Online Book Recommendation System Using KNN Algorithm

Misbah Rajput
Department of computer science
Sukkur IBA University

Abstract: A recommendation system is one of the top utilizations of data science. Each shopper Internet organization requires a recommendation system like Netflix, Youtube, a news source, and so forth. What you need to show out of a tremendous scope of things is a recommendation system.Online book perusing and selling websites like Kindle and Goodreads contend with one another on many variables. One of those significant elements is their book recommendation system. A book recommendation system is intended to prescribe books important to the buyer.The motivation behind a book recommendation system is to foresee purchaser's advantage and prescribe books to them in like manner. A book recommendation system can consider numerous boundaries like book content and book quality by sifting client reviews. In the segment underneath, I will acquaint you with an AI project on the book recommendation system utilizing Python.

Keywords: Books suggestion, online recommendation, book reviews system, KNN algorithm, using Python

INTRODUCTION: Proposal frameworks were advanced as keen calculations, which can create brings about the type of suggestions to clients. They diminish the above related with settling on best decisions among the bounty. Presently, Recommender frameworks can be carried out in any area from E-COMMERCE to organize security as customized administrations. They give advantage toboth the customer and the maker, by recommending things to shoppers, which can't be requested until the recommendations[1].These days how much data particularly in Internet development quickly. Finding essential data becomes more troublesome. Proposal frameworks plan to settle this sort of issues[2].In the event that we take every one of the books and every one of the users for modeling, Don't you suppose will it create a problem? So what we have to do is we have to decrease the number of users and books because we can't consider a user who has just registered on the website or has just read one or two books[3].

Problems faced by recommender systems:

A. Data Sparsity Data sparsity arises from the phenomenon that users in general rate just a limited number of items.

B. Cold Start Personalized recommender systems take advantage of users' previous history to make predictions. The cold start problem apprehends the personalized recommendations for customers with zero or negligible previous history.

C. Gray Sheep The collaborative filtering (CF) approach in recommender systems assumes that users' preferences are consistent among users. It is presumed that few of these consumers belong to a minor local area of users who have unprecedented likings, such users are incompliant with the CF fundamental hypothesis. They are gray sheep users.

D. Shilling Attacks Recommender systems which are based on collaborative filtering are vulnerable to "shilling attacks" due to their open nature. Shillers inject a few deceitful "shilling profiles" into the database of evaluations for altering the system's recommendation, due to which some inappropriate items are recommended by the system.

E. Scalability Recommender systems assume a significant part in online activities by making personalized recommendations to users, as finding what users are searching for among an enormous number of items in huge databases is a tedious work[4].

In this paper we propose utilizing recommendation frameworks for suggesting books. We fostered a framework,which learns client inclinations by requesting to rate books and picking most loved classes and afterward produce the rundown of books client most presumably might want to peruse.

