Overview

Member 3 : Divesh Sanjay Patil

Dataset : Amazon Books Metdata (Kaggle Dataset)

Objective : To create an interpret machine learning models like in irrigation and random forest that praise predict Amazon book popularity using meta data features and to identify the key factors influencing this predictions through model explain techniques such as feature, importance and sharp

Research Question : Using only book Metadata, can machine learning models, predict a books, popularity (ratings_count, text_review_count) and which metadata features are the strongest predictors of popularity?.

Dataset: Original data size :- 11119 Books . After Cleaning, filtering it and then preprocessing ,a balanced and high-quality of 10,463 books dataset was selected to meet the project requirements.

[156]:
```
#only choose the english langauges
english_languages=['eng','en-US','en-GB']

#through 'isin()'which checks if column contain multiple values. and through copy funcation 'amazon_books'in 'english_books'
english_books=(
    amazon_books[amazon_books['language_code'].isin(english_languages)]
    .copy()
)

english_books.shape
```

[156]:
```
(10463, 12)
```

[195]:
```
english_books.head(4)
```

[195]:

bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	publication_date	publisher
1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591	9/16/2006	Scholastic Inc.
2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221	9/1/2004	Scholastic Inc.
4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9780439554893	eng	352	6333	244	11/1/2003	Scholastic
5	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9780439655484	eng	435	2339585	36325	5/1/2004	Scholastic Inc.

KDD- Methodology

- 1. Data Preprocessing (Cleaning,handling missing values and remove noise):
- Data was examine using info(), describe(),shape to understand data types, missing values & overall structure.
- 0 duplicates rows were identified .
- Using isna().sum() the missing values was checked and identified as 0.
- Dropped some rows in rating counts with zero ratings because I want only the popular books according to my research question
- With the help of value_count (), books of multiple language was detected where I kept only English books and the rest were dropped to avoid the noise and messy data.

```
[10]: #Handle missing values
      raw_bookdataset.isna().sum()

[10]: 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

[12]: amazon_books=raw_bookdataset.copy()   #store'raw_bookdataset' in 'raw_books'

      #dropna() rows with missing values
      amazon_books=(
          amazon_books.dropna(
              subset=[
                  'ratings_count',
                  'text_reviews_count',
                  'average_rating',
                  '  num_pages',
                  'publication_date'
              ]
          )
      )
      #remove only with 0 ratings because i want only popularity book's accordong to research quetion.
      amazon_books=(
          amazon_books[amazon_books['ratings_count']>0]
      )

      amazon_books.shape  # with the help of '.shape' only shows the total rows and columns after removing content.

[12]: (11039, 12)
```

```
amazon_books['language_code'].value_counts().head(25)

language_code
eng      8843
en-US    1406
en-GB     214
spa       212
fre       140
ger        96
jpn        45
mul        19
zho        14
por        10
grc         9
en-CA       7
ita         5
enm         3
lat         3
rus         2
swe         2
nl          1
ara         1
msa         1
glg         1
wel         1
nor         1
tur         1
gla         1
Name: count, dtype: int64

#only choose the english langauges
english_languages=['eng','en-US','en-GB']

#through 'isin()'which checks if column contain multiple values. and through copy funcation 'amazon_books'in 'english_books'
english_books=(
    amazon_books[amazon_books['language_code'].isin(english_languages)]
    .copy()
)

english_books.shape

(10463, 12)

english_books.head(4)
```

bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	publi
1	Harry Potter and the Half-Blood Prince	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591	

- **3.Transformation (Feature Engineering and log transformation):**
- Converted publication date to proper date time using to_datetime ()
- Created some new columns, which helps my machine learning model to understand and more about the book characteristics and the popularity of it.
- Log transformation is used for it takes the large number of data which is compressed into small range.
- It is actually very useful when data is very large and very small and everything in between just like my dataset where there are many large values for that log transformation is used.

```
#feature enginerring
#create useful features.
only_english_books=english_books.copy()

#convert 'publication_date'into proper datetime type through '.to_datetime()'funcation.
only_english_books['publication_date']=(pd.to_datetime(
        only_english_books['publication_date'],
        errors='coerce'
    )
)

#drop those rows where on any punlication date available through '.dropna()' funcation
only_english_books=only_english_books.dropna(
    subset=['publication_date']
)
```

