Project : Book Rental Recommendation

Description

Book Rent is the largest online and offline book rental chain in India. They provide books of various genres, such as thrillers, mysteries, romances, and science fiction. The company charges a fixed rental fee for a book per month. Lately, the company has been losing its user base. The main reason for this is that users are not able to choose the right books for themselves. The company wants to solve this problem and increase its revenue and profit.

Project Objective

You, as an ML expert, should focus on improving the user experience by personalizing it to the user's needs. You have to model a recommendation engine so that users get recommendations for books based on the behavior of similar users. This will ensure that users are renting the books based on their tastes and traits.

Note: You have to perform user-based collaborative filtering and item-based collaborative filtering.

Dataset description:

> BX-Users: It contains the information of users.

- user_id - These have been anonymized and mapped to integers
- Location - Demographic data is provided
- Age - Demographic data is provided

  If available, otherwise, these fields contain NULL-values.

  > BX-Books:

- isbn - Books are identified by their respective ISBNs. Invalid ISBNs have already been removed from the dataset.
- book_title
- book_author
- year_of_publication
- publisher

  > BX-Book-Ratings: Contains the book rating information.

- user_id
- isbn

- rating - Ratings (`Book-Rating`) are either explicit, expressed on a scale from 1–10 (higher values denoting higher appreciation), or implicit, expressed by 0.

> Note: Download the "BX-Book-Ratings.csv", "BX-Books.csv", "BX-Users.csv", and "Recommend.csv" using the link given in the Book Rental Recommendation project problem statement

> Following operations should be performed:

- Read the books dataset and explore it
- Clean up NaN values
- Read the data where ratings are given by users
- Take a quick look at the number of unique users and books
- Convert ISBN variables to numeric numbers in the correct order
- Convert the user_id variable to numeric numbers in the correct order
- Convert both user_id and ISBN to the ordered list, i.e., from 0...n-1
- Re-index the columns to build a matrix
- Split your data into two sets (training and testing)
- Make predictions based on user and item variables
- Use RMSE to evaluate the predictions

