BOOK RATING PREDICTION MODEL

PYTHON Machine Learning LAB

A22 Cohort

Performed by:
C. Mpaga - J. Huang -J. Sanchez - C. Nwabo

Data ScienceTech Institute

Table of Contents

Executive Summary

In this application, we were instructed to predict average books rating from historical data about books average ratings and associated features. After data preprocessing and processing, we explore and model to deliver a model and it performance. Our results promote the generalized boosting machine model (gbm) in a regression setting to solve this problem.

Topic Introduction

Books ratings are important for authors, publishers, readers and books lovers. Very often, some of them need to check any book ratings before purchasing it. Sometimes, an author may also want to foresee how his book could be perceived or be rated. Machine learning could help to tackle this situation and deliver an automated solution. As an application, we are willing to create a machine learning solution to predict a given book's average rating.

Method

Repository URL
https://github.com/ChristopheMpagaDSTI/books_rating_project_python_main

The first task here was to load and inspect data. In this application, data are in csv format, with 11,127 lines and 12 variables. The variables are combined with numerical and categorical features. There was no data acquisition required, therefore, what we had to do is simply downloading the data, then we do inspect, preprocess and process on them. After that, we perform exploratory data analysis. Following by modeling our data and assessing the model performance from the final best model.

1. Data inspection

We started by reading the data and doing exploratory data analysis. The first problem we faced here was importing problems. There were 4 lines which did not respect the format of CSV and further led to the parsing error from pandas `read_csv` function. In this case, only 4 lines required manually modified and the importing process was succeeded afterwards.
We encountered some errors from variables names and uniform formatting in the same feature. The variable “num_pages”, was written with a leading blank space and the variable publication had some bad formatted dates and identation issues.

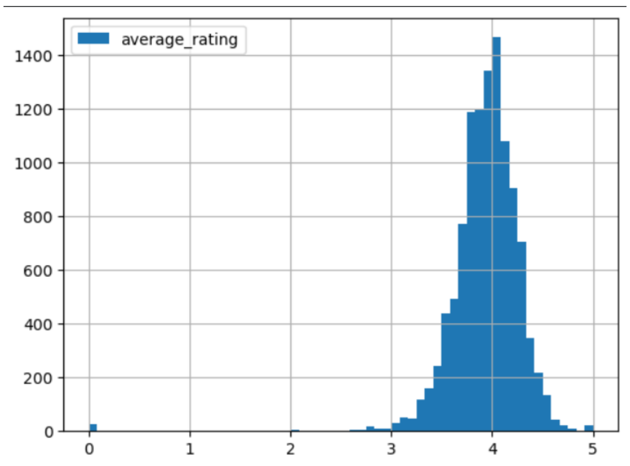
2.. Data preprocessing

Concerning the variable “num_pages”, we just renamed it by getting rid of the leading blank spaces in the variable name. Then we corrected bad dates format into “ddmmyyyy” format. (day/month/year).

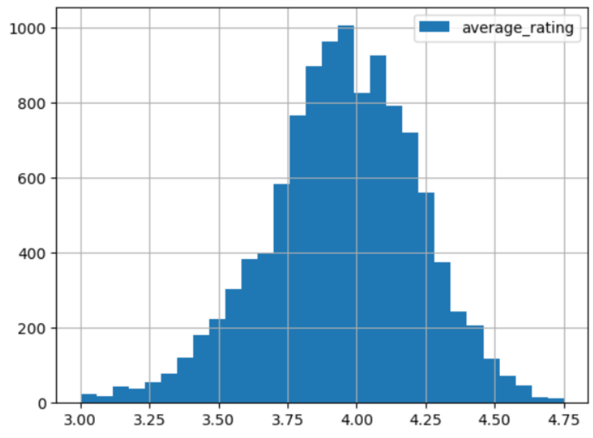
3.Exploratory Data Analysis (EDA)

In this part, we aimed to investigate variables distribution in one hand, then investigate the existence of any relation with target variable. We usually associated to a plot, when this was the case, a numerical summary. We performed two types of EDA, one dimensional EDA (1D EDA) and two dimensional EDA (2D EDA). In 1D EDA, we plotted distribution and associated some numerical summaries, like mean, median, standard deviation, frequency and inter quartile range, when the latter was needed. Most of the time variables are not normally distributed, therefore we used the median instead of the mean as central tendency statistics to describe the distribution in along with 1st and 3rd quantile. We would like to state that, target variable was not normally distributed but, we managed to remove outliers to get close to a normal distribution.