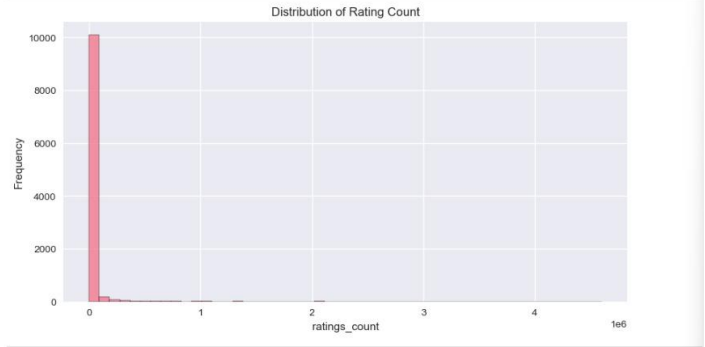
```
#make new column of publication year
only_english_books['publication_year']=(
    only_english_books['publication_date'].dt.year
)

#here assuming current year means particular book how many years the book has been in the market.
only_english_books['books_age']=(
    2025-only_english_books['publication_year']
)

# gives Rating per page  means compute a ratio with ratings divided by pages here use '+1' if in case any 'num_pages'is 0 or corrupte
only_english_books['rating_per_page']=(
    only_english_books['average_rating']/(only_english_books['  num_pages']+1)
)

#gives reviews per rating this ratio shows how many people to write a review or only gives the rating.
only_english_books['reviews_per_rating']=(
    only_english_books['text_reviews_count']/(only_english_books['ratings_count']+1)
)
```

```
#with the help of 'np.log1p'funcation compresses large values and makes the distribution more normal and smmoth as well as reduce imp
only_english_books['log_ratings_count']=np.log1p(
    only_english_books['ratings_count']
)

only_english_books['log_text_reviews_count']=np.log1p(
    only_english_books['text_reviews_count']
)

only_english_books.head()
```

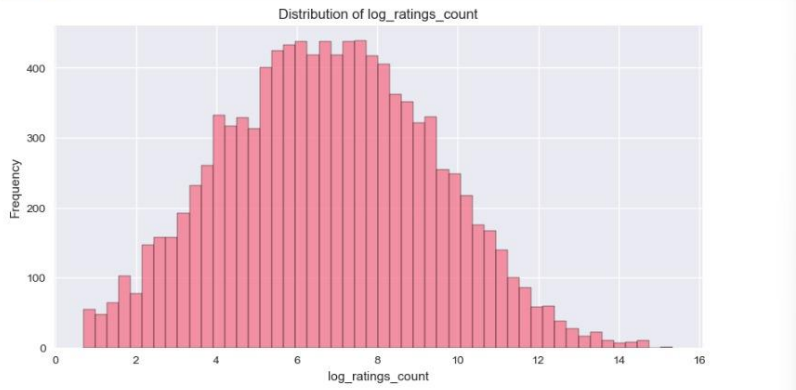
	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	p
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591	
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221	
		Harry Potter and the									

- Scaling :
- The histogram shows one tail bar on the left and almost no bars on the right, which is extremely right skipped.
- A large number of books have low rating counts(close to 0 -500). Small number of books have extremely high rating counts. Which indicates heavy skewness and outlaws, which negatively affect the raw rating count and it is not suitable for modelling due to extreme skewness so use log transformation.
- After applying lock transformation, bell ship cover was met, which shows schemes significantly reduce.
- The values are been evenly distributed from 1 to 15, which shows better variant more stable pattern for ML models
- Histogram looks balance and symmetric, which is ideal for regression feature, importance, and interpretation

```
#Exploratory Data Analysis
#with the help of plots get to know data understanding

#Distribution of rating count
plt.figure(figsize=(10,5))                    #it's shows figure width and height
sns.histplot(
    only_english_books['ratings_count'],
    bins=50
)                                             #hisplot creates histogram plots and split the whole range of 'ratings_count' into 50 
plt.title('Distribution of Rating Count') # it gives the title on the top of the chart
plt.xlabel('ratings_count') #it gives name for x-axis
plt.ylabel('Frequency')   # #it gives name for y-axis
plt.show() #show graph
```

