
season_num_4	0.000564

In the upper panel to the left we see the prediction of the categorized target average ratings based on the CDF of the log-transformed normal distribution. In the right we see the regression of the average ratings category without transformation. The results that we obtain is that we have a quite modest correlation in the left regression with an important error (0.32) compared to the expected values (1, 2 or 3). The coefficients show us that the most important variable is ***"avg_rating_author"*** and that the ratings and total reviews seem to cancel each other as they have opposed coefficients. The other coefficients seem to be very small when compared to their range of values and seem to be not very significant in the prediction. In the case of the prediction of the original ***"average_ratings"*** variable, we see very similar coefficients tendencies, but an interesting correlation of 0.72 with a somewhat small mean squared error (0.02)

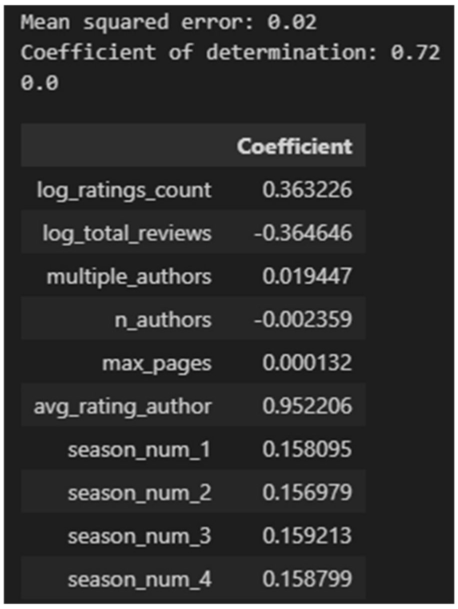






```
Mean squared error: 0.32
Coefficient of determination: 0.51
0.0
```

	Coefficient
log_ratings_count	0.312904
log_total_reviews	-0.315950
multiple_authors	0.064717
n_authors	-0.002299
max_pages	0.000406
avg_rating_author	2.162748
season_num_1	-6.635799
season_num_2	-6.664787
season_num_3	-6.677602
season_num_4	-6.666243



```
Mean squared error: 0.02
Coefficient of determination: 0.72
0.0
```

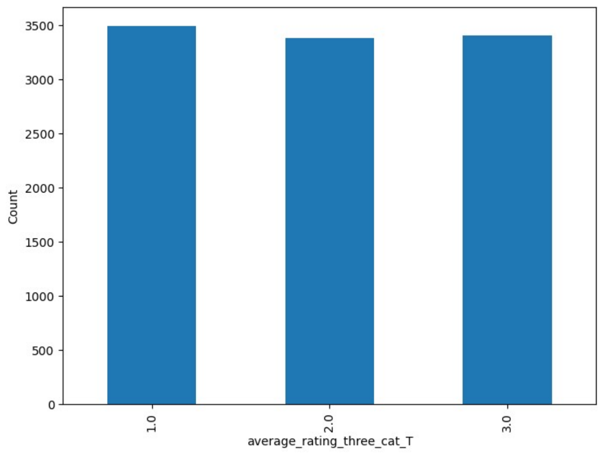
	Coefficient
log_ratings_count	0.363226
log_total_reviews	-0.364646
multiple_authors	0.019447
n_authors	-0.002359
max_pages	0.000132
avg_rating_author	0.952206
season_num_1	0.158095
season_num_2	0.156979
season_num_3	0.159213
season_num_4	0.158799

The second regression algorithm that we used was the Bayesian Ridge Regression. This technique is a way to estimate the relationship between a dependent variable (what you want to predict) and one or more independent variables (features or predictors) using probabilistic modeling. The algorithm assumes that the relationship between the dependent variable and the independent variables follows a linear function, but it also takes into account the uncertainty in the data and the parameters of the model. It uses a Bayesian approach to estimate the parameters of the linear model and to make predictions. Overall, the Bayesian Ridge Regression algorithm is a powerful and flexible method for linear regression that can handle noisy and incomplete data and provides uncertainty estimates for the predictions.

