Book Recommendation System

Team :
Harika Satti
Pranathi Krishna
Ravi Teja Reddy
Abdul Vahed Shaik
Naga Bathula

CMPE 255 Fall’ 2022

Project Report Sections

- Book Recommendations - Business Objective
- Data Sources & Understanding of Datasets
- Exploratory Data Analysis
- Data Cleanup
- Recommendation Models
  - Popularity Based
  - Content Based (**CountVectorizer** and **TfidfVectorizer**)
  - Collaborative (SVD and KNN)
- Flask Application Architecture & Screenshots
- Challenges
- Future Scope & Conclusion

Business Objectives (Project Goals)

The main objective is to create a recommendation system to recommend relevant books to users based on popularity and user interests.

Data Sources & Understanding of Datasets

Book-Crossing Dataset Collected by Cai-Nicolas Ziegler from the Book-Crossing community.

Contains 278,858 users providing 1,149,780 ratings (explicit / implicit) about 271,379 Source:

http://www2.informatik.uni-freiburg.de/~cziegler/BX

Book-Crossing dataset comprises of 3 files.

- **Users**
  Contains the users. (`User-ID`) have been anonymized and map to integers. Demographic data is provided (`Location`, `Age`) if available.
- **Books**
  Books are identified by their respective ISBN.Content-based information is given (`Book-Title`, `Book-Author`, `Year-Of-Publication`, `Publisher`). URLs linking to cover images are also given, appearing in three different flavours (`Image-URL-S`, `Image-URL-M`, `Image-URL-L`), i.e., small, medium, large.
- **Ratings**
  Contains the book rating information. Ratings (`Book-Rating`) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0.

Recommender Systems - Background

There are two major approaches to recommendation systems

a) **Content-based** : recommend books to a user that are similar to the ones the user preferred in the past.
b) **Collaborative** : predict users' preferences by analyzing relationships between users and interdependencies among books; and then extrapolate new associations

Another simple type of recommendations is **Popularity Based**
Popularity based recommendation systems are based on the rating of books by all the users.This type works with the current trend of the books which are highly rated.

Exploratory Data Analysis

EDA

The primary goal of EDA is to make data ‘clean’ implying that it should be devoid of redundancies. It aids in identifying incorrect data points so that they may be readily removed and the data cleaned. Furthermore, it aids us in comprehending the relationship between the variables, providing us with a broader view of the data and allowing us to expand on it by leveraging the relationship between the variables. It also aids in the evaluation of the dataset’s statistical measurements.

Outliers or abnormal occurrences in a dataset can have an impact on the accuracy of machine learning models. This dataset contained some missing or duplicate values. EDA is used to eliminate or resolve all of the dataset’s undesirable qualities.

EDA

Performed **Exploratory Data Analysis (EDA)** on all three datasets that employs a variety of techniques to:

- maximize insight into a data set
- uncover underlying structure
- extract important variables
- detect outliers and anomalies

Following are the steps performed in EDA:

- Loaded the data sets.
- Checked the description of the datasets like the data types, number of rows , columns, unique values etc
- Dropped duplicate values
- Replaced outliers
- Merged Books and Ratings data
- Preprocessed data

Data frames shape and head before EDA

```
[ ] ratings = pd.read_csv(r'/content/drive/MyDrive/Data Sets/BX-Book-Ratings.csv', delimiter=';', on_bad_lines='skip', encoding='ISO-8859-1')
    books = pd.read_csv(r'/content/drive/MyDrive/Data Sets/BX-Books.csv', delimiter=';', on_bad_lines='skip', encoding='ISO-8859-1')
    users = pd.read_csv(r'/content/drive/MyDrive/Data Sets/BX-Users.csv', delimiter=';', on_bad_lines='skip', encoding='ISO-8859-1')
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or se
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
print(books.shape)
print(ratings.shape)
print(users.shape)
```

```
(271360, 8)
(1149780, 3)
(278858, 3)
```

+ Code + Text

```
[ ] #Printing columns and head part of books dataframe
    print(books.columns)
    books.head()
```

