Python Machine Learning Labs

Develop an end-to-end Machine Learning Pipeline

Book Rating Prediction Model

**Group 5**

Marius Akre

Vaishnavi Jagtap

Patrick Githendu

Stefanny Peraza

**Instructor**

Hanna Abi Akl

Book Rating Prediction Model

## Executive Summary

The following analysis aims to predict book ratings using different machine learning methods. A model of this type could help predict whether a book would be well or poorly rated by the reading community, based on certain characteristics, and in such a way the reader could focus on those that would have better ratings. Although it is not the objective of this analysis, the available data could also be used to make recommendations not only based on the estimated score, but rather based on the characteristics that are most important to the reader. However, it should be noted that the characteristics or variables available for this study are quite limited and correspond only to the books that have been evaluated in the database of Goodreads[[1]](#footnote-2).

## Data

The database contains 12 columns and 11,127 registers. However, of these 12 columns there are four (*bookID, isbn, isbn13, title*), which correspond to book identifiers and are therefore discarded as predictor variables. The *authors* and *publishers* are also not taken as variables on this occasion. Finally, the *average\_rating* variable is the one to be predicted, therefore only 5 possible predictors have been taken into account for this model i.e. language\_code, num\_pages, ratings\_count, text\_reviews\_count and publication\_date.

## Data cleaning

During data loading, problems were detected in the correct identification of the columns due to the presence of "," in the names of the authors, and the use of the "," at the same time as separator character of the fields in the file. To solve this problem, the "," in the authors field was replaced by a " -"; this was applied specifically to 4 records that were showing this problem, thus it was possible to retain them.

## Exploratory analysis and data processing

In the initial analysis of the data, the structure is checked to ensure that it is in the appropriate format for the analysis. In the case of the predictors "num\_pages", "ratings\_count" and "text\_reviews\_count" it is verified that they are of integer type (*int64*), "language\_code" is a categorical variable and therefore is set to "*object*", and in the case of "publication\_date" incorrect dates were adjusted and date format applied. The idea with this last variable is to analyze whether the year of publication shows any relationship with the scores given by readers. In the case of the variable to be predicted "average\_rating", its type is verified to be *float64*, although it is worth mentioning that it was also categorized, turning it into a dichotomous variable to be able to apply other machine learning (ML) techniques.

Once these changes were made, the distributions of the variables were analyzed, and additional adjustments made. For example, in "language\_code" we observe more than 10 categories, which were regrouped to later analyze if the language in which the book is published has any relationship with the score. Books in English were grouped into a single category, books in French, Spanish, German and Japanese were left untouched, but the rest with lower frequency of appearance in the database were regrouped under a single category of others[[2]](#footnote-3). At first glance, the box plots between the variable to be predicted and the categories show little difference between the grouping means, but the variable is retained for evaluation in the methods (Appendix 4).

In the different graphs, it can be observed that the variables are biased and present extreme values, as well as different magnitudes, which makes it necessary to apply some transformations so that the algorithms are not negatively affected. The graphs of the numerical variables can be seen in Appendix 3.

The first modification is to delimit the variables "num\_pages" and "text\_reviews\_count", applying an upper bound equivalent to three standard deviations, as a method to deal with extreme values. There are many different methodologies that can be implemented and Standard Deviation Method is one of them. However, this is not enough, and therefore, in order to have variables with slightly more normal distributions and that work better in the regression and classification methods, the *RobustScaler* method is applied, with the parameter *quantile\_range=(25.0, 75.0)* which seeks to reduce the effect of the tails of the distributions in the transformation, centering and scaling the variables based on the defined interquartile range. Looking for more normality in the variables, PowerTransformer(method="yeo-johnson") was also applied[[3]](#footnote-4). In the case of categorical variables, *OneHotEncoder* method implemented in the preprocessing class of the *sklearn* library was applied.

On the other hand, the correlation between the variables is also reviewed (Appendix 5), and it is observed that there are two of them highly correlated "ratings\_count" and "text\_reviews\_count". Even though in some methods it does not necessarily affect that the predictors are correlated, in others it can be very problematic, therefore it is decided to eliminate one of the variables. The "text\_reviews\_count" is kept since it shows lower extreme values, discarding what seems to be redundant information. In this correlation analysis, it can also be observed that there is no relationship, at least linear, between the variable to be predicted and the numerical predictors, and this is a first sign that it will be difficult to obtain an adequate model to predict the "average\_rating". No other variable preselection is applied, given the small number of them available for this model. Instead, after performing some model runs, at least two new variables were created: the number of books written by the authors ("author\_number\_books") and the average of the "average\_rating" for the author ("author\_books\_avg\_rating"), both cases using the information available in the database[[4]](#footnote-5).

## Methodology

For model calibration, the samples are initially separated into training and test samples, in a ratio of 80%/20% respectively. As mentioned above, the objective is to perform two exercises, one for the prediction of the "average\_rating" in its numerical form and the other for the variable categorized.

In the case of both regression and classification methods, the aim is to create an initial baseline scenario to compare the performance of the methods and thus choose the one that shows the best results in order to subsequently adjust the hyperparameters and try to improve the final model. This comparison is made from a cross validation[[5]](#footnote-6) with the *cross\_val\_score* method implemented in the *sklearn* library, using 5 splits and shuffle the data before splitting into batches (for the cross validation in the classification case, the *RepeatedStratifiedKFold* method is used, which in addition to splits, allows repeating the process, in this case 5 repetitions were used with 5 splits each). In regression, the coefficient of determination of the prediction is used to compare between methods, and in the classification case, the area under the ROC curve.