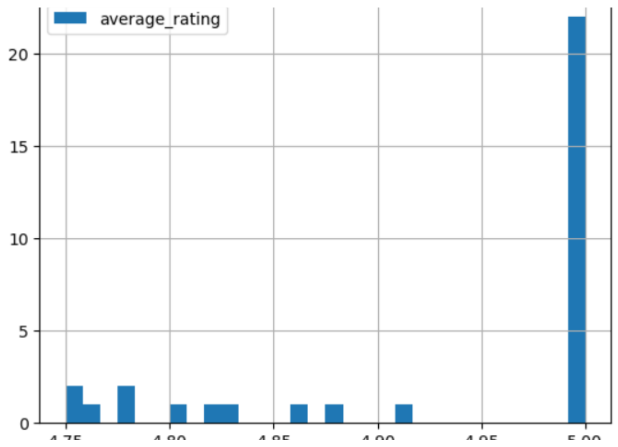
histogram of average ratings)



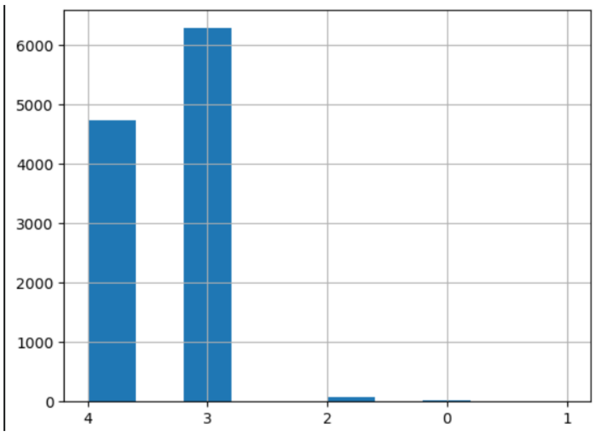
histogram of average ratings :range (0,4.75)



histogram of average ratings :range (4.75,5)



Discretize average_rating



4. Modeling

Here, we started by trying and testing our data on some common models like linear regression, regularized linear regression (lasso, ridge), tree-based models (random forest, gbm) and support vectors machines. At the end of this process, we promoted two applications: regression and classification. In regression, we keep Lasso and gbm models.

5. Model validation

At the end of modeling process, we chose for regression application lasso and generalized boosting machine. We then performed hyper-parameters search. This was done with a random search strategy. We ended-up with gbm as the good candidate over lasso, with 10 cv subsets. Following, CV, Lasso has a rmse of 0.12 while gbm has a 0.08 result. We promoted this model after getting it cross-validated, with ten (10) subsets.

6.Predictions

We perform a prediction on test set, a subset of 4440 rows and 4 variables, including target variable. We got a rmse of 0.024, which is an improvement.

Results

Gbm model returned a score of 0.021 at the validation stage and 0.24 at test stage.

Discussions

In this application we intend to realize an application via machine learning to predict books average ratings. This was a supervised machine learning application, average-rating as the target variable. We end-up a gbm model as our promoted model. This model shows a performance of 0.02 (rmse) at performance step. In this application it outperformed linear and other method like svm. This is inline with our hypothesis, stating that a nonlinear model would be a better choice for this application. However, we recommend further fine tuning of gbm via hyper-params search. This, in order to improve this work. A large grid of parameters or change of search strategy to a grid search could be good as an alternate way to improving this work.

Conclusion

Following, hyper-parameters search, gbm shows a far better rmse of 0.021 and lasso a rmse of 0.3.
GBM performance improved with hyper-parameters search, while lasso did not. Therefore for this application and in a regression setting, we would recommend gbm as final model.

In test step gbm yields 0.24, which is almost similar to the validation rmse. However, we would recommend further, hyper-parameters search and investigation to fine tune this model again with a large grid or grid strategy instead.