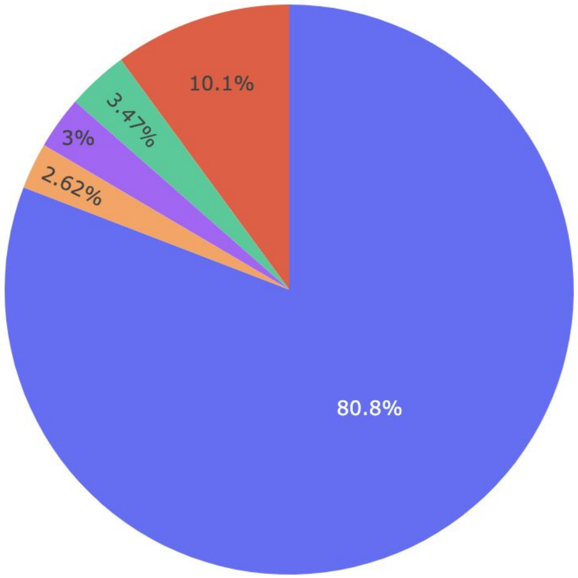
```
Index(['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher',
       'Image-URL-S', 'Image-URL-M', 'Image-URL-L'],
      dtype='object')
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	Image-URL-M	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/images/P/0195153448.0...	http://images.amazor
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/images/P/0002005018.0...	http://images.amazor
2	0060973129	Decision in	Carlo	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/0060973129.0...	http://images.amazor

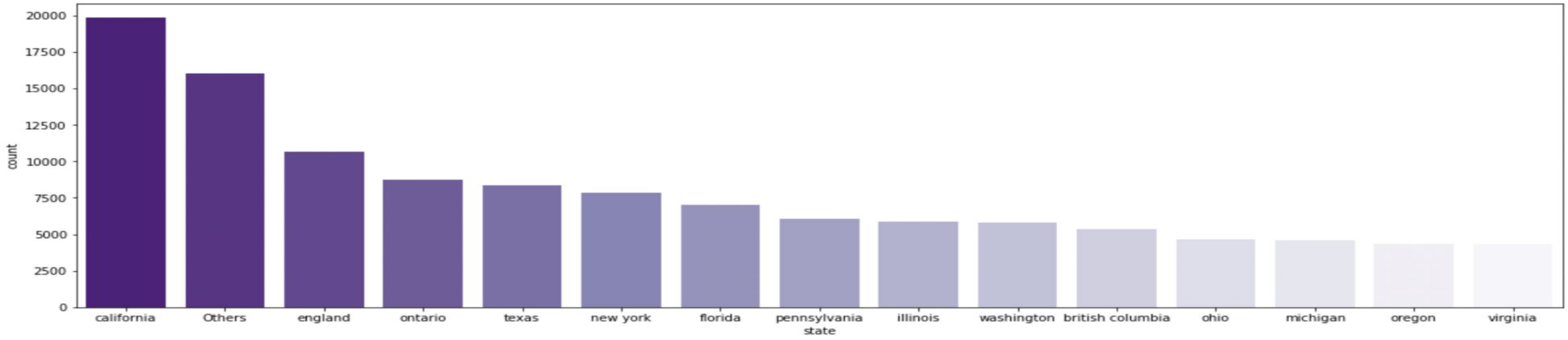
Books Data Preprocessing

```python
#As we don't require the url columns in the books dataframe, we will drop those columns
books = books.drop(['Image-URL-S', 'Image-URL-M', 'Image-URL-L'], axis=1)
books.head()
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company




```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x7f5df5c11310&gt;
```



Data Cleanup

All three datasets are cleaned separately

Books:

- Dropped three Image URL features.
- Checked for the number of null values in each column. Replace these three empty cells with ‘Other’.
- Checked for the unique years of publications.
- Converted the type of the years of publications feature to the integer.
- By keeping the range of valid years as less than 2022 and not 0, replaced all invalid years with the mode of the publications that is 2002.
- Upper-casing all the alphabets present in the ISBN column and removal of duplicate rows from the table

Users:

- Checked null values in the table.
- Checked for unique values present in the Age column. There are many invalid ages present like 0 or 244.
- By keeping the valid age range of readers as 12 to 70 replaced null values and invalid ages in the Age column with the mean of valid ages.
- The location column has 3 values city, state, and country. These are split into 3 different columns named; City, State, and Country respectively. In the case of null value, ‘other’ has been assigned as the entity value.
- Removed duplicate entries from the table.