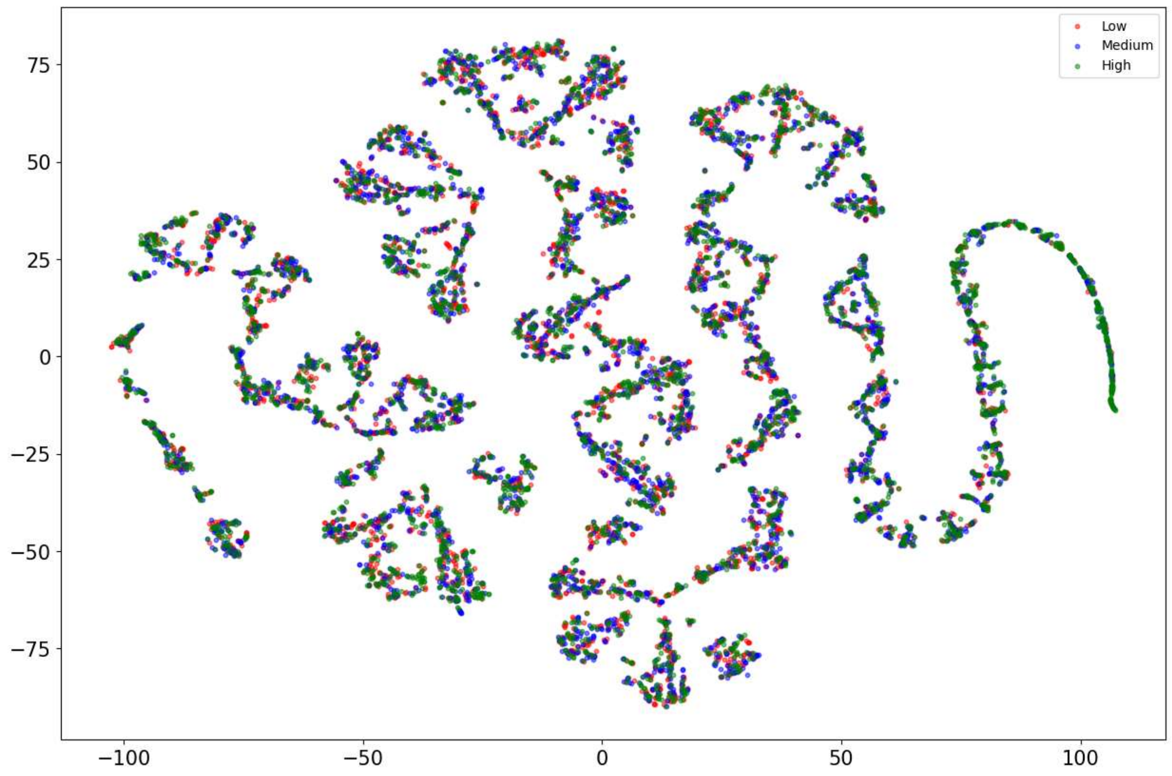
In comparison with the classic linear regression, we observe similarities in the coefficients that were found. For instance, it is interesting to see that this regression puts significantly higher coefficients for the season variables compared to the precedent method, and the importance of the average rating of the author is also taken into account. Nevertheless, the performance of both algorithms with our dataset is roughly the same.

5.2 Classification algorithms

The second strategy of our project was to use a classifier in order to be able to predict whether a book could be considered "bad", "medium" or "good". As described in the feature engineering chapter we transformed the target variable average ratings based on its CDF and we obtained a distribution with an equivalent number of members.