This work could be improved by considering a deep transformation of target or standardized numerical features. Since the average rating in book.csv dataset is not normally distributed, the biased average rating distribution led to the result of high-accuracy within the certain interval. In future exploration, we might have a chance to improve the prediction result by adding up potential data or features. The second possibility would be lying in the ISBN-13 feature where interesting information is provided such as the publisher, language published in, issuing agency of the publication code, etc. Through analyzing, we might be able to increase the span of the data frame.

About machine learning
https://www.kaggle.com/code/saurabhwadhawan/prediction-of-average-book-ratings-regression#Exploratory-Analysis-on-the-Dataset
https://towardsdatascience.com/random-forest-regression-5f605132d19d

About docker image
https://docs.docker.com/engine/reference/commandline/compose/
https://neptune.ai/blog/data-science-machine-learning-in-containers
https://github.com/adaltas/dsti-mlops-2023-spring/tree/main/09.docker-containers
https://towardsdatascience.com/dockerizing-jupyter-projects-39aad547484a

Categorical Approach Summary

Categorical Classification:

In this section we explore the possibility that using categorical targets may provide another viable method of books ratings predictions. We experiment with classifying the books with the following labels using logistic regression and random forest classification:

.10 = Targets ranging from 0.0 to 5.0 in .1 increments. This was attempted as it was close to the original problem statement, and it has been noted that, if an interested party is searching for a book by rating, a 3.2 will be listed before a 3.1 and might be more likely to be viewed by the searcher.

.25 = Targets ranging from 0.0 to 5.0 in .25 increments.
.50 = Targets ranging from 0.0 to 5.0 in .50 increments. These targets more closely resemble the way a human might summarize a rating system such as "I'd give that book 3.5 stars".

Whole numbers= This is the system that mirrors the choice available to Goodreads users as users are only able to rate books with complete stars starting with one star for the worst rating. This is also why later we experiment with dropping books from the dataset that have not received a rating.

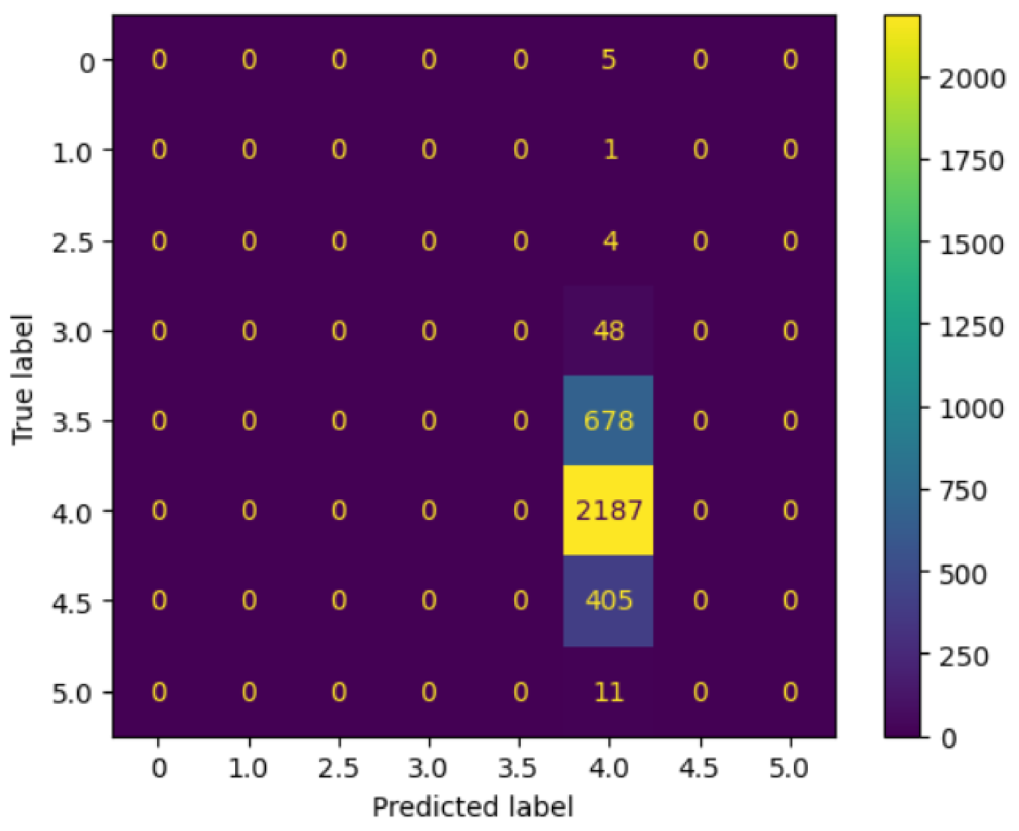
Custom Buckets = We experimented with interpreting the data distribution into custom buckets, roughly using standard deviation as a guide.