Ratings:

- Checked for null values in the table.
- Checked for Rating column and User-ID column to be an integer.
- Removed punctuation from ISBN column values.
- Upper-casing all the alphabets present in the ISBN column.
- Removed duplicate entries from the table.

Popularity Based Recommendations

Popularity Based Approach

Step 1 : For Each Book, calculate No of Votes(rating) casted by entire population of users
Step 2 : For Each Book, calculate Average/Mean rating by the entire population of users
Step 3 : Calculate Popularity, which is Book weighted average

Book weighted average formula:

Weighted Rating(WR)=[vR/(v+m)]+[mC/(v+m)]

- *v is the number of votes for the books;*
- *m is the minimum votes required to be listed in the chart; R is the average rating of the book; and*
- *C is the mean vote*

Step 4 : Order the Dataset with highest Popularity

```
def proess_popular_books(self):
    df = self.books1
    rating_count=df.groupby("Book-Title").count()["Book-Rating"].reset_index()
    rating_count.rename(columns={"Book-Rating":"NumberOfVotes"},inplace=True)

    rating_average=df.groupby("Book-Title")["Book-Rating"].mean().reset_index()
    rating_average.rename(columns={"Book-Rating":"AverageRatings"},inplace=True)
    print(rating_average.shape)

    popularBooks=rating_count.merge(rating_average,on="Book-Title")

    def weighted_rate(x):
        v=x["NumberOfVotes"]
        R=x["AverageRatings"]

        return ((v*R) + (m*C)) / (v+m)

    C=popularBooks["AverageRatings"].mean()
    m=popularBooks["NumberOfVotes"].quantile(0.90)

    popularBooks=popularBooks[popularBooks["NumberOfVotes"] >=250]
    popularBooks["Popularity"]=popularBooks.apply(weighted_rate,axis=1)
    popularBooks=popularBooks.sort_values(by="Popularity",ascending=False)
    print(popularBooks.shape)
    self.popular_books = popularBooks
    print('\n\Processed Popular Data')
```

~/Projects/BookRecommendationsApp/templates/home.html

Content Based Recommendations

Content Based Recommendations

Content-based filtering uses Book features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.

Book features used for similarity : Book Title, Author and Publisher

Step 1 : Create a new feature combining ["Book-Title","Book-Author","Publisher"]

Step 2 : Use NLTK and Regex library to cleanup the Data

Step 3 : Create matrix of token counts using **CountVectorizer** and **TfidfVectorizer** (2 Methods)

Step 4 : Calculate Cosine Similarity between each of the Book in the matrix

Content Based Recommendations

```python
data_books = self.books1.groupby('Book-Title').count()['Book-Rating'].reset_index().sort_values('Book-Rating', ascending=False)
data_books.columns = ['Book-Title','Total-Rating']

self.final_data_books = self.books1.merge(data_books, left_on='Book-Title', right_on='Book-Title', how='left')

self.final_data_books = self.final_data_books[self.final_data_books['Total-Rating'] > 70 ].reset_index(drop=True)

def text_preprocessing(sms):
    # removing punctuations
    sms_wo_punct = [x for x in sms if x not in string.punctuation]
    sms_wo_punct = ''.join(sms_wo_punct)

    # making into lower case
    sms_wo_punct = sms_wo_punct.lower()

    return sms_wo_punct

self.final_data_books['Processed-Book'] = self.final_data_books['Book-Title'].apply(text_preprocessing)

## Implementing CountVectorizer
(variable) final_data_books: DataFrame
countvec = CountVectorizer(stop_words='english
(variable) final_data_books: DataFrame
countvec_matrix = countvec.fit_transform(self.final_data_books['Processed-Book'])

self.cosine_mat_count = cosine_similarity(countvec_matrix, countvec_matrix)

## Implementing TF-IDF Vectorizer
tfidf = TfidfVectorizer(ngram_range=(1, 2), min_df = 1, stop_words='english')
tfidf_matrix =  tfidf.fit_transform(self.final_data_books['Processed-Book'])

self.cosine_mat_tfidf = cosine_similarity(tfidf_matrix, tfidf_matrix)

print('\n\nDone Content Based Recommender system')
```