As our data was multidimensional, we wanted to use some visualization technique to see if we could get some understanding of the distribution of the variables. We performed t-SNE which is a machine learning technique used for data visualization, particularly for high-dimensional data. It is a dimensionality reduction technique that helps to visualize complex data in a lower-dimensional space (typically 2D or 3D) by preserving the similarities between data points. The method works by computing a probability distribution for pairs of high-dimensional objects such that similar objects have a higher probability of being chosen. It then constructs a low-dimensional map where the objects are placed so that the pairwise similarities are preserved as much as possible.



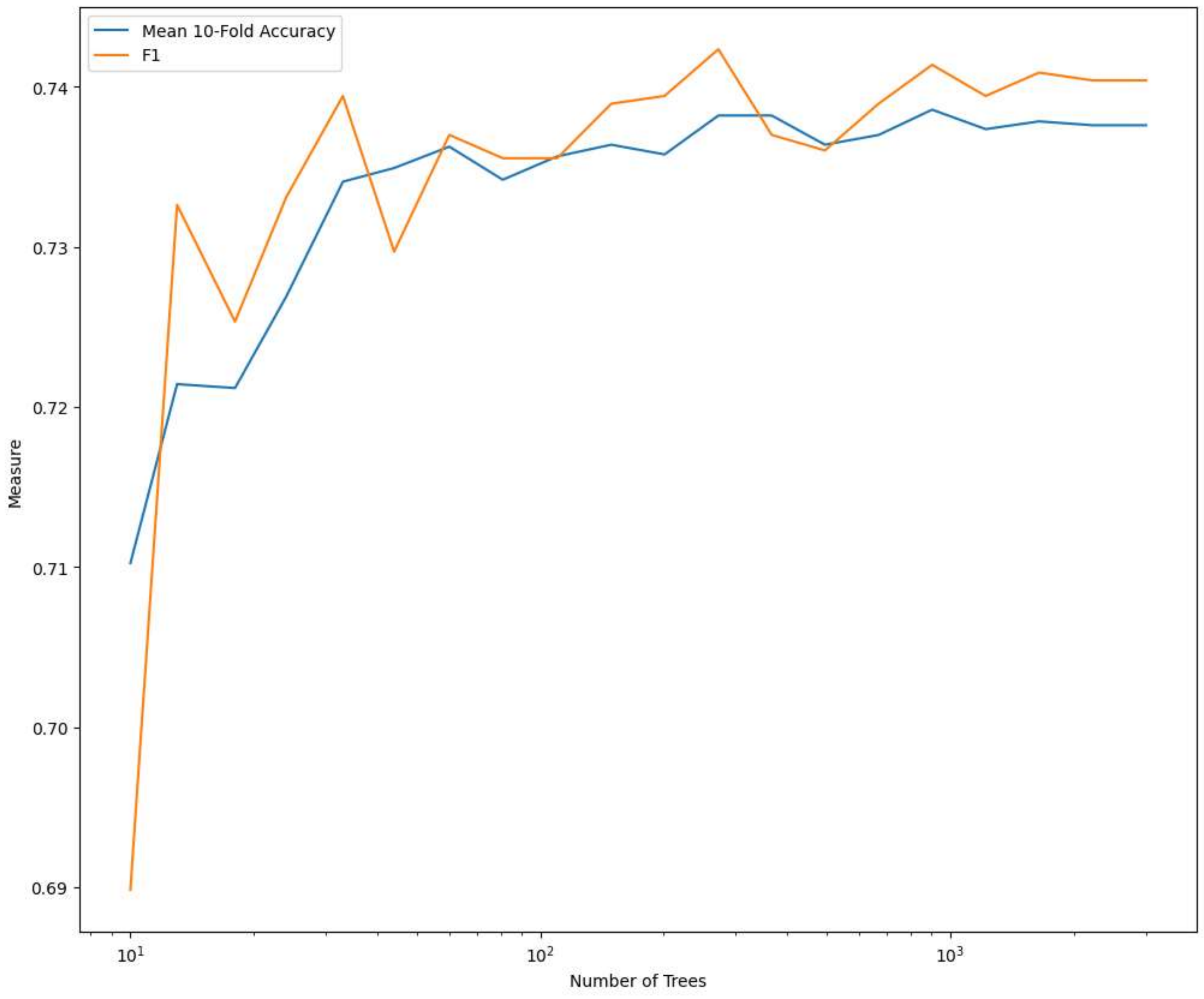
The result of using this technique show the clustering of some observation but in a way that is not clearly dependent on the rating variable suggesting that the clustering is based on some other feature in our dataset.