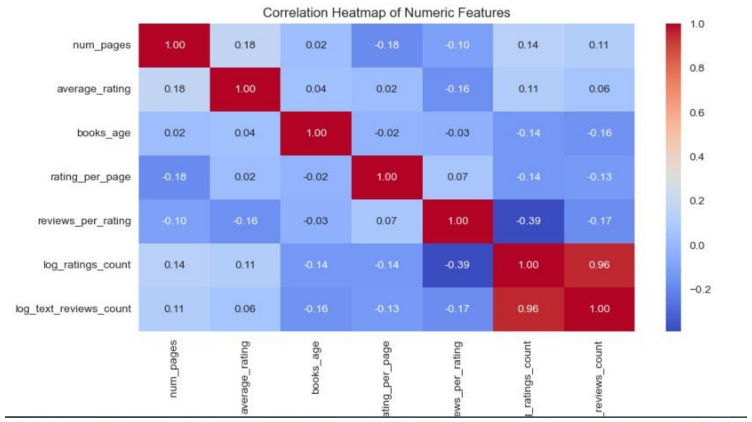



```
#Distribution after log transform

plt.figure(figsize=(10,5))
sns.histplot(
    only_english_books['log_ratings_count'],
    bins=50
)
plt.title('Distribution of log_ratings_count')
plt.xlabel('log_ratings_count')
plt.ylabel('Frequency')
plt.show()
```



Heatmap : Use a correlation hit map of numeric features with the output shows that how strongly two numeric values moves together where the value ranges from plus one which is perfect positively relationship which is red colour and -1, which show the perfect negative relationship which is blue in colour and no relationship that is zero means very weak which has been shown in white and light colour. It generally help us to understand that which features are useful for prediction

```
#correlation heatmap

main_columns=['  num_pages',
              'average_rating',
              'books_age',
              'rating_per_page',
              'reviews_per_rating',
              'log_ratings_count',
              'log_text_reviews_count'
             ]
plt.figure(figsize=(10,5))

#'corr()'calculate correlation matrix and annot=True tells seaborn write the number inside each square.
#'fmt='.2f' this formats numbers to two decimal places.and cmap gives colour

sns.heatmap(
    only_english_books[main_columns].corr(),
    annot=True,
    fmt='.2f',
    cmap='coolwarm'
)
plt.title('Correlation Heatmap of Numeric Features')
plt.show()
```



4.Model Preparation:

- Feature set(X) used the num_pages,average_rating,books_age, rating_per_page,reviews_per_rating
- Feature set (Y) is the target variable is log_ratings_count.
- Data is split into 80 % of training data and 20 % of testing.
- Stratify = y makes sure the rating distribution reman consistent in both train and test sets.
- Random state=42 makes sure about split is reproducible for consistent results.

```
#prepare x and y for training and testing.
#y is the target variable
y=only_english_books['log_ratings_count']    #predict popularity of books. it's target.
features_columns=['  num_pages',
                  'average_rating',
                  'books_age',
                  'rating_per_page',
                  'reviews_per_rating'
                 ]                           #feature base on metadata means in the dataset.

#here define the x
x=only_english_books[features_columns    ]# all the book information here
x.head()
```

	num_pages	average_rating	books_age	rating_per_page	reviews_per_rating
0	652	4.57	19	0.006998	0.013166
1	870	4.49	21	0.005155	0.013571
2	352	4.42	22	0.012521	0.038522
3	435	4.56	21	0.010459	0.015526
4	2690	4.78	21	0.001776	0.003959

```
#split data into train and test

x_train,x_test,y_train,y_test=train_test_split(
    x,y,
    test_size=0.2,
    train_size=0.8,                    #training data tech the ML model
    random_state=42                    #testing data check how the model learn.
                                       #0.2(20%)=test, 0.8(80%)=training , random_state it make sure the split is reproducible.
)

x_train.shape,x_test.shape
```

```
((8369, 5), (2093, 5))
```