Collaborative Recommendation (Item - User based) models

Singular Value Decomposition(SVD)

SVD is a Matrix Factorization algorithm that decomposes the matrix A into three matrices such that

$$A = UWV^T$$

Where :

- U is a m x n matrix of the orthonormal eigenvectors of AA^T
- V^T is the transpose of a nxn matrix containing the orthonormal eigenvectors of A^TA
- W is a nxn Diagonal matrix of the singular values which are the square roots of the eigenvalues of A^TA

Implementation of SVD using Surprise library

Surprise is a Python scikit based library specially made for building and analyzing recommender systems that deal with explicit rating data.

```
#Choosing surprise SVD Algo
algo = surprise.prediction_algorithms.matrix_factorization.SVD(n_factors=100,n_epochs=9,lr_all=0.01,reg_all=0.1)
```

The Root Mean Squared error for implementing SVD on out dataset is calculated as below

```
#Train the trainset
algo.fit(trainset)
predictions = algo.test(testset)
```

```
#Calculate Prediction accuracy of SVD
accuracy.rmse(predictions)
```

```
RMSE: 0.6987
0.6987454966354412
```

Retrieving the predictions for the users

```python
from collections import defaultdict

def get_top_n(predictions, n=10):

    # First map the predictions to each user
    top_n = defaultdict(list)
    top_n2 = defaultdict(list)

    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))

    # Then sort the predictions for each user and retrieve the k highest ones
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]
        top_n2[uid] = list(zip(*top_n[uid]))[0]


    return top_n, top_n2
```

Predictions for the Users for the books that they haven't rated

2313	My Sister's Keeper : A Novel (Picoult, Jodi)	Weirdos From Another Planet!	Harry Potter and the Chamber of Secrets Postca...	Dilbert: A Book of Postcards	Scientific Progress Goes 'Boink': A Calvin an...	Harry Potter and the Prisoner of Azkaban (Book 3)	Harry Potter and the Sorcerer's Stone (Book 1)	The Calvin & Hobbes Lazy Sunday Book	A Tree Grows in Brooklyn	Griffin & Sabine: An Extraordinary Corresp...
10314	Scientific Progress Goes 'Boink': A Calvin an...	Harry Potter and the Chamber of Secrets Postca...	Dilbert: A Book of Postcards	The Giving Tree	84 Charing Cross Road	Where the Red Fern Grows	It's A Magical World: A Calvin and Hobbes Coll...	The Neverending Story	The Secret Garden	Weirdos From Another Planet!
77480	My Sister's Keeper : A Novel (Picoult, Jodi)	Harry Potter and the Chamber of Secrets Postca...	Dilbert: A Book of Postcards	Scientific Progress Goes 'Boink': A Calvin an...	Weirdos From Another Planet!	Calvin and Hobbes	East of Eden (Oprah's Book Club)	Seabiscuit: An American Legend	Where the Red Fern Grows	The Authoritative Calvin and Hobbes (Calvin an...
98391	Scientific Progress	Harry Potter and the Chamber of	Weirdos From Another	Dilbert: A Book of	Where the Red	The Little	Griffin & Sabine: An	Calvin and	The Authoritative Calvin and	84 Charing

