Project: Book Rating Prediction Model

Course: Python Machine Learning Labs

DSTI

Omar AIT ALI

Project Summary:

The dataset provided is a curation of Goodreads books based on real user information. It can be used for many tasks like predicting a book's rating or recommending new books.

Description the columns of this Dataset:

1) bookID: A unique identification number for each book.

2) title: The name under which the book was published.

3) authors: The names of the authors of the book. Multiple authors are delimited by "/".

4) average_rating: The average rating of the book received in total.

5) isbn: Another unique number to identify the book, known as the International Standard Book Number.

6) isbn13: A 13-digit ISBN to identify the book, instead of the standard 11-digit ISBN.

7) language_code: Indicates the primary language of the book. For instance, "eng" is standard for English.

8) num_pages: The number of pages the book contains.

9) ratings_count: The total number of ratings the book received.

10) text_reviews_count: The total number of written text reviews the book received.

11) publication_date: The date the book was published.

12) publisher: The name of the book publisher.

Project Objectives:

The objective of the project is to predict the average rating assigned to each book. The project can be submitted as a Jupyter Notebook and should include exploratory analysis of the data, feature engineering and selection, model training and evaluation.

List of libraries:

Pandas: is an open-source library that provides high-performance, easy-to-use data structures and data analysis tools for the Python programming language. It is used extensively in data science and machine learning applications.

NumPy: is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting and discrete Fourier transforms…

Matplotlib: is a comprehensive open-source library for creating static, animated, and interactive visualizations in Python.

Seaborn: is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating informative and visually appealing statistical graphics.

Scikit-learn: is a Python library for machine learning that provides a wide range of tools for predictive data analysis.

I- Data Exploration

1. Read Data

```
# Load the dataset
df = pd.read_csv("/Users/p105660/Documents/ML_Project_Final/books.csv", error_bad_lines = False)
```

```
C:\Users\p105660\AppData\Local\Temp\ipykernel_11608\3144709383.py:2: FutureWarning: The error_bad_lines argument has been de
precated and will be removed in a future version. Use on_bad_lines in the future.


  df = pd.read_csv("/Users/p105660/Documents/ML_Project_Final/books.csv", error_bad_lines = False)
Skipping line 3350: expected 12 fields, saw 13
Skipping line 4704: expected 12 fields, saw 13
Skipping line 5879: expected 12 fields, saw 13
Skipping line 8981: expected 12 fields, saw 13
```

We observe that the dataset contains 4 bad lines. Therefore, it is necessary to proceed with the correction of the dataset.

```
# Number of lines and columns before cleaning
df.shape
```

```
(11123, 12)
```

We have 11123 lines before correction.

2- Check invalid values

```
# Drop lines with wrong number of columns
df = df.drop([587, 3350,4704,8981])
```

```
# Number of lignes and columns after cleaning
df.shape
```

```
(11119, 12)
```

I drop the bad lines to have just the data with the good format.

```
# check number of null lines for each column
df.isna().sum()
```

```
bookID                0
title                 0
authors               0
average_rating        0
isbn                  0
isbn13                0
language_code         0
  num_pages           0
ratings_count         0
text_reviews_count    0
publication_date      0
publisher             0
dtype: int64
```

Check number of null lines for each column.

```
# Drop the space for num_pages and all columns if it exist
df.columns = df.columns.str.replace(' ', '')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11119 entries, 0 to 11122
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   bookID              11119 non-null  int64
 1   title               11119 non-null  object
 2   authors             11119 non-null  object
 3   average_rating      11119 non-null  float64
 4   isbn                11119 non-null  object
 5   isbn13              11119 non-null  int64
 6   language_code       11119 non-null  object
 7   num_pages           11119 non-null  int64
 8   ratings_count       11119 non-null  int64
 9   text_reviews_count  11119 non-null  int64
 10  publication_date    11119 non-null  object
 11  publisher           11119 non-null  object
dtypes: float64(1), int64(5), object(6)
memory usage: 1.1+ MB
```

Drop the space if it exists for all columns before starting the analysis part.

