Machine Learning Project: Books Rating Prediction

Supervised by: Pr.Hannah Abi Akl

Project lead by: Salma Abdou, Yosr Noureddine, Jade Vieval, Nathan De Blecker

2023/2024

Looking into the Data Set

While trying to load the database, an error message was encountered. For this reason, we corrected the error manually in the csv files.

Exploratory Data Analysis

1. Exploration

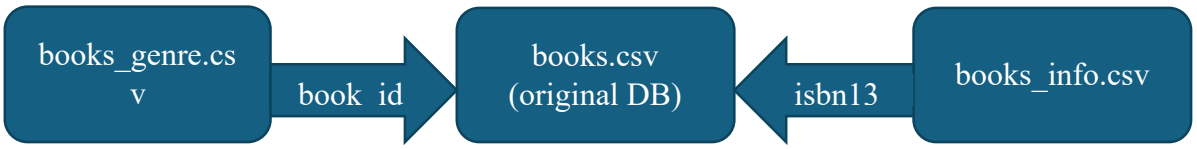
We started with a first attempt of featuring and training the model with the provided dataset, however the results were not reliable, and the model wasn't giving accurate predictions of ratings. After that, we thought of another way of solving the problem which is by enriching the database with more features coming from other databases downloaded on the web:

- **df_author**: average_rating, author_id, text_reviews_count, name, ratings_count
- **df_genre**: book_id, genres
- **df_info**: book_id, country_code, authors, isbn13

The previous databases were refined to keep only the crucial data for our model.
Moreover, we chose the drop the country_code column because US is the only value so its presence in our database is literally useless.

2. Joining the data sets

- In order to prevent having missing values in our Data sets we only used inner join.
- We joined books.csv with books_info.csv into final_df using the isbn13 instead of isbn because it contains more digits, so it was more reliable. Following this join, we have a new column: authors_y , which will enable us to do a join later with author_info.csv but before we need to correct it because it's a dictionary. It must be noted that the book_Id of books.csv and books_info.csv are not the same, that's why we haven't used this column for the join.
- Since the book_Id of books_genre.csv and books.csv are the same, we used this column to join the two data sets, we will have to feature the genre in books_genre.csv because it's not in a good shape for trainning.



=> At the end, our final database contains 9289 rows.

Important: Eliminating rows from the dataset is typically viewed as a less favorable method, however in this solution it was used with meticulous care and consideration.

First Solution: Feature Engineering and Training

1. Creating the weighted mean for authors

The authors_y column comprises dictionaries containing author IDs and their respective roles in the book. Our focus lies solely on extracting the author IDs to retrieve corresponding information from the author_info.csv file.

We've developed a script to parse each author ID, associate it with the data in author_info.csv, and calculate a compiled weighted mean for all authors contributing to each book. This weighted mean is computed based on both the average rating and the rating count attributed to each author.
We opted for a weighted mean approach because it accounts for the varying reliability of ratings, giving greater weight to authors with higher rating counts, which we deem indicative of greater rating reliability.

For some reason, our script couldn't run in our notebook, so we had to add arguments to it.

2. Vectorizing language_code

Each language code was transformed into a vector form to facilitate the assessment of their relationships with other features. The correlation matrix, however, indicated weak correlations for the language codes, which justified our decision to eliminate these specific columns.

3. Publication_date

We aim to investigate the potential impact of the publication date's components—year, month, and day—on the average rating. To facilitate this analysis, we have separated the publication date into three distinct columns for the year, month, and day. This restructuring allows us to generate a correlation matrix alongside the average rating.

4. Dropping some columns

In our dataset cleanup process, we eliminated columns that did not offer significant value to the model development. A case in point is the 'publisher' column, which had a vast array of unique values, making it unsuitable for effective classification or vectorization, leading to an inability to draw any robust insights.

5. Training with Linear Regression

Linear regression was the chosen method for training our model as this is one of simplest model, with a low complexity and enables an easy interpretation of features, yet the resultant high Root Mean Squared Error highlighted a significant precision shortfall in the model's outputs.

6. Training with Random Forest

We trained the model with random forest, as this method is known for its high accuracy and its robustness to outliers, however the score was slightly better with the linear regression.

=> Whether it is with Linear Regression or Random Forest, our root mean squared error scores were too high, so we need to think about another solution.

Second Solution: Feature Engineering, Refining and Training

1. Vectorizing the Genre column

The genre column represents a collection of genre groups, each of which we aim to vectorize. Our approach involves identifying the distinct genre groupings, yielding a set of 10 unique groups. Given their manageable quantity, we can proceed to vectorize these groups effectively.

2. Counting words in the book title, and the number of authors

We were wondering if the number of words in the book titles and/or the number of authors of a book had an impact on the average rating.

3. Correlation matrix

Some genres are well correlated to the average rating (comics_graphic, non-fiction), as well as the number of pages (with a score of 0.19*) but most importantly the author mean rate (the previously calculated weighted mean) with a score of 0.66, which is impressive.

*** NB: While a correlation coefficient of 0.19 may seem modest, it holds importance within the context of our dataset. Despite not being highly significant in isolation, it stands out as one of the strongest correlations observed among all variables. Therefore, dismissing it would overlook potentially valuable insights it could offer.**

4. Training with Linear Regression

We trained our new model with Linear Regression again.

We obtained a better score, with a lower Root Mean Squared Error of 0.188 instead of 0.196 with the previous model. We gain in precision and accuracy in predicting the average rating of a book.

5. Training with Random Forest

We obtained about the same Root Mean Squared Error as the Linear Regression above: 0.188.

6. Training with GBM XGBoost

GBM with XGBoost can capture complex relationships between variables and tends to have high predictive performance, but it may be more prone to overfitting than Random Forest.

This training doesn't improve our Root Mean Squared Error score, it stays consistent: about 0.188.

7. Training with Neural Network

We wanted to train our model with neural networks as it can also capture highly complex relationships between variables and can handle complex data. However this training has a negative impact on the Root Mean Squared Error score, it increases it: about 0.225.