PES University-EC campus
UE20CS312 - Data Analytics Final Report
Section: F, G, I and J

Project Title : Book recommendation System

Project Team Name : VS

SRN1: PES2UG20CS352 Name1: SRUSHTI S N

SRN2: PES2UG20CS391 Name2: VISMAYA R

SRN3: PES2UG20CS406 Name3: SHUBHANGI SADHWANI

PROBLEM STATEMENT:

As recommendation systems become critical in many industries, we aim to build one for the "bibliophiles".

As the user selects a book, we recommend other book titles he might be interested in, based on various similarity factors.

Importance of our problem statement

- In a world where search engines can supply the user with any information and resource they need, searching manually for the next thing a user is interested in can become laborious.

- The user and item data can help improve overall services and ensure that they are according to user's preferences.

- It provides a better experience for the users by giving them a broader exposure to many different products they might be interested in.

Importance of our problem statement

- Books are the summary of human knowledge and a world of magnificent escape.

- While there are multiple popular and effective recommendation models for domains such as Movies, Series etc.

- The project depends deeply on domain knowledge to be able to provide more relevant recommendations like other content-based systems.

Our Dataset

The dataset provides close to seven thousand books containing identifiers, title, subtitle, authors, categories, thumbnail url, description, published year, average rating, and number of ratings.

https://www.kaggle.com/datasets/dylanjcastillo/7k-books-with-metadata

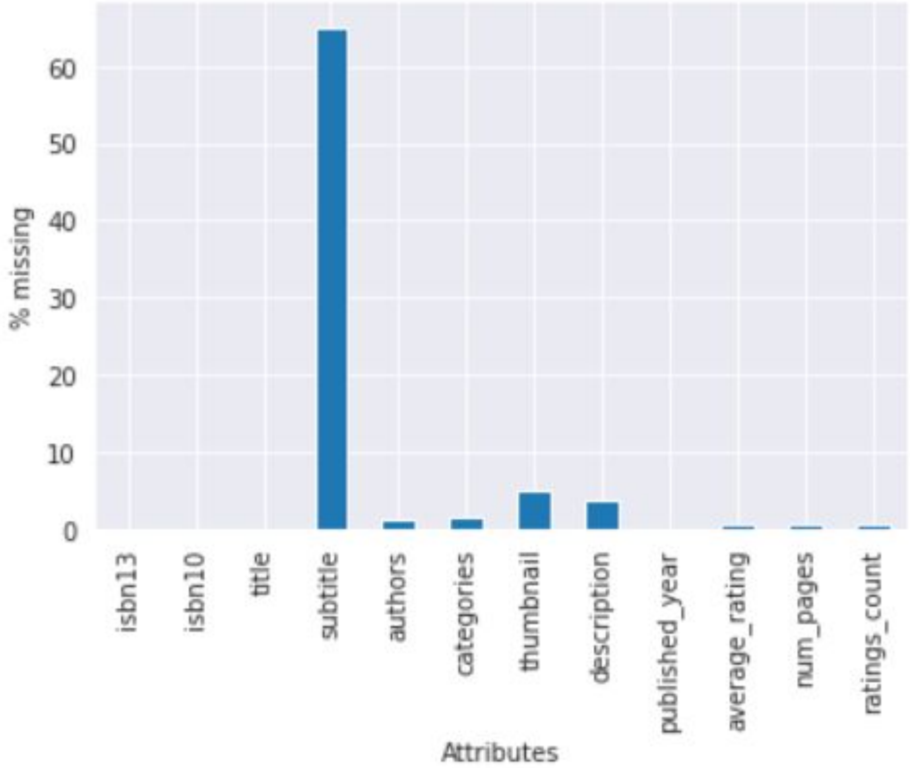
```
In this dataframe, there are:
6810 rows,
12 columns

 The attributes/features are:
isbn13              int64
isbn10             object
title              object
subtitle           object
authors            object
categories         object
thumbnail          object
description        object
published_year    float64
average_rating    float64
num_pages         float64
ratings_count     float64
dtype: object
```

```
df.head()
```