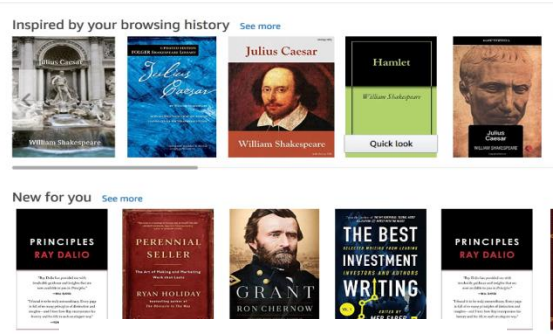


Fig : 1

Related work: Over the most recent 20 years there has been a considerable growth in the field of recommender systems. The research works like . The objective of the most recommendation system is to predict the buyer's interest and recommends the books appropriately. This book recommendation has considered numerous parameters like content of the book and nature of the book by doing collaborative filtering of ratings by the other buyers. [1][2].They reduce the overhead associated with settling on best decisions among the plenty. Presently, Recommender systems can be implemented in any space from E-commerce to network security as personalized services. They provide benefit to both the consumer and the manufacturer, by suggesting items to consumers, which can't be demanded until the recommendations[3].The book suggestion that utilizes Collaborative filtering with Jaccard Similarity (JS) to give more precise proposals. JS is based on an index determined for a couple of books. It is a proportion of normal clients (clients who have evaluated the two books) partitioned by the amount of clients who have evaluated the two books exclusively. Bigger the quantity of normal clients higher will be the JS Index and subsequently better suggestions. Books with high JS index (more suggested) will show up on top of the suggested books list[4].Recommender frameworks have turned into a fundamental examination field since the development of the primary paper on collaborative filtering during the 1990s. By and large, these frameworks are expressed as the emotionally supportive networks which assist clients with tracking down content, items, or administrations, (for example, books, motion pictures, music, TV programs, and sites) by get-together and looking at ideas from different clients, and that implies audits from different foundations, and clients. These frameworks are comprehensively characterized into collaborative filtering (CF) and content-based filtering (CB). CF is a data filtering practice that is based on the client's assessment of things or past buys records[5][6].In this paper we propose involving suggestion frameworks for

suggesting books. We fostered a framework, which learns client inclinations by requesting to rate books and picking most loved classifications and afterward create the rundown of books client most presumably might want to read.The benefit of this framework is in its speed and straightforwardness. The majority of the current administrations need a profile history data and other data that need a chance to give clients proposals while our point was to produce suggestions for clients in an exceptionally fast manner.

Methodology: Book Recommendation System with python.i have used item based collaborative filterinhg approach as follows,In this section, I will take you through how to construct a Book recommendation system with Machine Learning utilizing Python. I will start this errand by bringing in the necessary Python libraries and the dataset:

Language: Python used in this project to implement the code with Jupyter notebook

Libraries: We had imported following libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import neighbors
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

Dataset: books.csv

Description: book dataset contains the following attributes information:

- ✓ 1.bookID
- ✓ 2.title
- ✓ 3.authors
- ✓ Average Rating
- ✓ isbn
- ✓ isbn13
- ✓ language code
- ✓ num_pages
- ✓ ratings_count
- ✓ text_reviews_count
- ✓ publication_date
- ✓ publisher

Data Exploration:

The dataset that contains data about the books, who
data.

We have used only top 10 ratings :

```
top_ten = df[df['ratings_count'] > 1000000]
top_ten.sort_values(by='average_rating', ascending=False)
plt.style.use('seaborn-whitegrid')
plt.figure(figsize=(10, 10))
data = top_ten.sort_values(by='average_rating', ascending=Fal
sns.barplot(x="average_rating", y="title", data=data, palette
```

Ranking of data is found using data present in boo

```
df2.loc[ (df2['average_rating'] >= 0) & (df2['average_rating'] <= 1), 'rating_between'] = "between 0 and 1"
df2.loc[ (df2['average_rating'] > 1) & (df2['average_rating'] <= 2), 'rating_between'] = "between 1 and 2"
df2.loc[ (df2['average_rating'] > 2) & (df2['average_rating'] <= 3), 'rating_between'] = "between 2 and 3"
df2.loc[ (df2['average_rating'] > 3) & (df2['average_rating'] <= 4), 'rating_between'] = "between 3 and 4"
df2.loc[ (df2['average_rating'] > 4) & (df2['average_rating'] <= 5), 'rating_between'] = "between 4 and 5"
```

```
most_books = df.groupby('authors')['title'].count().reset_index().sort_values('
plt.figure(figsize=(15,10))
ax = sns.barplot(most_books['title'], most_books.index, palette='inferno')
ax.set_title("Top 10 authors with most books")
ax.set_xlabel("Total number of books")
totals = []
for i in ax.patches:
    totals.append(i.get_width())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_width()+.2, i.get_y()+.2,str(round(i.get_width())), fontsize=
```

me to assign the r

```
rating_df = pd.get_dummies(df2['rating_between'])
language_df = pd.get_dummies(df2['language_code'])
```

```
most_rated = df.sort_values('ratings_count', ascending = False).head(1
plt.figure(figsize=(15,10))
ax = sns.barplot(most_rated['ratings_count'], most_rated.index, palette = 'inferno')
totals = []
for i in ax.patches:
    totals.append(i.get_width())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_width()+.2, i.get_y()+.2,str(round(i.get_width())),
plt.show()
```

wing data:

ıde
grades

```
features = pd.concat([rating_df,
                      language_df,
                      df2['average_rating'],
                      df2['ratings_count']], axis=1)
```

We will attempt to use this column to find the mos

To find the relation between average score and given process.

The algorithm will find the median for all and equaliz

```
df.average_rating = df.average_rating.astype(float)
fig, ax = plt.subplots(figsize=[15,10])
sns.distplot(df['average_rating'],ax=ax)
ax.set_title('Average rating distribution for all books',fontsiz
ax.set_xlabel('Average rating',fontsize=13)
```

```
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler()
features = min_max_scaler.fit_transform(features)
```

algorithm implementation:

```
model = neighbors.NearestNeighbors(n_neighbors=6, algorithm='ball_tree')
model.fit(features)
dist, idlist = model.kneighbors(features)
```

The model buld the list on the basis of different featu

Average rating:

```
ax = sns.relplot(data=df, x="average_rating", y="ratings_count", color = 'red', sizes=
plt.title("Relation between Rating counts and Average Ratings",fontsize = 15)
ax.set_axis_labels("Average Rating", "Ratings Count")
```

```
def BookRecommender(book_name):
    book_list_name = []
    book_id = df2[df2['title'] == book_name].index
    book_id = book_id[0]
    for newid in idlist[book_id]:
        book_list_name.append(df2.loc[newid].title)
    return book_list_name

BookNames = BookRecommender('Harry Potter and the Half-Blood Prince (Harry Potte
BookNames
```

Data Preparation:

Here we will implement the rating column.

Results: following are the results for the given implementation of the book recommendation system:

	bookID	average_rating	isbn13	num_pages	ratings_count	text_reviews_count
count	11123.000000	11123.000000	1.112300e+04	11123.000000	1.112300e+04	11123.000000
mean	21310.856963	3.934075	9.759880e+12	336.405556	1.794285e+04	542.048099
std	13094.727252	0.350485	4.429758e+11	241.152626	1.124992e+05	2576.619589
min	1.000000	0.000000	8.987060e+09	0.000000	0.000000e+00	0.000000
25%	10277.500000	3.770000	9.780345e+12	192.000000	1.040000e+02	9.000000
50%	20287.000000	3.960000	9.780582e+12	299.000000	7.450000e+02	47.000000
75%	32104.500000	4.140000	9.780872e+12	416.000000	5.000500e+03	238.000000
max	45641.000000	5.000000	9.790008e+12	6576.000000	4.597666e+06	94265.000000

Fig: 2

From the outcomes above, we can see that our scores are somewhere in the range of 0 and 5. We also get to learn about different sections, for instance, the normal of the mean scores and different information that could end up being useful to us in the subsequent stages.

From the outcomes above, we can see that our scores are somewhere in the range of 0 and 5. We also get to learn about different sections, for instance, the normal of the mean scores and different information that could end up being useful to us in the subsequent stages.