After this analysis we proceed with the Random Forest algorithm to predict our transformed target variable ***"average_ratings".*** Random forest is a machine learning algorithm that is used for both regression and classification tasks. It is based on the idea of ensemble learning, which combines multiple individual models to make more accurate predictions.

In random forest, multiple decision trees are created and combined to form a "forest". Each decision tree is built by randomly selecting a subset of the available features and a subset of the data points, a process known as bootstrapping. This randomness helps to prevent overfitting and improves the accuracy of the model.

Once the decision trees are built, new data is classified or predicted by passing it through each decision tree in the forest. The final output is determined by combining the predictions of all the decision trees. This process is called "voting" in classification tasks, where the most common class predicted by the trees is chosen, and "averaging" in regression tasks, where the average of the predicted values is taken.

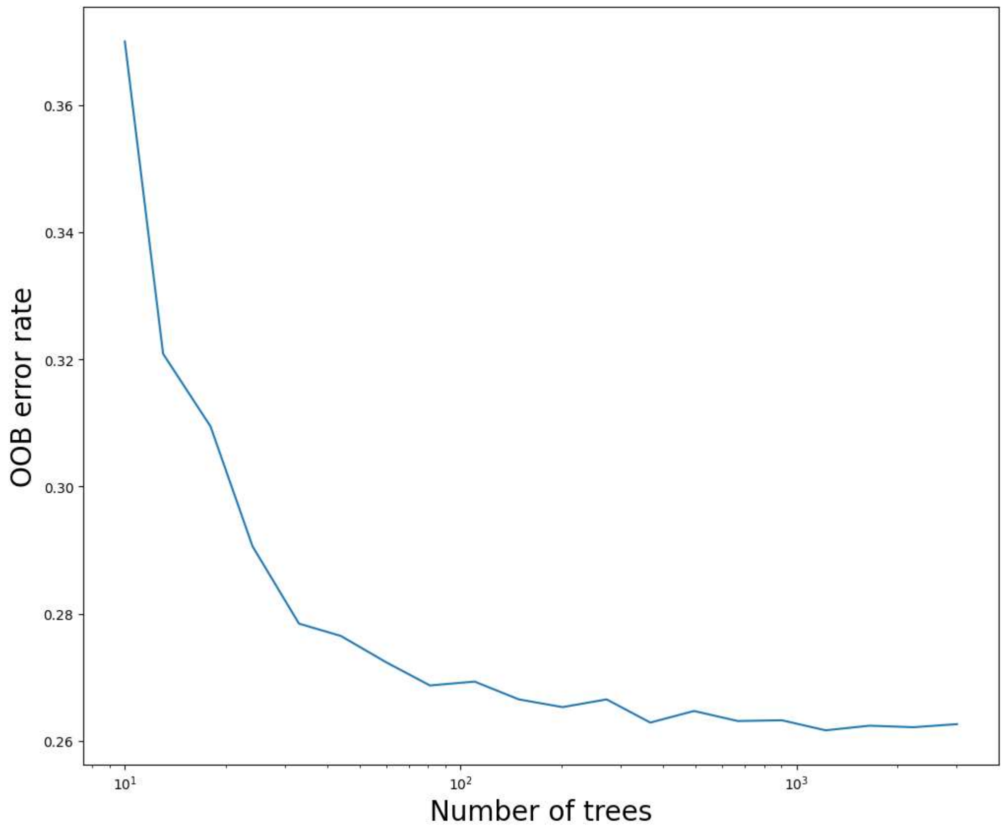
In order to find the best parameters of our model we used the cross-validation methods available from the sklearn library. Our principal objective was to know which was the optimal amount of n_estimators (trees) to get the best performance of the model.