	isbn13	title	authors	categories	description	published_year	average_rating	num_pages	ratings_count
0	9.780000e+12	gilead	marilynne robinson	fiction	a novel that readers and critics have been eag...	2004.0	3.85	247.0	361.0
1	9.780000e+12	spider's web	charles osborne;agatha christie	detective and mystery stories	a new 'christie for christmas' -- a full-lengt...	2000.0	3.83	241.0	5164.0
2	9.780010e+12	the one tree	stephen r. donaldson	american fiction	volume two of stephen donaldson's acclaimed se...	1982.0	3.97	479.0	172.0
3	9.780010e+12	rage of angels	sidney sheldon	fiction	a memorable, mesmerizing heroine jennifer -- b...	1993.0	3.93	512.0	29532.0
4	9.780010e+12	the four loves	clive staples lewis	christian life	lewis' work on the nature of love divides love...	2002.0	4.15	170.0	33684.0

Analysis of missing data



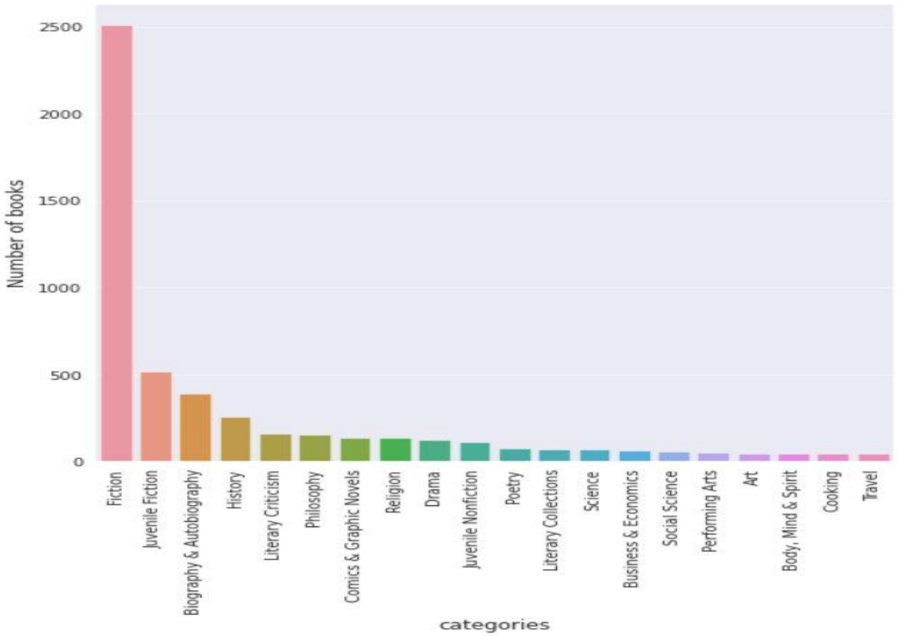
Mean Value Imputation

Data Imputation: We performed mean imputation for average rating and ratings count.

Dimensionality reduction

- *Dropping the feature:*
  There are 65.02% of missing values in the feature 'subtitle', so we can conclude it is as incomplete data and drop the feature.
- *Dropping null instances: Drop all the rows with missing values in the following columns: 'authors', 'categories','description'*
- *Dropping unnecessary columns: ‘published_year’, ‘num_pages’, ‘isbn10’,’isbn13’, ‘thumbnail’*

EDA



The top three categories of books preferred by the readers belong to the categories of 'fiction' ,'juvenile fiction' and 'biography & autobiography' Respectively.

The genre "Fiction" has almost 5x more examples than the second place "Juvenile Fiction"

The dataset is hence unbalanced.

However, there is no problem for now, since we will only use these for filtering our dataset and calculate the cosine similarity.

