Data ScienceTech Institute

****

Machine learning with Python Labs

****

TECHNICAL REPORT ON MACHINE LEARNING BOOK REVIEWS PROJECT

Written by:

Jean-Luc Boa Thiemele (DS A22)

Deogratias Allakonon (DS A22)

Clement Adebisi (DE A22)

Submitted to (Professor):

Hanna Abi Hakil

Academic year 2022 - 2023

CONTEXT

The objective of this machine-learning project is to develop a predictive model that can accurately classify books into classes of ratings. The dataset used contains 11123 rows and 12 columns of features such as authors, number of pages, ratings count, and text reviews count etc. The variable to predict is the average rating which is a continuous variable. Our challenge will be to analyze the data and create proper classes to make our classification. Prior to analysis, the dataset underwent processing steps including handling missing values, outliers, adding of new features etc. The goal here is to implement a model that will give an idea on the rating of a book based on a set of features. Having those predictions could help readers in the choice of books to read or buy.

I. EXPLORATORY DATA ANALYSIS AND PREPROCESSING

During the exploration, the main information we have withdrawn from each feature are discussed as follows.

Language code: There are 27 different languages in the dataset and the most occurring is English.

- We regrouped ‘en-US’,’en-GB’ with ‘eng’ because it does not mean much to an English reader.
- 95.4% of the book in the dataset are written in English.

Title: When focusing on title column, we remark that there is 7% of doubloons (when we do not count the first occurrence as a doubloon.) But taking the other columns into account make each row stand for itself as they provide different information about the book. The unique identifier of the book ‘ISBN’ changes at each row and the information in the other columns changes most of the time. It can be considered as an update or a versioning of books. The following analysis is led while considering only the title columns.

- 93% of the titles are unique and the uniqueness of the title might limit the generalization ability of the model due to overfitting. If the model encounters a new title during prediction that was not in the training data, it might struggle to make accurate prediction.
- The title of books is written in their language code. The inconsistency in title formatting poses challenges when extracting information using NLP techniques like tokenization, word embedding etc.

Processing: For all these reasons, we will not include the title column in our final set of features.

Authors: Stephen King, with 40 books wrote the most books among the 6639 unique authors.

- Group of authors are counted as one author when counting for the number of books.
- One Hot Encoding will not be possible because of the huge number of unique authors.
- Isolating authors (by exploding the author’s columns) will lead to repeated rows which is not great for training.

Processing: To make better sense of the author column, we are going to add the average ratings of authors as a new variable in our dataset. This new feature is found in a json file ‘goodreads_book_authors.json’ that contains the average ratings of authors that has been rated on goodreads and we will use this rating to design a new rating based on individual authors that took part in writing a book. We made a lighter version by only saving the name of the authors and its average rating in a dictionary (105Mo to 19.8Mo), this is what we will be using for the project. To be clear we will not use the ratings obtained directly in the books.csv file.

Publisher: Vintage is the most occurring publisher with 318 appearances among 2290 unique publishers.

- Group of publishers are counted as one publisher in the books.csv dataset.
- Variations in the name of publishers (one publisher having their names written in different forms) make it difficult to regroup them.
- Like authors columns, One Hot Encoding will not be possible.
- Isolating publisher (by exploding the publisher's columns) will lead to repeated rows which is not great for training.

Processing: *We will drop the publisher column.*

ISBN10, ISBN13: From those elements we could have got a lot of information but unfortunately, they are not in the right format.

For example: 978-0-545-01022-1

'978' indicates that this is a book -- '0' represent the country or the language area (here it is English) --'545' is the publisher identifier -- '01022" is the publication element (title) -- 1 is the check digit.

The problem is that the publisher identifier and the publication element can have different lengths. So having ISBN13 written like 9780545010221 makes it difficult to draw information confidently. For example, publisher identifier could be 5450 and title element could be 1022 we could never be sure.

Processing: We will drop these columns.