The upper panel shows the relationship between the performance scores, notably the mean 10-Fold Accuracy and the F1 in relation to the number of trees used by the Random Forest Classifier. What we observe is a trend that grows but stabilizes after 100 trees.

In parallel we performed an error measurement of the Random Forest using the technique (out-of-bag error) OOB. The OOB error is a measure of the performance of a random forest model. It is calculated by using the data points that were not included in the bootstrap sample used to train a particular decision tree. These data points are called out-of-bag (OOB) samples. The OOB error is calculated by predicting the OOB samples using the decision tree that was not trained on those samples, and then comparing the predicted values to the actual values.

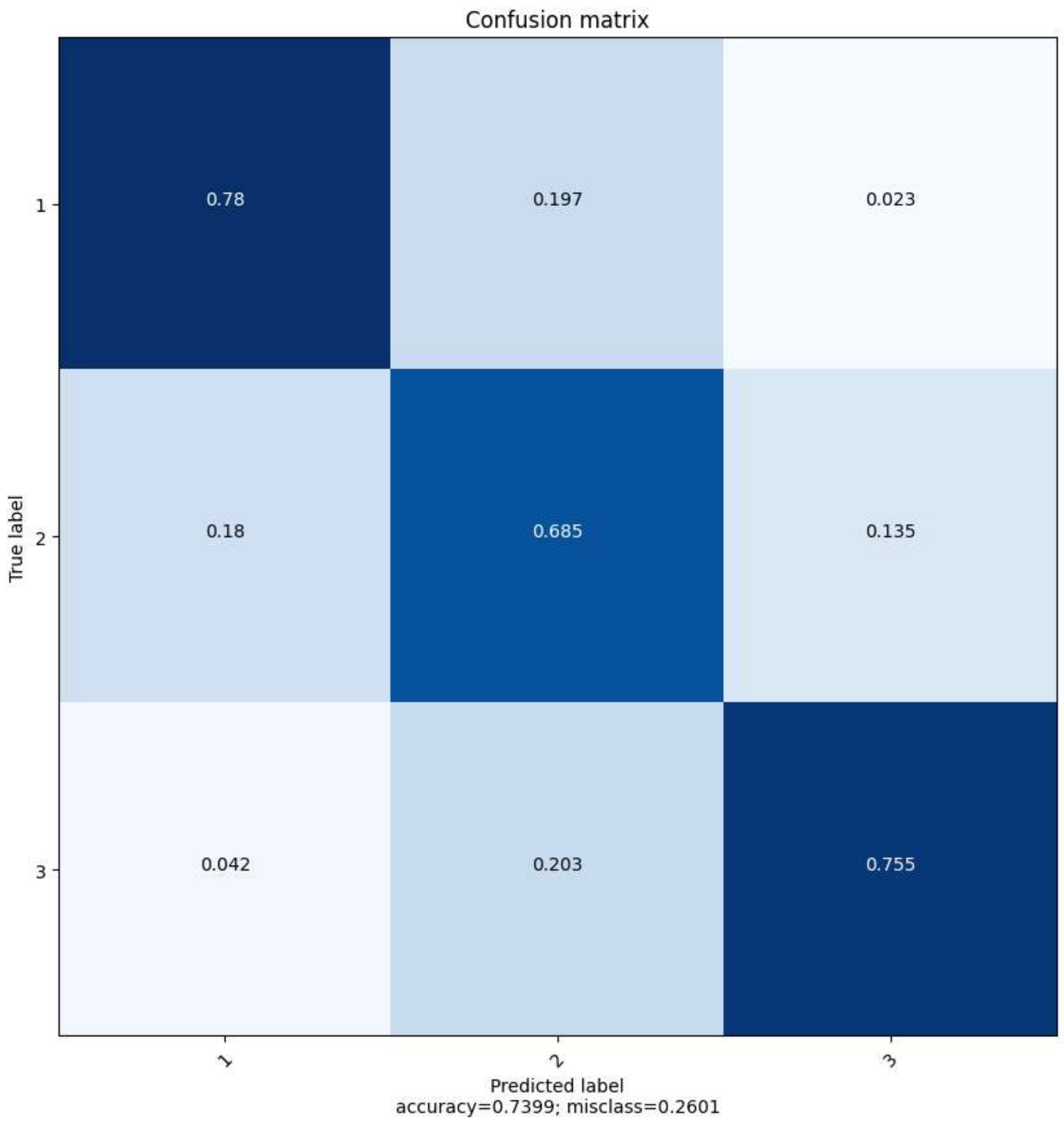
The OOB error provides an estimate of how well the random forest model is likely to generalize to new data. This is because the OOB samples were not used in training the decision tree, so they represent new data that the model has not seen before. By using these samples to calculate the OOB error, we can get an idea of how well the model is likely to perform on new, unseen data.