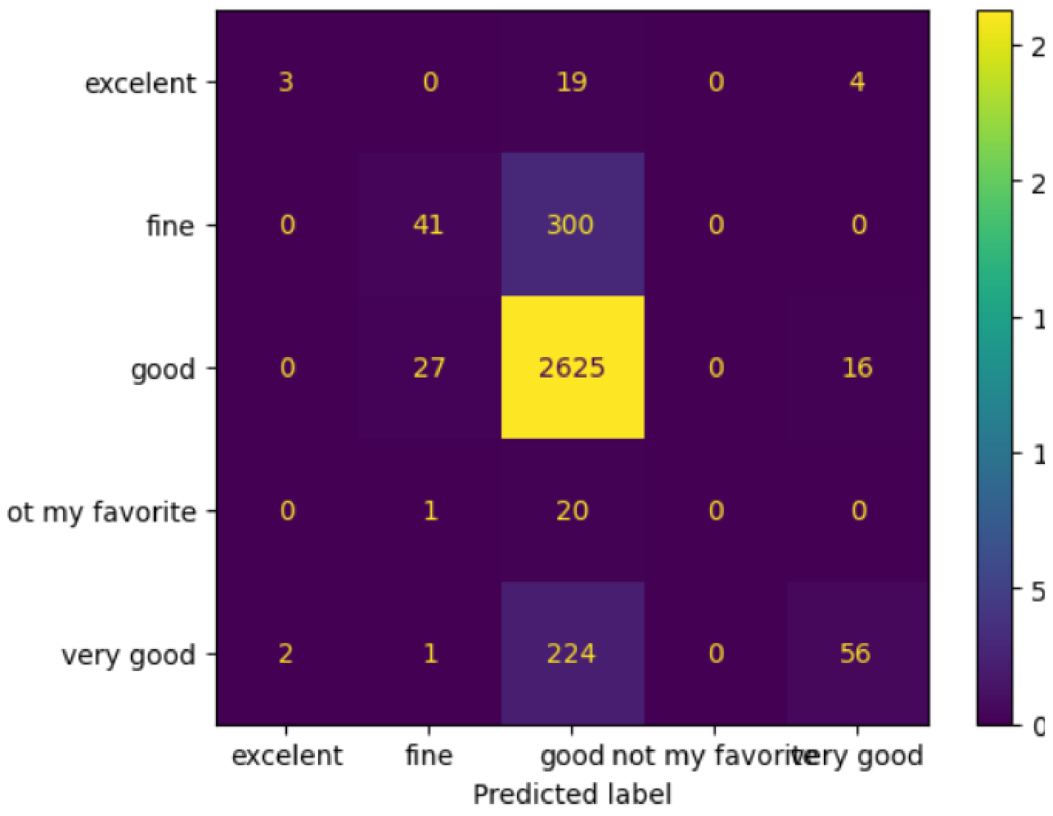
Not My Favorite: All books rated below 3.0
Fine: Books falling between 3.6 and 4.3
Very Good: Books falling between 4.3 and 4.6
Excellent: Books rated above 4.6

Throughout the experiments, we found that F1 and accuracy scores improved as the buckets got smaller. This solution proved to be less interesting as it was found that, as the buckets got smaller, the machine gained accuracy by simply predicting the majority class.

While the experiments in turning the prediction problem into a classification problem did not yield the best results, it was discovered that, in every case, adding a transformed authors column to the features improved both accuracy and F1 scores by anywhere from 2%-11%

Confusion Matrix Display


True label
Predicted label


excelent
fine
good
ot my favorite
very good
excelent
fine
good
not my favorite
very good
Predicted label