In [1]:
```
# Importing the required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read the books dataset and explore it

In [2]:
```
books=pd.read_csv('BX-Books.csv',low_memory= False,encoding='ISO-8859-1')
```

In [3]:
```
#import chardet
#with open('BX-Books.csv', 'rb') as rawdata:
   # result = chardet.detect(rawdata.read(100000))
#result
```

In [4]:
```
books.head()
```

Out[4]:

	isbn	book_title	book_author	year_of_publication	publisher
0	195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
1	2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada
2	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial
3	374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux
4	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company

In [5]:
```
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271379 entries, 0 to 271378
Data columns (total 5 columns):
 #   Column               Non-Null Count   Dtype 
---  ------               --------------   ----- 
 0   isbn                 271379 non-null  object
 1   book_title           271379 non-null  object
 2   book_author          271378 non-null  object
 3   year_of_publication  271379 non-null  object
 4   publisher            271377 non-null  object
dtypes: object(5)
memory usage: 10.4+ MB
```

In [6]:
```
books.isna().sum()
```

Out[6]:
```
isbn                 0
book_title           0
book_author          1
year_of_publication  0
publisher            2
dtype: int64
```

Clean up NaN values

In [7]:
```
# droping the NaN values from books dataset
books.dropna(inplace=True)
```

In [8]:
```
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 271376 entries, 0 to 271378
Data columns (total 5 columns):
 #   Column               Non-Null Count   Dtype 
---  ------               --------------   ----- 
 0   isbn                 271376 non-null  object
 1   book_title           271376 non-null  object
 2   book_author          271376 non-null  object
 3   year_of_publication  271376 non-null  object
 4   publisher            271376 non-null  object
dtypes: object(5)
memory usage: 12.4+ MB
```

Read the data where ratings are given by users

In [9]:
```
book_ratings=pd.read_csv('BX-Book-Ratings.csv',low_memory= False, encoding='ISO-88
```

In [10]:
```
book_ratings.head()
```

Out[10]:

	user_id	isbn	rating
0	276725	034545104X	0
1	276726	155061224	5
2	276727	446520802	0
3	276729	052165615X	3
4	276729	521795028	6

In [11]:
```
book_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
 #   Column   Non-Null Count    Dtype
---  ------   --------------    -----
 0   user_id  1048575 non-null  int64
 1   isbn     1048575 non-null  object
 2   rating   1048575 non-null  int64
dtypes: int64(2), object(1)
memory usage: 24.0+ MB
```

Check for Duplicates

In [12]:
```
books[books.duplicated()]
```

Out[12]:

	isbn	book_title	book_author	year_of_publication	publisher

In [13]:
```
book_ratings[book_ratings.duplicated()]
```

Out[13]:

	user_id	isbn	rating
11338	709	9.78E+12	9
21604	4334	6.31E+11	0
28622	6575	6.31E+11	0
58876	11676	9.78E+12	9
58877	11676	9.78E+12	9
...	...	...	...
1047765	250634	9.78E+12	0
1047766	250634	9.78E+12	10
1047767	250634	9.78E+12	10
1047768	250634	9.78E+12	10
1047769	250634	9.78E+12	0

146 rows × 3 columns

In [14]:
```
# drop the duplicates form book_rating dataset
book_ratings.drop_duplicates(inplace=True)
```

In [15]:
```
book_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1048429 entries, 0 to 1048574
Data columns (total 3 columns):
 #   Column   Non-Null Count    Dtype
---  ------   --------------    -----
 0   user_id  1048429 non-null  int64
 1   isbn     1048429 non-null  object
 2   rating   1048429 non-null  int64
dtypes: int64(2), object(1)
memory usage: 32.0+ MB
```

Take a quick look at the number of unique users and books

In [16]:
```
# Number of unique users
book_ratings.user_id.nunique()
```

Out[16]:
```
95513
```

In [17]:
```
# Number of unique books
book_ratings.isbn.nunique()
```

Out[17]:
```
322102
```

In [18]:
```
book_ratings=pd.merge(book_ratings,books,on='isbn')
```

In [19]:
```
book_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 941138 entries, 0 to 941137
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   user_id              941138 non-null  int64
 1   isbn                 941138 non-null  object
 2   rating               941138 non-null  int64
 3   book_title           941138 non-null  object
 4   book_author          941138 non-null  object
 5   year_of_publication  941138 non-null  object
 6   publisher            941138 non-null  object
dtypes: int64(2), object(5)
memory usage: 57.4+ MB
```

Convert ISBN variables to numeric numbers in the correct order

In [20]:
```
# Import label encoder
from sklearn.preprocessing import LabelEncoder
```

In [21]:
```
le=LabelEncoder()
book_ratings['book_code']=le.fit_transform(book_ratings['isbn'])
```

Convert the user_id variable to numeric numbers in the correct order

In [22]:
```
book_ratings['user_code']=le.fit_transform(book_ratings['user_id'])
```

Convert both user_id and ISBN to the ordered list, i.e., from 0...n-1

In [23]:
```
book_ratings.sort_values(by=['user_code','book_code'],inplace=True)
```

Re-index the columns to build a matrix

In [24]:
```
book_ratings.reset_index()
```

Out[24]:

	index	user_id	isbn	rating	book_title	book_author	year_of_publication	
0	169263	2	195153448	0	Classical Mythology	Mark P. O. Morford	2002	Unive
1	169330	8	074322678X	5	Where You'll Find Me: And Other Stories	Ann Beattie	2002	
2	169337	8	080652121X	0	Hitler's Secret Bankers: The Myth of Swiss Neu...	Adam Lebor	2000	Cit
3	169340	8	1552041778	5	Jane Doe	R. J. Kaiser	1999	N
4	169341	8	1558746218	0	A Second Chicken Soup for the Woman's Soul (Ch...	Jack Canfield	1998	Comm
...	...	...	...	...	...	...	...	
941133	169230	278854	425163393	7	Kat Scratch Fever (Kat Colorado Mysteries)	Karen Kijewski	1998	
941134	158327	278854	515087122	0	The Cat Who Ate Danish Modern (Cat Who... (Pap...	Lilian Jackson Braun	1990	J
941135	169245	278854	553275739	6	In Her Day	Rita Mae Brown	1988	Bant
941136	169256	278854	553578596	0	Wicked Fix : A Home Repair is Homicide Mystery...	SARAH GRAVES	2000	
941137	65208	278854	553579606	8	Ashes to Ashes	TAMI HOAG	2000	

941138 rows × 10 columns

In [25]:
```
book_ratings
```

Out[25]:

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher
169263	2	195153448	0	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
169330	8	074322678X	5	Where You'll Find Me: And Other Stories	Ann Beattie	2002	Scribner
169337	8	080652121X	0	Hitler's Secret Bankers: The Myth of Swiss Neu...	Adam Lebor	2000	Citadel Press
169340	8	1552041778	5	Jane Doe	R. J. Kaiser	1999	Mira Books
169341	8	1558746218	0	A Second Chicken Soup for the Woman's Soul (Ch...	Jack Canfield	1998	Health Communications
...	...	...	...	...	...	...	...
169230	278854	425163393	7	Kat Scratch Fever (Kat Colorado Mysteries)	Karen Kijewski	1998	Berkley Publishing Group
158327	278854	515087122	0	The Cat Who Ate Danish Modern (Cat Who... (Pap...	Lilian Jackson Braun	1990	Jove Books
169245	278854	553275739	6	In Her Day	Rita Mae Brown	1988	Bantam Books
169256	278854	553578596	0	Wicked Fix : A Home Repair is Homicide Mystery...	SARAH GRAVES	2000	Bantam
65208	278854	553579606	8	Ashes to Ashes	TAMI HOAG	2000	Bantam

941138 rows × 9 columns

In [26]:

```
# filtered the popular books with atleast 50 ratings
x=book_ratings.groupby('book_code').count()['rating']>=50
famous_books=x[x].index
filter_book_ratings=book_ratings[book_ratings['book_code'].isin(famous_books)]
filter_book_ratings
```

Out[26]:

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher
64069	9	440234743	0	The Testament	John Grisham	1999	Dell
79886	9	452264464	6	Beloved (Plume Contemporary Fiction)	Toni Morrison	1994	Plume
8792	14	971880107	0	Wild Animus	Rich Shapero	2004	Too Far
94192	16	345402871	9	Airframe	Michael Crichton	1997	Ballantine Books
94605	16	345417623	0	Timeline	MICHAEL CRICHTON	2000	Ballantine Books
...	...	...	...	...	...	...	...
169122	278851	1558531025	8	Life's Little Instruction Book (Life's Little ...	H. Jackson Brown	1991	Thomas Nelson
60953	278854	042516098X	7	Hornet's Nest	Patricia Daniels Cornwell	1998	Berkley Publishing Group
162361	278854	375703063	7	A Virtuous Woman (Oprah's Book Club (Paperback))	Kaye Gibbons	1997	Vintage Books
158327	278854	515087122	0	The Cat Who Ate Danish Modern (Cat Who... (Pap...	Lilian Jackson Braun	1990	Jove Books
65208	278854	553579606	8	Ashes to Ashes	TAMI HOAG	2000	Bantam

201945 rows × 9 columns

Split your data into two sets (training and testing)

In [27]:
```
# Dropped the unneccesary columns and assigned the dataset to new variable
book_ratings_conv=filter_book_ratings[['user_code','book_code','rating']]
```

In [28]:
```
book_ratings_conv
```

Out[28]:

	user_code	book_code	rating
64069	2	126701	0
79886	2	141597	6
8792	5	256888	0
94192	6	84427	9
94605	6	84679	0
...	...	...	...
169122	83641	29398	8
60953	83643	6586	7
162361	83643	103200	7
158327	83643	145589	0
65208	83643	157944	8

201945 rows × 3 columns

In [29]:
```
# Import the train test split model
from sklearn.model_selection import train_test_split
train_data,test_data=train_test_split(book_ratings_conv,test_size=0.3)
```

In [30]:
```
train_data.shape,test_data.shape
```

Out[30]:
```
((141361, 3), (60584, 3))
```

Make predictions based on user and item variablest

In [31]:
```
# prepare a array of matrix book_title vs user_id
data_matrix=filter_book_ratings.pivot_table(index='book_title',columns='user_id',v
```

In [32]:
```
data_matrix.fillna(0,inplace=True)
```

In [33]:
```
data_matrix
```

Out[33]:

user_id	9	14	16	17	26	32	39	42	44	51	...	278813	278819	278828	278832
book_title															
16 Lighthouse Road	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1984	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1st to Die: A Novel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2010: Odyssey Two	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
204 Rosewood Lane	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Year of Wonders	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
You Belong To Me	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
Zen and the Art of Motorcycle Maintenance: An Inquiry into Values	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
Zoya	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
\O\" Is for Outlaw"	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

1735 rows × 37739 columns

In [34]:
```
# Using Cosine_simislarity obtain the distances of nearest neighbhour
from sklearn.metrics.pairwise import cosine_similarity
similarity_scores=cosine_similarity(data_matrix)
```

In [35]:
```
similarity_scores.shape
```

Out[35]:
```
(1735, 1735)
```

In [36]:
```
# defined the function for recommendation engine
def recommendation(book_name):
    index=np.where(data_matrix.index==book_name)[0][0] # to obtain the index of th
    # to obtain the top 10 similarity scores for the selected book
    similar_items=sorted(list(enumerate(similarity_scores[index])),key = lambda x:

    for i in similar_items:
        print(data_matrix.index[i[0]])
```

In [37]:
```
# testing the recommendation engine
recommendation('Zoya')
```

```
Kaleidoscope
Secrets
Special Delivery: A Novel
Star
No Greater Love
Fine Things
Heartbeat
Morning, Noon &amp;amp; Night
The Ranch
```