Content-Based recommendation system

Reasons for choosing:

1. *Our dataset contains significant amount of attribute information as opposed to user rating data*
2. *The attributes in our dataset are text-rich, so a content-based system is particularly well suited*
3. *Our dataset is comparatively small*

Algorithm

1. *Data wrangling on the 'description' and 'authors' columns to extract relevant keywords and combine them into 'keywords' column*

The Recommendation:

2. *Fetch the 'category' of the title and filter the dataset based on the category*
3. *Convert the reduced dataset to 'title' based indexing*
4. *Calculate the 'cosine similarity' of the title with other titles based on the 'keywords' attribute.*
5. *Get top 10 most similar titles and sort them based on weighted rating.*
6. *Return the top 5 books with the highest weighted rating as the recommendations*

Weighted rating

- *We use Average ratings and ratings count to assign weighted average score.*
- *Use weighted average to sort the recommended books.*
- *Calculate all the components based on the formula*
- *W=Weighted rating*
- *R=Average ratings*
- *C=Mean of the average ratings*
- *m=Minimum number of ratings*
- *v=ratings count*

$$W = \frac{Rv + Cm}{v + m}$$

https://medium.com/@developeraritro/building-a-recommendation-system-using-weighted-hybrid-technique-75598b6be8ed

Weighted rating

```
## WEighted average
v=df['ratings_count'] #11800, 4500, 4466, etc.
R=df['average_rating'] # 7.2, 6.9, 6.3, 7.6, etc.
C=df['average_rating'].mean() # 6.092171559442011
m=df['average_rating'].quantile(0.70) # 581.0

df['weighted_rating']=((R*v)+ (C*m))/(v+m)
df['weighted_rating']
```

```
0       3.850915
1       3.830080
2       3.969110
3       3.930000
4       4.149973
          ...
6803    3.733733
6804    3.820291
6805    4.488117
6808    3.931679
6809    3.767251
Name: weighted_rating, Length: 6408, dtype: float64
```

Recommender function

```
def recommend(title):
    title= title.lower()
    titlerow = df.loc[df['title']== title].iloc[0]   #Fetch the category of our title
    category=titlerow['categories']
    data = df.loc[df['categories'] == category]   # MATCH THE CATEGORY WITH THE COLUMN "CATEGORIES" OF THE DATASET
    if len(data)<=5: ##As our dataset is unbalanced, if the matching category contains no other book title
      data=df           #,then we ommit the category filtering

    data.reset_index(level = 0, inplace = True, drop=True)  # RESET INDEX
    indices = pd.Series(data.index, index = data['title'])   # INDEX TO A PANDAS SERIES

    tf = TfidfVectorizer(analyzer='word', ngram_range=(2, 2), min_df = 1, stop_words='english',sublinear_tf=True)
    tfidf_matrix = tf.fit_transform(data['keyword'])
    similarity = cosine_similarity(tfidf_matrix, tfidf_matrix) # CALCULATE THE SIMILARITY MEASURE

    title_index = indices[title].tolist() # GET THE INDEX OF ORIGINAL TITLE
    if not(type(title_index) is int):
        title_index=title_index[0]    #if more than one matching index exists, take the 1st one
        inds=indices[title].tolist()
        for i in inds:   #to drop other rows with the same title
          if i!=title_index:
            data.drop(i,inplace=True)

# PAIRWISE SIMILARITY SCORES
similarity = list(enumerate(similarity[title_index]))
similarity = sorted(similarity, key=lambda x: x[1], reverse=True) # SORT THE BOOKS
similarity  = similarity [1:11] # GET TOP 10 MOST SIMILAR BOOKS
book_indices = [i[0] for i in similarity]

#Weighted Rating method
top5_rated = data['weighted_rating'].iloc[book_indices]
wsort = top5_rated.sort_values(ascending = False)
wsort_top5 = wsort[:6]
wsort_top5.to_frame()

wsort_indices = wsort_top5.index    # INDICES OF TOP 5
rec = data[['title']].iloc[wsort_indices]    # TOP 5 RECOMMENDATION
print(rec['title'])    # PRINT THE BOOKS TITLE
```