Publication_date : The most occuring date is 10/01/2005 and there are 3679 unique values. We will extract the months and the years and use it for the training.

Num_pages:

- 2.63% of the rows have either 0 pages or a number of pages >1000 (maximum is 6576)

  Processing: The rows with zero page (76 rows) will be replaced by the mean value. Rows with number of pages superior to 1000 (217 rows) will be suppressed from the database as they are found to be boxset or volumes of books which is no longer a single book (bearing in mind that our goal is to get rating for single books). We will keep this variable in our feature set

Average ratings: The scores range from 0-5 with very few records between 0 and 2.

Rating_count, text_reviews_count and average_ratings:

- We cannot have cases where the rating count is 0 and the average_rating is different from 0.
- The linear correlation of Rating_count, text_reviews_count with average_ratings are respectively 0.04 and 0.03.
- The linear correlation between Rating_count and text_reviews_count is 86%

Processing & features engineering: We dropped rows where ratings_count is 0 and average_rating is greater than 0 because it is impossible for a book to have an average rating when there are no records of how many different ratings it has got. Given that there is a strong relation between rating_count and text_reviews count and a weak correlation between average_rating count, we engineered a new feature with rating_count and text_reviews which is the ratio of the text_reviews to the rating_count. After computing this metric, we have a correlation of -14% with average_rating. We will keep that variable and suppress rating_count and reviews count.

Scatterplot interpretation: When scatterplotting the features with average_ratings we saw that there is concentration of points on certain values. Performing a regression analysis on this data can be challenging because the model might struggle to capture meaningful patterns. This is the reason we transformed the regression problem into a classification one. We created the variable average_ratings_category for the classification.

II. DATA ANALYSIS

The results of the data analysis part are fully explained in the data analysis notebooks (data_analysis. ipynb). Those are the keys point:

- We confirm that Stephen king is the man who wrote the most book (95 books).
- We determined the 10 most rated and 5 bottom rated authors.
- We determined correlation between the features and average_rating
- The language is mainly distributed into the English code.
- The Illiad is the book with the most appearances.
- Twilight is the most rated book

III. MODEL SELECTION

-XGBOOST: XGBoost is an implementation of gradient boosting, which iteratively improves the model's performance by focusing on the weaknesses of previous iterations. This makes it particularly effective for improving the model's predictive accuracy. Our dataset contains features that may exhibit complex relationships and non-linearity. XGBoost's ability to model complex interactions and patterns in the data makes it a strong contender. Like random forest, it provides a mechanism for assessing feature importance, allowing us to identify the most relevant features contributing to predictions. This feature helps with model interpretability. It also prevents overfitting which is essential for maintaining model generalization and avoiding fitting noise in the data. This is also a model that has proven itself by allowing teams to win kaggle competitions and this makes it well-suited for large datasets like ours.

-RANDOM FOREST: The dataset size (about 12,000 rows) is sufficient for Random Forest to build a diverse ensemble of decision trees, leading to better generalization. It is also robust against overfitting, which is crucial to prevent the model from memorizing the noise in the data and finally the ensemble nature of Random Forest reduces variance and provides a feature importance measure, aiding in model interpretation.

DECISION TREE: We also going to use a decision tree because it makes minimal assumptions about the data distribution or relationships between features. It can handle non-linearity in data, give insight into the most important features for making predictions and it is easy to understand.

IV. MODEL TRAINING AND EVALUATION

First, we started by creating a category from the average rating features, we called it *average_rating_category*. The classes are as shown in the table below.