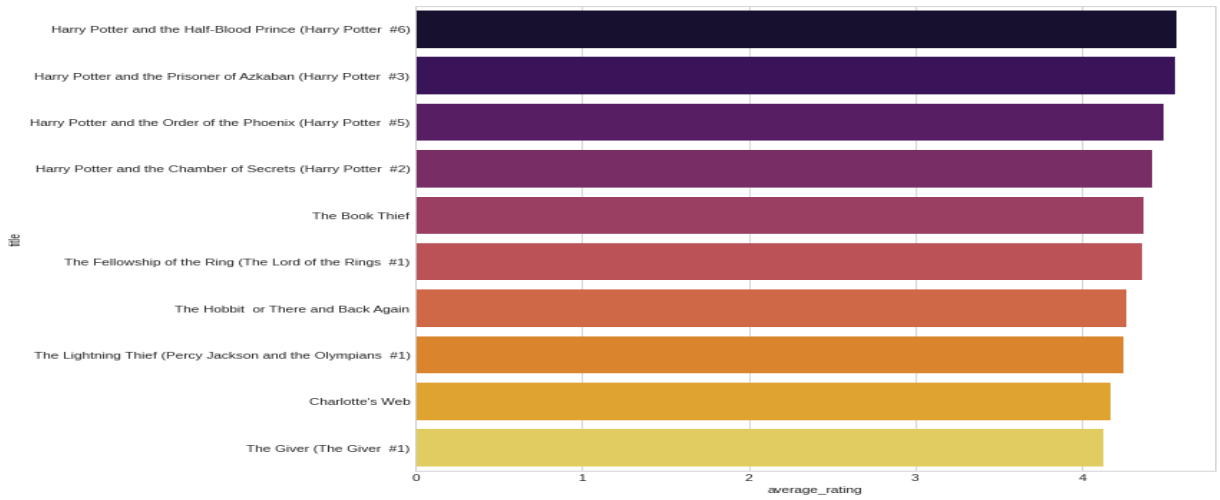


Fig: 3

The outcomes above show us the main 10 books in our information. We saw that the maximum score in our information was 5.0 however we see no books in the above outcome with a score of 5.0. Without a doubt, we have filtered these books as indicated by the quantity of notes. We'll rank them as per the quantity of books they've composed as long as those books are available in the information.

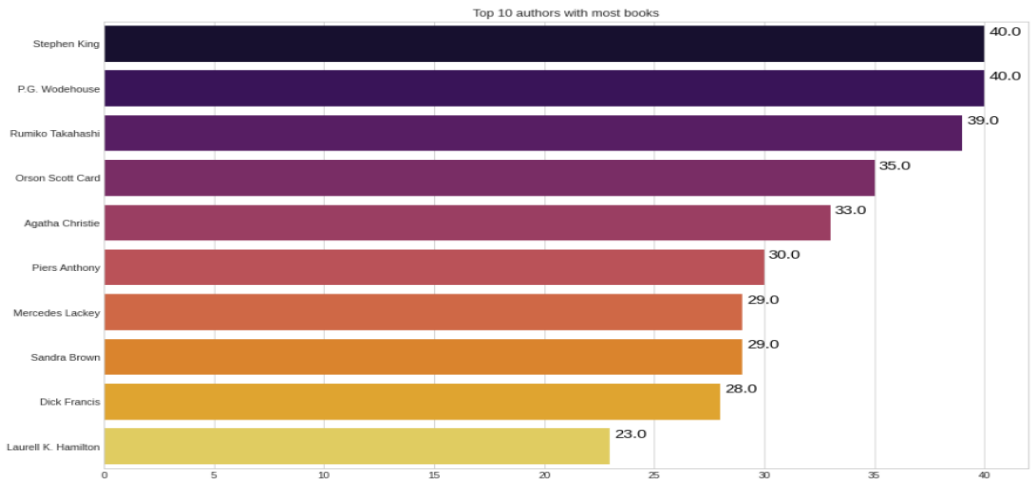
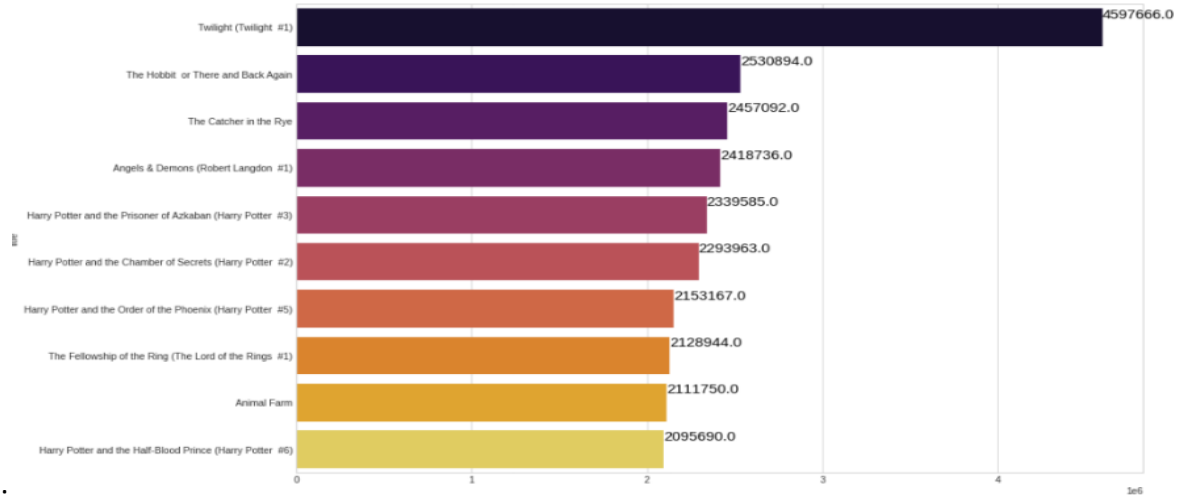