Thus, in order to predict the "average\_rating" (continuous variable), the following linear regression algorithms were initially tested: Ordinary least squares Linear Regression (LinR), k-nearest neighbors regressor (KNN), Linear least squares with l2 regularization (RGE) y Linear Model trained with L1 prior as regularizer (aka the Lasso-- LSO), with different Alpha values in both cases, the Linear Support Vector Regression (LSVR), random forest regressor (RFReg) and even transformed data with a *PolynomialFeatures* method was used for applying a LinR on these new variables (polyLR). The documentation on these methods is extensive so no details will be provided in this report, all methods can be consulted in the *sklearn* documentation. The results of the summarized runs are presented in Appendix 6, where it is observed that although the *RFReg* shows better results these do not even reach 10%. The regression models show practically no predictive capacity when using the available variables confirming what was observed during the Exploratory and Data Analysis.

Given the above, we proceeded with the exercise of trying to predict the "average\_rating" but dichotomized, where 4 was chosen as the threshold point since it practically corresponds to the median of the distribution (HigRat = 42.5%, LMRat = 57.4%). So that, those books with a score higher than this value were classified as high rating and the rest as low rating. The new objective will be to establish whether, based on the available characteristics, it is possible to estimate whether a book will be ranked with a high score or not. For this exercise, the following methods were applied: Logistic Regression (LR with solver="lbfgs"), Linear Discriminant Analysis (LDA), the k-nearest neighbors vote (KNN), Random Forest Classifier (RF), Decision Tree Classifier (CART), Gaussian Naive Bayes (NB), and C-Support Vector Classification (SVM). The RF and SVM methos show the better results (Appendix 6), therefore, these are selected to perform the hyperparameter adjustment and see if it is possible to achieve better results.

## Results

Given the set of hyperparameters to be estimated, both methods yield very similar results. It's crucial to emphasize that the estimation was done for the probability of belonging to each category rather than directly predicting the category itself. Both of these methods offer such options, enabling the selection of a cut-off point based on the chosen indicator by the analyst.. The *fbeta\_score* method from *sklearn.metrics* was used for this purpose, however in the end the cut-off point that does not seem to sacrifice so much f1-score or precision per category is one closer to the original proportion between the categories and not necessarily the one suggested by the method, in any case, the definition of the cut-off point will depend on the objectives of the business case. The following table summarizes the methods with the best parameters found by *RandomizedSearchCV*, given the grid of parameters provided, the results for the best estimator in the cross-validations, the cut-off point chosen to create the categories (from the estimated probabilities), and the area under the curve given the established cut-off point.



In the Appendix 7 the classification report is shown. The observed values, hovering around 0.6, indicate that there is much room for improvement in the model. Finally, the "author\_books\_avg\_rating" variable was added to the dataset and the RF model was run again. As explained before, this variable is based on the variable to be predicted "average\_rating", and as expected the results for precision, accuracy and f1-score, all increased to a value close to 0.9 (Appendix 8). This is a variable that at first glance has a circular relationship with the value to be predicted, especially for those writers who only have one book, however the results are added as an academic exercise, and could be an interesting aspect to be considered in those models where an adjustment can be made each time a new data is entered, machine learning in real time and maybe not in cases where the models are retrained and deployed with a lower periodicity.

It is worth mentioning that for the final implementation, the random forest method was selected, which is hosted in a git repository (daily build runs automatically as well through a Continuous Integration) and a simple deployment exercise was performed on a web site (more detail in Appendix 9).

Appendix

Appendix 1 Number of records per language



Appendix 2 Descriptive statistics, numeric variables



Appendix 3 Distributions and Box-plot of numerical variables

|  |  |
| --- | --- |
| A group of graphs showing different sizes of numbers  Description automatically generated with medium confidence | A screenshot of a graph  Description automatically generated |

Appendix 4 Box-plot of average rating by language code grouped

A graph with blue and gray squares

Description automatically generated with medium confidence

Appendix 5 Correlation matrix

A graph of a number of books

Description automatically generated with medium confidence

Appendix 6 Algorithm comparison

|  |  |
| --- | --- |
| Regression algorithms | Classification algorithms |

Appendix 7 Classification report and confusion matrix



|  |  |
| --- | --- |
| SVM | RF |

Appendix 8 Classification report adding "author\_books\_avg\_rating" variable



Appendix 9 Hosting, Building and Deployment

GitHub repository for hosting and building: Continuous Integration (CI) with Artifacts produced.  
  
<https://github.com/ma23dsti/PythonMLProj>

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Wep App

<http://127.0.0.1:8000/>

A screenshot of a computer

Description automatically generated

|  |
| --- |
| A screenshot of a computer  Description automatically generated |
| A screenshot of a book rating  Description automatically generated |

1. Goodreads (Jan, 2007)\_. the world’s largest site for readers and book recommendations.. https://www.goodreads.com/ [↑](#footnote-ref-2)
2. Appendix 1 for further details. [↑](#footnote-ref-3)
3. The optimal parameter for stabilizing variance and minimizing skewness is estimated through maximum likelihood. [↑](#footnote-ref-4)
4. When used, the same transformations were applied to them as to the numerical variables. [↑](#footnote-ref-5)
5. On training sample. [↑](#footnote-ref-6)