Classes/Categories	**0**	**1**	**2**	**3**
Ratings	0-2	2-3	3-4	4-5
Number of elements in the class	7	54	4994	3643

The dataset was then split into training and testing sets using a common practice of 80/20 ratio with a parameter ‘stratify’ which allows to split with respect to the composition of the initial dataset. 80% of the data was used for training the model while the remaining 20% was reserved for evaluating the model’s performance. The evaluation of the model was performed using classification report of scikit-learn which compute the **precision**, the **recall** and **F1 score.**

Addressing Imbalance in the Train Set: The introduction of the new variable, average_ratings_category, has revealed an imbalance issue within the dataset. This imbalance occurs due to differing class frequencies. To overcome this challenge, we have implemented a strategy centered around Synthetic Minority Over-sampling Technique (SMOTE).

SMOTE Implementation: The SMOTE algorithm plays an important role in mitigating the class imbalance. By generating synthetic data points for classes with fewer instances than the majority class, SMOTE ensures that all classes are represented evenly. As a result, each class will eventually contain an equal number of instances, specifically 4994 in this case. Through this technique, we aim to create a more balanced training set.

V. RESUTS AND DISCUSSION

The result represents the training and evaluation outcomes for three different classifiers: DecisionTreeClassifier, XGBClassifier, and RandomForestClassifier. Let us analyze the result and discuss its implications:

Training Results (see results in runs/exp1):
The training phase reports various metrics such as accuracy, precision (weighted), recall (weighted), and F1-score (weighted) for each classifier. These metrics help us understand how well the models have learned from the training data. The performances of the models are as depicted in the table below.

	Accuracy	**Precision**	**F1-Score**	**Recall**
DecisionTreeClassifier	84.94%	84.78%	84.81%	84.94%
XGBClassifier	88.14%	88.07%	88.06%	88.14%
RandomForestClassifier	89.31%	89.31%	89.26%	89.31%

Evaluation Results (see results.txt in project_report folder): The evaluation phase assesses the classifiers performance on the test data. For all classifiers, class label '0' has very low precision, indicating that the classifier struggles to predict this class correctly. We can attribute this to insufficient instances of this class in the test set. We could argue same for class label '1' as it also has relatively low precision across all classifiers. Class labels '2' and '3' generally have higher precision, recall, and F1-scores, indicating better performance on these classes. The model is more robust with these classes because they are found abundantly in the training set. The macro avg and weighted avg metrics provide overall performance metrics for the models. These averages account for class imbalance and provide a holistic view of the models' capabilities.

VI. MODEL SELECTION

XGBClassifier and RandomForestClassifier outperform DecisionTreeClassifier in terms of both training and evaluation metrics. We could make this inference because the performance on the test set indicate that XGBClassifier and RandomForestClassifier yield better results.

RandomForestClassifier generally gives a better result across the three evaluation metrics (75%) than XGBClassifier.

VII. GRID SEARCH

Can XGBClassifier be better than RandomForestClassifier after a grid search?

These are the parameter we are going to look into:

[{'n_estimators': [100], 'max_depth': [20, 50], 'learning_rate': [0.04, 0.05, 0.08]}]

The best possible combination of parameter is [{'n_estimators': [100], 'max_depth': [20], 'learning_rate': [0.8]}].

The prediction of the model on the test set is as follow:

	precision	recall	f1-score	Support
0	0	0	0	2
1	0.21	0.21	0.21	14
2	0.77	0.77	0.77	1247
3	0.70	0.70	0.70	915
Accuracy	-	-	0.74	2178
Macro Avg	0.42	0.42	0.42	2178
weighted avg	0.74	0.74	0.74	2178

- The results with the best parameters after a grid search show similar patterns when considering the accuracy. We have a decrease of 1 % in the precision and F1-score indicating that the grid search does not improve the performance of our models.
- As we saw above, the low performance on class 0 and 1 is due to a lack of informative features and the imbalance in the dataset.

In order to get better result, we can consider further hyper parameter tuning or feature engineering to improve the model's performance, especially for class 0 and 1.

Overall, while the accuracy and F1-scores are decent, there is still room for improvement, particularly for class 0 and 1. For this grid search experiment, XGBClassifier does not perform better than our random forest when comparing both models on test set.