Model result

```
recommend("Murder on the Orient Express")
```

```
515      absent in the spring and other novels
2370                      the body in the library
2377                                      prophet
2369                         murder at the vicarage
2374                              death on the nile
2373                           a murder is announced
Name: title, dtype: object
```

```
recommend("harry potter")
```

```
243                        the harry potter collection
239       harry potter and the half-blood prince (book 6)
229     harry potter and the prisoner of azkaban (book 3)
389                     harry potter and the goblet of fire
202     harry potter and the order of the phoenix (boo...
219        harry potter and the sorcerer's stone (book 1)
Name: title, dtype: object
```

Model Evaluation

- *Evaluation of recommender systems is usually done in 3 ways: Online Evaluation, User Studies and Offline Evaluation.*
- *As we do not have already existing users, we cannot perform user studies.*
- *For offline evaluation, we would need historical data of the recommendations, which were not a part of our dataset and therefore we chose to go with Online evaluation.*

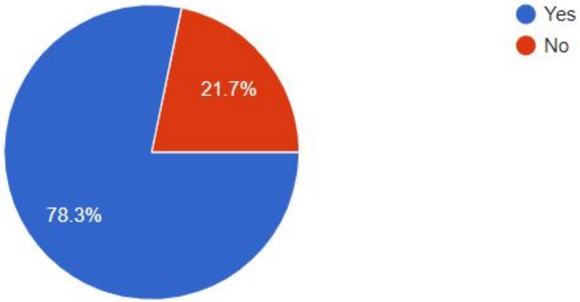
Online Evaluation:

- *The Online evaluation results for this model were obtained by performing a user survey on our peers and fellow book readers via 'google forms'.*
- *The survey was conducted to evaluate the quality of the recommender model for effectiveness based on secondary goals of the recommender systems such as Novelty, Serendipity and Diversity.*

Model Evaluation

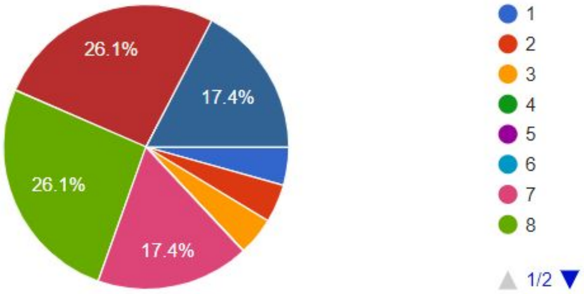
Do you enjoy reading books?

23 responses



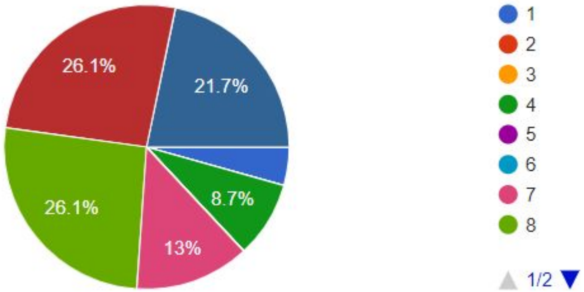
Were you recommended any books that you weren't aware of? (NOVELTY)

23 responses



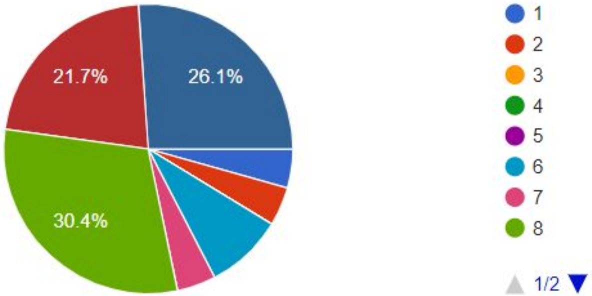
Did you get any books that you would be interested to read? (SERENDIPITY)

23 responses



Did you find variety in the books recommended? (DIVERSITY)

23 responses



Considering the threshold as 7, anything above it being a positive response, we obtained the following results:

- Among the surveyed group, it was found that 76.2% of the users received a recommendation they did not know of before.
- 81% of the users took interest in the books that were recommended to them
- 85.7% of the users agreed that they found variety in the books recommended to them by the recommender system.

Thank You