3- Describe the data

```
df.describe()
```

	bookID	average_rating	isbn13	num_pages	ratings_count	text_reviews_count
count	11119.00000	11119.000000	1.111900e+04	11119.000000	1.111900e+04	11119.000000
mean	21312.59097	3.934102	9.759873e+12	336.394820	1.794094e+04	541.962497
std	13094.80650	0.350539	4.430554e+11	241.172409	1.125171e+05	2576.984016
min	1.00000	0.000000	8.987060e+09	0.000000	0.000000e+00	0.000000
25%	10277.50000	3.770000	9.780345e+12	192.000000	1.040000e+02	9.000000
50%	20315.00000	3.960000	9.780586e+12	299.000000	7.450000e+02	46.000000
75%	32104.50000	4.140000	9.780873e+12	416.000000	5.000500e+03	238.000000
max	45641.00000	5.000000	9.790008e+12	6576.000000	4.597666e+06	94265.000000

Generate descriptive statistics for all numeric columns in the DataFrame.

```
vars_categ = ['title', 'authors', 'isbn', 'language_code','publisher']
df[vars_categ].describe()
```

	title	authors	isbn	language_code	publisher
count	11119	11119	11119	11119	11119
unique	10345	6637	11119	27	2289
top	The Brothers Karamazov	P.G. Wodehouse	0439785960	eng	Vintage
freq	9	40	1	8905	318

Generate descriptive statistics for all categorical columns in the DataFrame.

Remak :

- We have 11119 books.
- Most of books are in English (8905/11119).
- P.G. Wodehouse is the author with the greatest number of books (40 books).
- The Brothers Karamazov is the most recurrent book (9 times).
- Most publisher is Vintage (318 books).

II. Data Analysis and ML

1- Type of problem

We are going to try to predict the average rating of each book, we are in the context of a supervised problem.

Predicting a value: it's a regression. We have a regression problem.

2- Data analysis

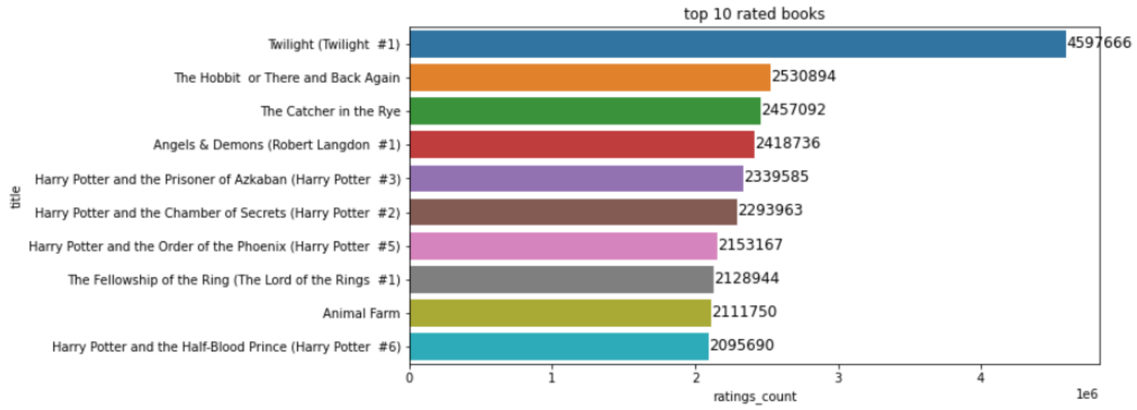
```
# The column to predict is average_rating
df.columns
```

```
Index(['bookID', 'title', 'authors', 'average_rating', 'isbn', 'isbn13',
       'language_code', 'num_pages', 'ratings_count', 'text_reviews_count',
       'publication_date', 'publisher'],
      dtype='object')
```

The column to predict is average_rating, we need to predict the average and compare with the real average rating.