Implementing KNN to find similar books

- K-Nearest Neighbors (KNN) is a type of instance-based learning that can be used for recommendation systems.
- It works by storing a set of items (in this case, books) and their associated features (such as the title, author, genre, etc.). When a new book is introduced, the KNN algorithm compares the features of the new book to the features of the books in the stored set. It then identifies the K number of books that are most similar to the new book, based on the similarity of their features.
- Here, we have implemented KNN to compare the title of a books to find similar books.

```
def collabarative_filtering(self):

    print('\n\nStarted building Collabarative Memory based Approach')

    books_data = self.books1.groupby('Book-Title')['Book-Rating'].count().reset_index().sort_values('Book-Rating', ascending=False).reset_index(drop=True)

    ## Using KNN
    ## Considering only books having atleast 50 ratings
    books_data = books_data[books_data['Book-Rating']>50]

    ## Next we merge the above dataset with the books1 dataset
    self.books_data_final = pd.merge(books_data,self.books1, left_on = 'Book-Title', right_on='Book-Title', how='left')

    #### Building a matrix with users as columns and book title as rows
    self.books_user = self.books_data_final.pivot_table(index='Book-Title', columns='User-ID', values='Book-Rating_y').fillna(0)
    books_user_matrix = csr_matrix(self.books_user)

    ## Now we build with knn neighbours using cosine similarity
    self.model = NearestNeighbors(metric = 'cosine', algorithm = 'brute', p=2)
    self.model.fit(books_user_matrix)
```

```
    temp_data = self.books_data_final[self.books_data_final['Book-Title'] == bookName]
    if len(temp_data) == 0:
        return ["No book found"]

    ## Here I am getting 6 nearest neighbours(includes the entered book as well) for particular book
    distances, indices = self.model.kneighbors(self.books_user.loc[bookName].values.reshape(1, -1), n_neighbors = 6)

    ## Printing all recommended books
    similar_item = []
    print('\n\n\n')
    for i in range(0, len(distances.flatten())):
        if i == 0:
            print('Recommendations for book:',bookName,'\n\n')
        if i > 0:
            print(i,': ',self.books_user.index[indices.flatten()[i]],', with distance of',distances[0][i])
            similar_item.append(self.books_user.index[indices.flatten()[i]])
    print('\n\n\n')
    return similar_item
```