Fig: 4

From the above chart, Stephen K P.G. Wodehouse have the most books in the data. Both authors have 40 books in our dataset followed by Rumiko Takahashi and Orson Scott Card.

'll investigate which books have been reviewed the most.We will attempt to utilize this segment to find the most remarked books present in our information



:

Fig: 5

We can see that Twilight has been rated more times than any other book! Also, these ratings are all in the millions! So that means Twilight has been reviewed over 4 million times, followed by The Hobbit or There and Back Again and The Catcher in the Rye which has been reviewed over 2 million times.track down a relation between our average score and the number of scores.

We will likewise check the dissemination of average scores with the number of pages in a book, the language utilized in the book and the number of text surveys:

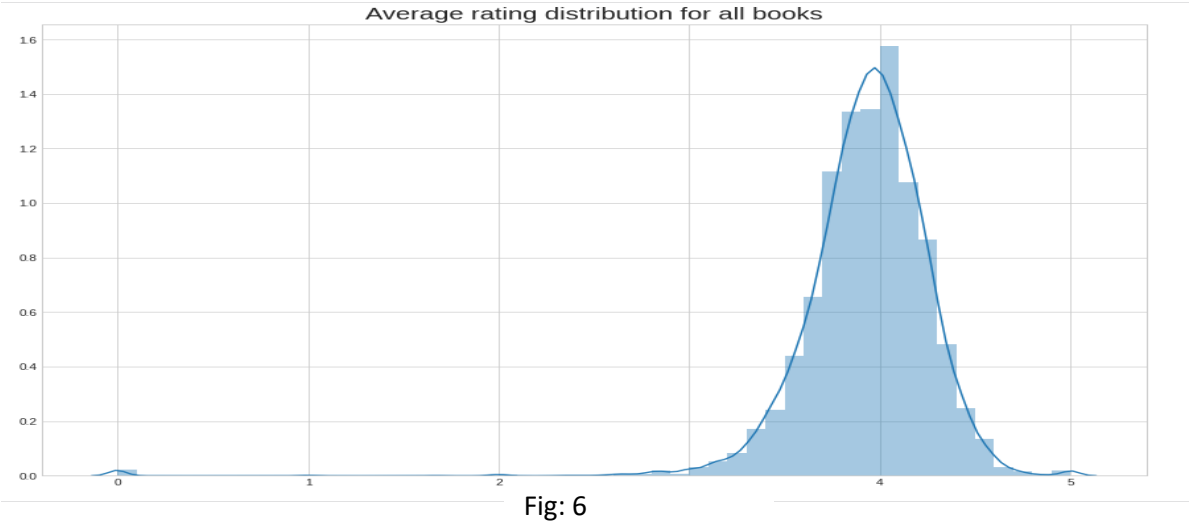


Fig: 6

Final step, We have developed a machine learning model for recommending books and presently we should make a function utilizing Python. At the point when this function is called, we should pass the name of the book to it. The model will attempt to find books in light of the elements. We'll store those book names that the framework suggests in a rundown and return them toward the end.

```
['Harry Potter and the Half-Blood Prince (Harry Potter  #6)',
 'Harry Potter and the Order of the Phoenix (Harry Potter  #5)',
 'The Fellowship of the Ring (The Lord of the Rings  #1)',
 'Harry Potter and the Chamber of Secrets (Harry Potter  #2)',
 'Harry Potter and the Prisoner of Azkaban (Harry Potter  #3)',
 'The Lightning Thief (Percy Jackson and the Olympians  #1)']
```

With this, we reach the finish of a machine learning project on the book proposal framework. As may be obvious, our model shows a really good outcome.

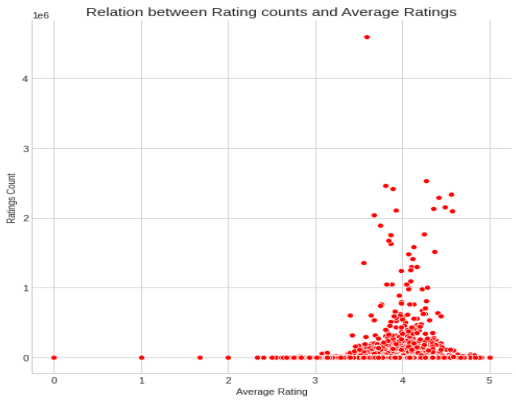


Fig: 7