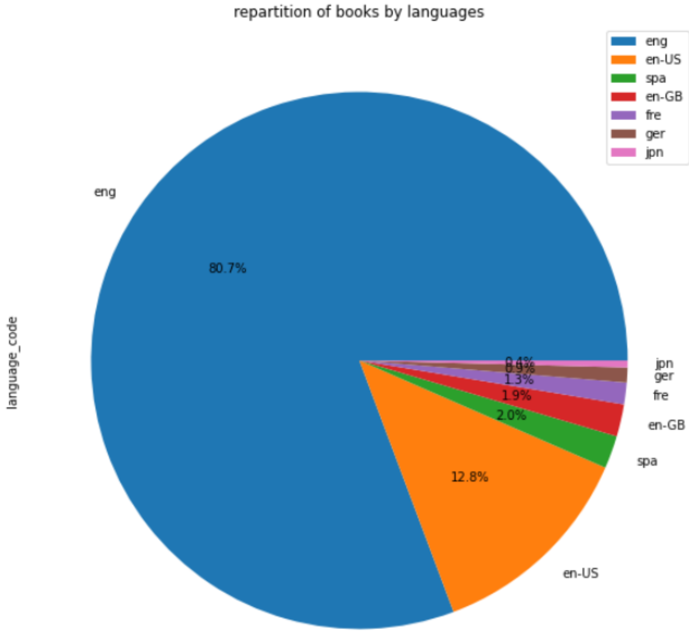
```
# The top 10 rated books
top_10 = df.nlargest(10, ['ratings_count']).set_index('title')['ratings_count']
plot_dims = (10, 5)
fig, ax = plt.subplots(figsize=plot_dims)
sns.barplot(top_10, top_10.index)
for i in ax.patches:
    ax.text(i.get_width()+.3, i.get_y()+0.5, str(round(i.get_width())), fontsize = 12)
plt.title('top 10 rated books')
plt.xlabel('ratings_count')
plt.show()
```

```
C:\Program Files (x86)\python39\V_3.9.8\soft\WPy64\python-3.9.8.amd64\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `dat
a`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



We list the top 10 rated books. Twighlit is the top one with 4597666 rates.

```
# Repartition books par by languages
df['language_code'].value_counts().head(7).plot(kind = 'pie', figsize=(10, 10), autopct='%1.1f%%').legend()
plt.title('repartition of books by languages')
plt.show()
```

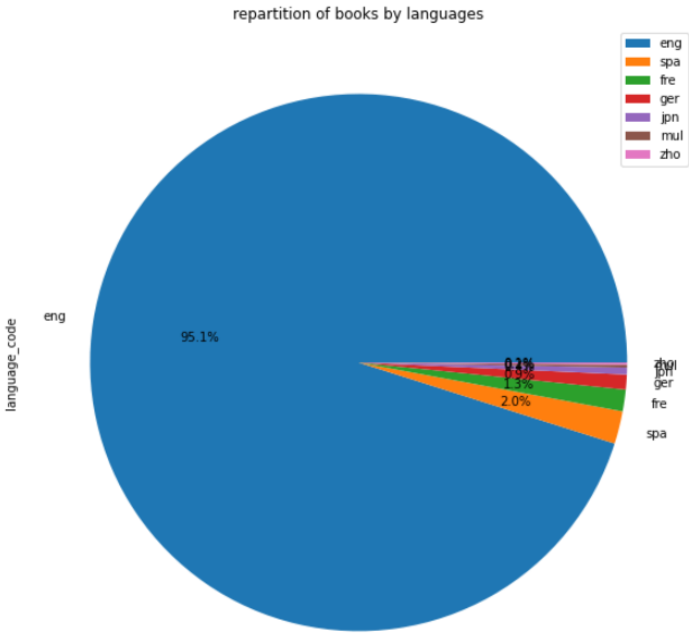


Repartition books par by languages. 80% are in Eng and 12.8% are in Eng-US and the third category is Spanish with 2%.

```
df['language_code'] = df['language_code'].replace('en-US', 'eng')
```

```
df['language_code'] = df['language_code'].replace('en-GB', 'eng')
```

```
# # Repartition books by languages after data processing
df['language_code'].value_counts().head(7).plot(kind = 'pie', figsize=(10, 10), autopct='%1.1f%%').legend()
plt.title('repartition of books by languages')
plt.show()
```



Here we replace en-US and en-GB by eng only. we have now 95% of books are in eng.

```
df_top_4_languages.shape
```

```
(10987, 12)
```

```
df_top_4_languages.language_code.value_counts()
```

```
eng    10526
spa      218
fre      144
ger       99
Name: language_code, dtype: int64
```

Create a new dataset with only the top 4 languages. So here we have English, Spanish, French and German.

3- Split the data

The objective is to split the data of the new dataset df_top_4_languages into two categories based on the ratings_count

df_HighRate	df_LowRate
It contains books that have received the total number of ratings the book received >= 100000.	It contains books that have received the total number of ratings the book received < 100000.
Length: (352, 12)	Length: (10635, 12)

4- Feature engineering

In this step, we will encode the categorical variables for our two datasets (df_HighRat, df_LowRat). The objective of this step is to assign numerical values to all textual variables.

```
# Encoding the auther variables for df_HighRate dataset
df_HighRate['authors'] = preprocessing.LabelEncoder().fit_transform(df_HighRate['authors'])
df_HighRate
```

We encore authors, title, and publisher variables exactly with the same logic. Each author for example should have a specific id.