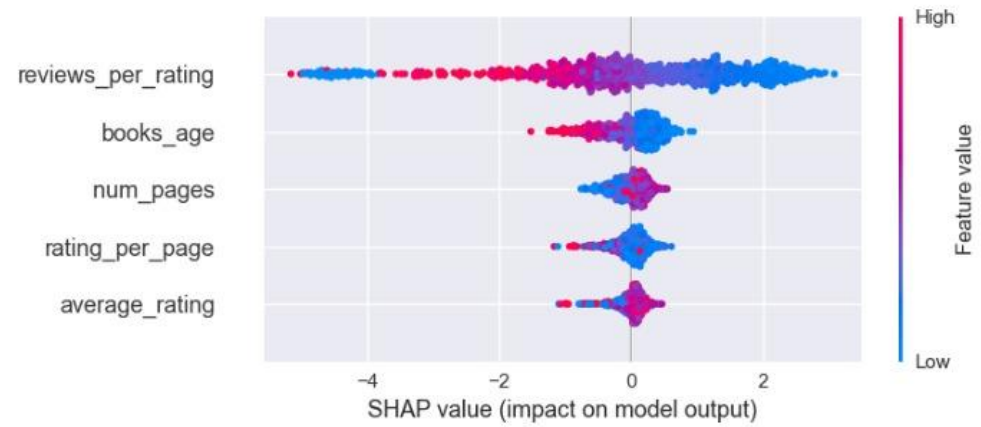
- **Linear Regression Model and Random Forest:**
- The comparison table shows that I have used two models that is linear regression and the random forest, so I comparing both the models we get the values as RMSE 2.313915, which tells that it makes an average prediction error of plus -2.31 law rating units which is quite large and whereas MAE is equals to 1.808 tells that on average predictions are 1.8 units away from the truth values and R Square which is equal to 0.2085 and this means that the model only explains 20% of the variation in book popularity so basically by saying at this three values we can say that linear regression is weak and it can not capture the pattern in the data. It performs poorly because the features are non-linear relationships.
- The second model random forest output where RMSE is 1.90 that is much smaller error compare to the linear regression where MAE equals to 1.49,which means that on average the prediction error drops by -18% whereas R square which is 0.463 that is random forest. Explain 46% of the variation in book popularity. This is a huge improvement over the linear regression by 20%.So by comparing both the models random forest is the best model because the random forest build many trees and it learns a complex patterns like branches, whereas it has predicted the book popularity more accurately as compare to linear regression as a random forest work better with the countless data. It handles the feature interactions and also it handles outliers and skewed data.'

]:

	model1	RMSE	MAE	R2
0	Linear Regression	2.313915	1.808256	0.208586
1	Random Forest	1.905587	1.493010	0.463257

- SHAP :
- This plot explains how each feature affects the Random Forest model's predictions of book popularity (log_ratings_count).
- Pink / Red shows the High feature value example: high average rating, high reviews_per_rating
- Blue color showing the Low feature value example: low rating, fewer reviews, fewer pages
- SHAP helps us interpret the Random Forest by showing how each feature contributes to predicting book popularity. Pink dots represent high feature values, blue dots represent low values, and the x-axis shows whether the feature pushes the prediction up or down. We found that 'reviews_per_rating' is the most influential feature, strongly increasing popularity when high. This insight is only visible through SHAP, which satisfies the project requirement for model interpretability.

```
#SHAP (Shaply Additive exPlanations) interpretability for random forest
x_sample_data=x_test.sample(      #here through'.sample()'funcation pick up therendom subset of rows from x_test
    min(1000,len(x_test)),        # here x_test takes only 1000 rows because  SHAP works slow on very big data.
    random_state=42             #select the random books from dataset
)
shap_explainer=shap.TreeExplainer(randomforest_model)  # here 'shap.TreeExplainer' build the a special explainer designed for tree models simply it is wo
shap_values=shap_explainer.shap_values(x_sample_data)  #through SHAP get to know how random forest predicts book popularity so explain it.
shap.summary_plot(shap_values,x_sample_data)   #here SHAP creates the beeswarm plot to represent the view of feature importance and direction
```



Conclusion :

- This project shows that Amazon book popularity can be predicted using metadata features, with reviews_per_rating being the most important factor.
- Books that receive more written reviews relative to ratings tend to be significantly more popular. Other features like average rating, book age, and number of pages have smaller effects.Between the two models tested, Random Forest performed much better than Linear Regression, proving that book popularity follows a non-linear pattern.
- SHAP analysis confirmed these findings by clearly explaining how each feature influences predictions.Overall, the study concludes that reader engagement drives popularity, and Random Forest is the most suitable model for predicting book popularity in this dataset.