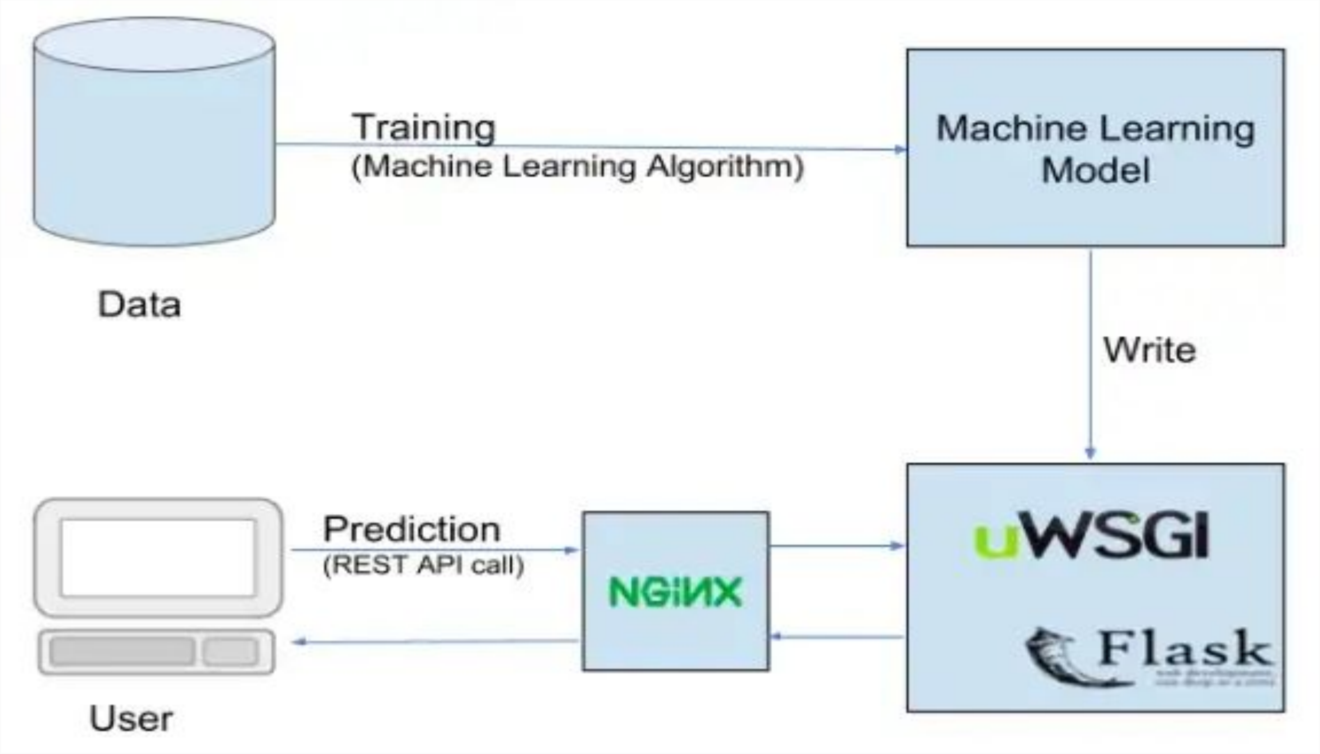
Challenges

Challenges

1. Handling sparsity was a major challenge since the user interaction was not present for majority of books
2. Understanding the metric for evaluation was another challenge
3. Since the data consisted of text data, data cleaning was another challenge.
4. Decision making on missing value imputations and outlier treatment was also quite challenging.

Flask Application & Screenshots

Flask Application Architecture for serving Recommendation models



Book Recommendation System

Data Analysis

Data Visualization

Recommendation Types:

Popularity Based Recommender System
Content Based Recommender System(CountVectorizer)
Content Based Recommender System(Tfidf Vectorizer)
Collaborative Based Recommender System(Item Based - KNN)
Hybrid Based Recommender System(User Item Based - SVD)

Home Page

Book Recommendation System

Popularity Based Recommender System

Enter number of Books you want see by popularity (Top Rated in descending Order)

10

Search

Number of top Books: 10

Popular Books:

Harry Potter and the Prisoner of Azkaban (Book 3)

To Kill a Mockingbird

Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))

Harry Potter and the Chamber of Secrets (Book 2)

Tuesdays with Morrie: An Old Man, a Young Man, and Life's Greatest Lesson

The Secret Life of Bees

The Da Vinci Code

The Lovely Bones: A Novel

The Red Tent (Bestselling Backlist)

Where the Heart Is (Oprah's Book Club (Paperback))

Top 10 Most Popular Books

Book Recommendation System

Content Based Recommender System (Count Vectorizer)

Enter the Book Name:

The Da Vinci Code

Search

Book Name:

Recommended Books:

Book Recommendation System

Content Based Recommender System (Count Vectorizer)

Enter the Book Name:

Search

Book Name: The Da Vinci Code

Recommended Books:

The Brethren

In Her Shoes : A Novel

A Painted House

Jurassic Park

Isle of Dogs

Book Recommendation System

Content Based Recommender System (TF IDF Vectorizer)

Enter the Book Name:

Search

Book Name: The Da Vinci Code

Please find the search results here:

The Brethren

In Her Shoes : A Novel

A Painted House

Jurassic Park

Isle of Dogs

Book Recommendation System

Collaborative Based Recommender System(Item Based - KNN)

Enter the Book Name:

The Da Vinci Code

Search

Book Name: The Da Vinci Code

Please find the search results here:

Angels & Demons

Middlesex: A Novel

Digital Fortress : A Thriller

The Secret Life of Bees

The Lovely Bones: A Novel

Top 5 Book Recommendations for the Book: The Da Vinci Code

Book Recommendation System

Hybrid(User - Item) Based Recommender System

Enter a user id:

2313

Search

User Id: 2313

Please find the search results here:

My Sister's Keeper : A Novel (Picoult, Jodi)

Dilbert: A Book of Postcards

Harry Potter and the Chamber of Secrets Postcard Book

The Amber Spyglass (His Dark Materials, Book 3)

Scientific Progress Goes 'Boink': A Calvin and Hobbes Collection

Weirdos From Another Planet!

The Giving Tree

The Little Prince

Where the Sidewalk Ends : Poems and Drawings

Seabiscuit: An American Legend

Top 10 Book Recommendations for User 2313

Future Scope

Future Scope

1. Implementing a content-filtering based recommendation system (given more information regarding books)
2. Compare the results with the existing collaborative filtering based system
3. Exploring various clustering approaches based on various factors of users
4. Implement voting algorithms to recommend items to users based on the cluster to which they belong

Conclusion

Conclusion

- From EDA we found out that the Top-10 rated books were mostly novels
- Majority of the readers were of the age bracket 20-35 and most of them come from European and North American countries.
- If we look at the ratings distribution most of the books have high ratings with maximum books being rated are 8.
- Ratings below 5 are less in number
- It was observed that for collaborative filtering SVD technique worked way better than NMF with lower Mean Absolute Error (MAE)