Concomitantly, we the rise in performance with the number of trees of the Random Forest, we that the OOB error rate has decreasing rate that stabilizes after 100 trees.

With this data we decided to perform the Random Forest Classifier with number of 400 trees and to get the detailed confusion matrix of the algorithm prediction.

The confusion matrix shows that the classifier exhibits an accuracy of 0.7399 and thus a misclass of 0.26. Graphically we observe that most of the classification errors come from adjacent classes and that discrimination is quite reliable when it depends on differentiating class 1 from class 3. As a consequence, class number 2 is the one that gives the most misclassifications as it is neighboring the two other classes.

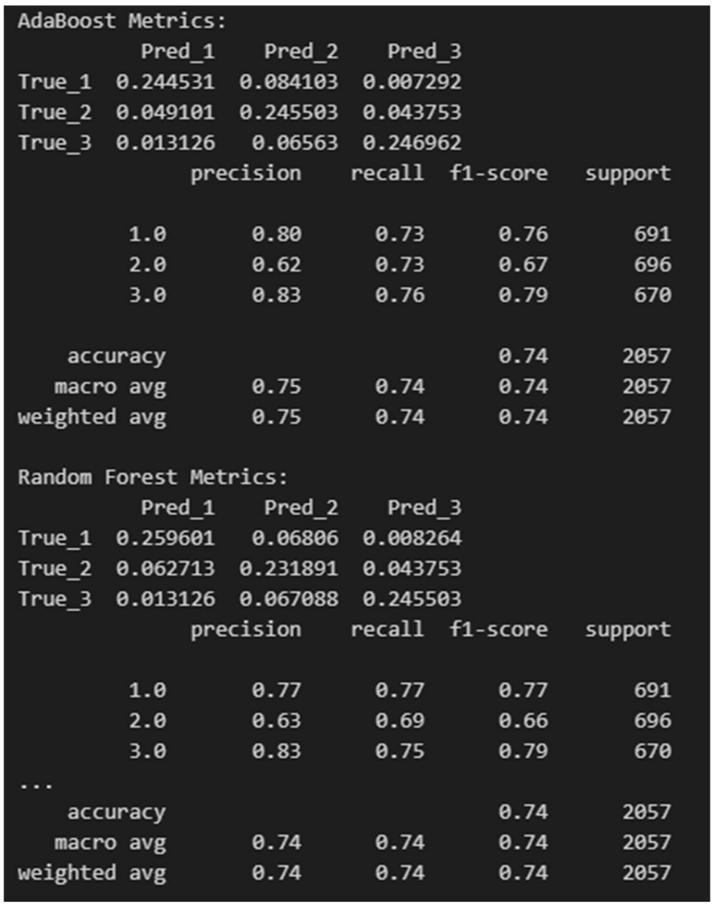


The second algorithm that was used is called Adaboost (Adaptive Boosting). It works by iteratively training weak classifiers on different subsets of the data and adjusting the weights of the misclassified data points. The final prediction is a weighted combination of the weak classifiers.

