PROJECT BOOKS

INTRODUCTION

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 in order to make our classification. Prior to analysis, the dataset underwent processing steps including handling missing values, outliers, adding of new feature 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, will help readers in the choice of their next book.

EXPLORATORY DATA ANALYSIS AND PREPROCESSING

During the exploration, the main information we have withdrawn from each features are the following

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

* We regroup ‘en-US’,’en-GB’ with ‘eng’ as it is still English
* 95.4% of the book in the dataset are written in English

**Title:** When focusing on title columns, we remark that there is 7% of doubloons (when we do not count the first occurrence as a doubloons.) but taking the other columns into account make each rows 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 title are unique and 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 pose challenges when extracting information using NLP techniques like tokenization, word embedding etc

**Processing**: For all those reasons, we will not include title in our final set of feature.

**Authors:** Stephan King wrote the most books with 40 appearances and there is 6639 unique authors.

* Group of authors are counted as one author
* 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 of that column, we are going to add the average ratings of authors as a new variable. The Goodread website has a json file ‘goodreads\_book\_authors.json’ that contains the average ratings of authors and we will use it to add this information.

**Publisher:** Vintage is the most occurring publisher with 318 appearances. There is 2290 unique publisher.

* Group of publisher are counted as one publisher
* Variations in the name writing of publishers 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 columns.

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

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

‘978’ indicate 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 length. 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 cannot be sure.

**Processing**: We will drop those columns.

**Publication\_date** : The most occuring date is 10/01/2005 and there is 3679 unique values. We will extract the month and the year from the date and use them during our data analysis part.

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 replace by the median as the distribution is left skew. Rows with number of pages superior to 1000 (217 rows) will be suppress from the database as there are boxset, volumes etc. We will keep that variable in our feature set

average\_ratings: the scores ranges from 0-5. There is little data between 0-2

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 iss 86%

Processing & features engineering: We well drop rows where ratings\_count is 0 and average\_rating different from 0. Given that there is a strong relation between rating\_count and text\_reviews count and a weak correlation between average\_rating count , we are going to ingenieer a new feature with rating\_count and text\_reviews which is the ratio of the text\_reviews over the rating\_count. After computing this metric we will have a correlation of 15% 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 a concentration of point 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 transform the regression problem into a classification. We created the variable average\_ratings\_category for the classification

DATA ANALYSIS

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

* We confirm that Stephan king is the man who wrote the most book.
* We determine the 10 highly rated and 5 bottom rated authors
* We determine correlation between the features and average\_rating
* The language are mainly distributed into the English code.

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 prevent overfitting which is essential for maintaining model generalization and avoiding fitting noise in the data. This is also a model who have proven itself by allowing team to win kaggle competition and this makes it well-suited for large datasets like ours.

-RANDOM FOREST : The dataset size (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.

MODEL TRAINING AND EVALUATION

First, we started by creating category from the average rating features, we called it average\_ratning\_category. The classes are as follow: 0:0-2, 1:2-3, 3:3-4 and 4:4-5.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 of 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 were perform using classification report of scikit-learn which compute the precision, the recall and f1 score.

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:

- 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.

- DecisionTreeClassifier achieved the lowest performance across all metrics, with an accuracy of 84.33%, precision of 84.27%, recall of 84.34%, and F1-score of 84.15% on the training data.

- XGBClassifier and RandomForestClassifier demonstrated higher training performance compared to DecisionTreeClassifier. XGBClassifier achieved an accuracy of 88.99%, precision of 89.13%, recall of 88.99%, and F1-score of 88.94%. RandomForestClassifier achieved an accuracy of 89.31%, precision of 89.31%, recall of 89.35%, and F1-score of 89.19%.

Evaluation Results:

- 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. This is due to insufficient instances of this class in the test set.

- Class label '1' also has relatively low precision across all classifiers, suggesting that they struggle to correctly predict this class as well for the same reason as for the class ’0’.

- Class labels '2' and '3' generally have higher precision, recall, and F1-scores, indicating better performance on these classes.

- 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.

Model Selection

- XGBClassifier and RandomForestClassifier outperform DecisionTreeClassifier in terms of both training and evaluation metrics.

- The models' performances on the test set indicate that XGBClassifier and RandomForestClassifier yield better results, with XGBClassifier slightly ahead in terms of accuracy, precision, recall, and F1-score.

GRID SEARCH

Given that XGBClassifier is the better one without tuning hyperparametre. We are going to apply a grid search on it in order to obtain the best parameter possible.