Conclusion: Recommender systems are ending up being a valuable device that will give idea to client as per their prerequisite. Filtering is utilized to further developed recommendation precision in the first recommender systems. To accomplish this exactness most memory-based methods and algorithms were figured out and upgraded under some situation (e.g., kNN metrics, singular worth decomposition, and so on.). At this stage, to worked on the nature of the recommendations a few half breed approaches are utilized (fundamentally collaborative filtering and content filtering).The heightening requests of Online Data have prompted the innovation of new strategies for introducing or rather prescribing various items to the clients. This paper involves Collaborative filtering also Association data Mining to prescribe various kinds of books to the clients. Both these procedures make a decent Hybrid Recommendation System and furthermore disposes of the Data Sparsity and Cold Start problem. At last, the estimations of both the algorithms give exact outcomes.

5. Patil, Abhay E., et al. "Online book recommendation system using association rule mining and collaborative filtering." *International Journal of Computer Science and Mobile Computing* 8.11 (2019): 83-87.

6. Thorat, Poonam B., Rajeshwari M. Goudar, and Sunita Barve. "Survey on collaborative filtering, content-based filtering and hybrid recommendation system." *International Journal of Computer Applications* 110.4 (2015): 31-36.

References:

1:Rajpurkar, S., Bhatt, D., Malhotra, P., Rajpurkar, M.S.S. and Bhatt, M.D.R., 2015. Book recommendation system. *IJIRST–International Journal for Innovative Research in Science & Technology*, *1*(11), pp.314-316.

2.Kurmashov, N., Latuta, K. and Nussipbekov, A., 2015, September. Online book recommendation system. In *2015 Twelve International Conference on Electronics Computer and Computation (ICECCO)* (pp. 1-4). IEEE.

3.Mathew, P., Kuriakose, B. and Hegde, V., 2016, March. Book Recommendation System through content based and collaborative filtering method. In *2016 International conference on data mining and advanced computing (SAPIENCE)* (pp. 47-52). IEEE.

4. Rana, Avi, and K. Deeba. "Online book recommendation system using collaborative filtering (with Jaccard similarity)." *Journal of Physics: Conference Series*. Vol. 1362. No. 1. IOP Publishing, 2019.