The AdaBoost algorithm starts by assigning equal weights to all the data points in the training set. Then, a weak classifier is trained on a subset of the data. The weak classifier is a simple model that is only slightly better than random guessing, such as a decision tree with only one split. After training the weak classifier, the algorithm updates the weights of the misclassified data points to give them higher importance in the next iteration.

The algorithm continues to iteratively train weak classifiers on different subsets of the data, with the weights of the misclassified data points being updated after each iteration. The final prediction is a weighted combination of all the weak classifiers, with the weights being proportional to their accuracy.

The main difference between AdaBoost and the random forest is that AdaBoost focuses on training weak classifiers and adjusting the weights of the misclassified data points, while random forest focuses on building multiple decision trees and combining their predictions. AdaBoost tends to perform well on small to medium-sized datasets with few features, while the random forest is more suitable for larger datasets with many features.

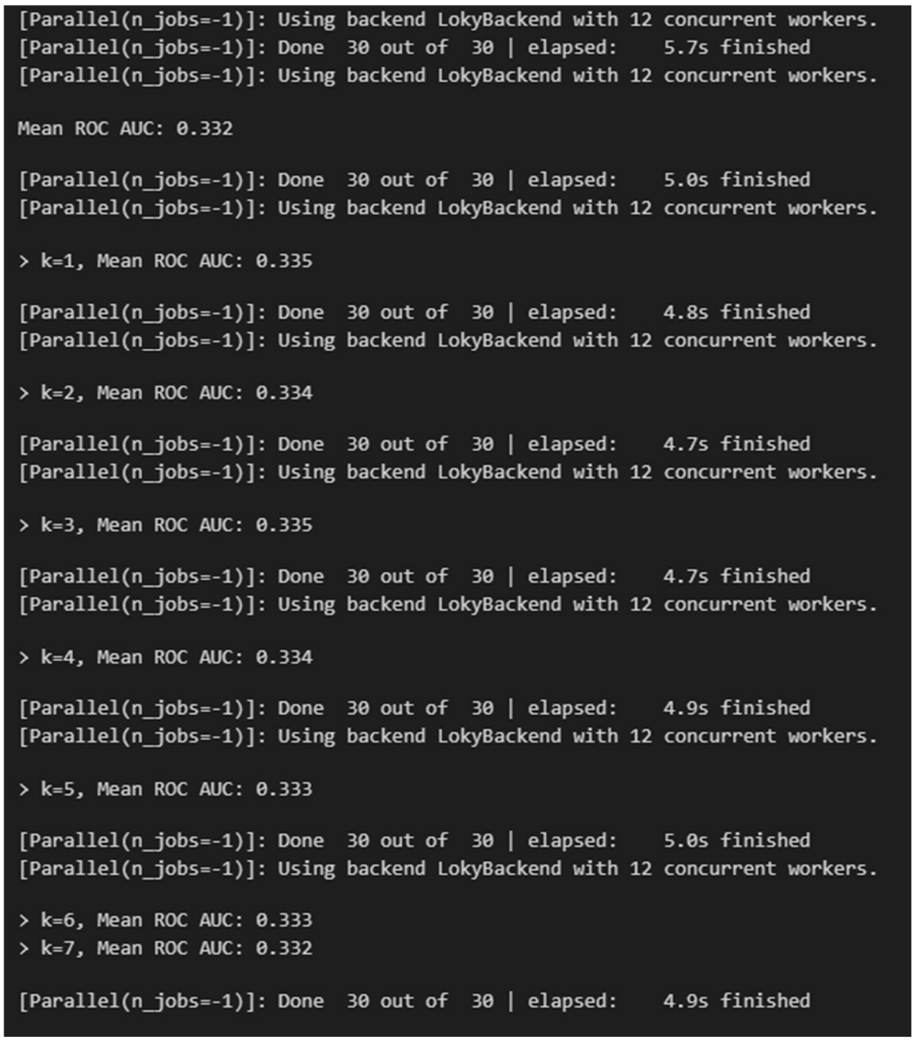


```
AdaBoost Metrics:
          Pred_1    Pred_2    Pred_3
True_1  0.244531  0.084103  0.007292
True_2  0.049101  0.245503  0.043753
True_3  0.013126   0.06563  0.246962
              precision    recall  f1-score   support

         1.0       0.80      0.73      0.76       691
         2.0       0.62      0.73      0.67       696
         3.0       0.83      0.76      0.79       670

    accuracy                           0.74      2057
   macro avg       0.75      0.74      0.74      2057
weighted avg       0.75      0.74      0.74      2057

Random Forest Metrics:
          Pred_1    Pred_2    Pred_3
True_1  0.259601   0.06806  0.008264
True_2  0.062713  0.231891  0.043753
True_3  0.013126  0.067088  0.245503
              precision    recall  f1-score   support

         1.0       0.77      0.77      0.77       691
         2.0       0.63      0.69      0.66       696
         3.0       0.83      0.75      0.79       670
...
    accuracy                           0.74      2057
   macro avg       0.74      0.74      0.74      2057
weighted avg       0.74      0.74      0.74      2057
```