```
# Encoding the language_code variables for df_HighRate dataset
encod_HR = pd.get_dummies(df_HighRate['language_code'])
cols = df_HighRate.columns.isin(['en-US', 'eng', 'fre', 'spa']).any()
if  cols == False:
    df_HighRate = pd.concat([df_HighRate, encod_HR], axis = 1)
print(df_HighRate.shape)
df_HighRate.head()
```

We Encode the language_code variables using get_dummies() function, and after we create a new column for each language, then we identify using binary code witch language is right for each book.

5- Selected model

For this case I'm ready to build the Linear Regression Model because Linear regression models are easier to adjust than models that are nonlinearly related to their parameters, and the statistical properties of the resulting estimators are easier to determine.

```
# split 80% of the data to the training set and 20% of the data to test set
df_train, df_test = train_test_split(df_LowRate,test_size = 0.2)
```

We split the data on two set, 80% as a training data and 20% as a test data.

```
## Prediction on the data set df_LowRate
X = df_LowRate.drop(['average_rating', 'language_code', 'isbn','isbn13','publication_date'], axis = 1)
y = df_LowRate['average_rating']

# split 80% of the data to the training set and 20% of the data to test set
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

#Create a linear regression model
regrlinear = LinearRegression()

# Train the model using the training data
regrlinear.fit(x_train, y_train)

# Predict the values of the test set
y_pred = regrlinear.predict(x_test)

# Display the regression coefficients
print('regression coefficients:', regrlinear.coef_)

# compare the actual and predicted values
df = pd.DataFrame({'actual': y_test, 'predicted': y_pred})
df['diff'] = df['actual'] - df['predicted']
print(df)
```

After splitting data into two set (80% training set and 20% test set), we create a linear regression model using the function LinearRegression(). The next step is to train the model using the training data.

We compute and display the regression coefficients. Finally, we display the actual and predicted values.

Performance of the model:

The performance of a linear regression model is typically evaluated using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

- Performance for df_LowRate (model1):

	actual	predicted	diff
3732	4.12	3.947800	0.172200
4461	4.28	4.073418	0.206582
5703	4.12	3.922140	0.197860
9213	4.21	3.965566	0.244434
5839	4.01	3.868952	0.141048
...	...	...	...
2354	4.10	3.930704	0.169296
1682	4.30	4.024139	0.275861
3943	3.36	3.925074	-0.565074
10269	3.42	3.884892	-0.464892
4182	3.32	3.855454	-0.535454

2127 rows × 3 columns

```
Mean Absolute Error (MAE): 0.22065635687167104
Mean Squared Error (MSE): 0.08492482748375448
Root Mean Squared Error (RMSE): 0.29141864642427134
Mean Absolute Error (MAE): 0.22065635687167104
```

- Performance for df_HighRate (model2):

	actual	predicted	diff
23	4.36	3.999243	0.360757
736	4.19	4.029507	0.160493
5093	4.29	4.176891	0.113109
591	4.07	4.024806	0.045194
3591	4.03	3.987512	0.042488
...	...	...	...
2852	3.82	3.918921	-0.098921
1556	4.00	4.005360	-0.005360
2960	4.44	4.191063	0.248937
936	3.91	4.011580	-0.101580
1742	3.83	3.995250	-0.165250

71 rows × 3 columns

```
Mean Absolute Error (MAE): 0.1547905334628017
Mean Squared Error (MSE): 0.03928908700467158
Root Mean Squared Error (RMSE): 0.1982147497152308
Mean Absolute Error (MAE): 0.1547905334628017
```

Performances Analysis :

Based on the provided error values, model2 appears to be the better one.

- Mean Absolute Error (MAE): model2 has a MAE of 0.1547905334628017, which is lower than the MAE of model1, which is 0.22065635687167104. This means that, on average, the predictions of model2 are closer to the actual values than those of model1.

- Mean Squared Error (MSE): model2 has a MSE of 0.03928908700467158, which is lower than the MSE of model1, which is 0.08492482748375448. The MSE gives more weight to larger errors because it squares the residuals. A lower MSE therefore indicates that model2 has fewer large errors than model1.

- Root Mean Squared Error (RMSE): model2 has a RMSE of 0.1982147497152308, which is lower than the RMSE of model1, which is 0.29141864642427134. This means that, on average, the square root of the squared differences between the predicted and actual values in model2 is lower than that of model1.