The results of using AdaBoost are quite similar to those obtained by the Random Forest, and do not clearly allow to discriminate whether one method is best than the other. Overall, both methods give an accuracy value of 0.74.

Despite the fact that our target dataset was balanced in terms of the 3 classes that were predicted, we decided to use the SMOTE method for experimentation purposes. SMOTE method was used for undersampling and oversampling the target classes and the algorithm was launched by using default parameters.

ROC AUC metric was used to monitor the evolution of SMOTE in the function of the neighbor’s parameters. As shown in the next figure, SMOTE algorithm didn’t exhibit any major improvement in the performance of the classifier.



```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    5.7s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

Mean ROC AUC: 0.332

[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    5.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

> k=1, Mean ROC AUC: 0.335

[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    4.8s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

> k=2, Mean ROC AUC: 0.334

[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    4.7s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

> k=3, Mean ROC AUC: 0.335

[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    4.7s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

> k=4, Mean ROC AUC: 0.334

[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    4.9s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

> k=5, Mean ROC AUC: 0.333

[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    5.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

> k=6, Mean ROC AUC: 0.333
> k=7, Mean ROC AUC: 0.332

[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    4.9s finished
```

2. Deployment

No deployment of the model is considered to date, the code exists only in a github repository.

3. Results and recommendations

We have been able to analyze and determine features that were used in the regression of classification models that allowed us to predict our target average rating or its categorical derivative. The results obtained with both classical Linear Regression and Bayes Ridge Regression show an R^2 correlation of 0.72 and 0.02 mean squared error for both algorithms. We observe a lower correlation (0.5) and higher error (0.32) when trying to predict the categorical target. Considering classification, we only tested using the categorical target and we obtained an accuracy of 0.74 for both Random Forest and AdaBoost algorithms. SMOTE didn't seem to have a significant impact on the accuracy of predictions using AdaBoost, probably because the target was already sufficiently balanced.

Interpretation of coefficients of linear regression suggests a relevant weight of the average rating of the author of a book. This result seems understandable as one of the most common ways of searching for a good book is to take into account the overall quality of the writer. Feature selection analysis suggested that the average rating of the publisher could be an interesting feature account. As a perspective, we would continue to improve the methods by using the average rating of the publisher, the feature could be useful as it can be complicated to predict the rating of a book when an author is new and an average score is not possible to obtain.

In the case of classification algorithms, their explainability is more difficult but the results are in accordance with those obtained with regression algorithms. It would be interesting to see if the accuracy depends mostly on the average rating of the author as seen in the regression cases.