BookRent is the largest online and offline book rental chain in India. 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.

Objective: You, as an ML expert, 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 books based on their individual tastes.

Actions to Perform:

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 to numeric numbers in the correct order. Do the same for user_id. Convert it into numeric order. Convert both user_id and ISBN to the ordered list i.e. from 0...n-1. Re-index columns to build matrix later on. Split your data into two sets (training and testing). Calculate the cosine similarity. Use the evaluation metrics to make predictions.

The data use used is from http://www2.informatik.uni-freiburg.de/~cziegler/BX/

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df_user =pd.read_csv("BX-Users.csv",encoding='latin-1')
```

```
C:\Users\ctoqu\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3146: DtypeWarning: Columns (0) have mixed ty
pes.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [3]:
```
df_user.head()
```

Out[3]:

	user_id	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN

In [4]:
```
df_user.isna().any()
```

Out[4]:
```
user_id     False
Location     True
Age          True
dtype: bool
```

In [5]:
```
median = df_user['Age'].median()
df_user['Age'].fillna(median, inplace=True)
```

In [6]:
```
df_user.head()
```

Out[6]:

	user_id	Location	Age
0	1	nyc, new york, usa	32.0
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	32.0
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	32.0

Reading and exploring books data

In [7]:
```
#Column_names = ['isbn','book_title']

df_books = pd.read_csv('BX-Books.csv', encoding='latin-1')
```

```
C:\Users\ctoqu\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3146: DtypeWarning: Columns (3) have mixed ty
pes.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [8]:
```
df_books.head()
```

Out[8]:

	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 [9]:
```
df_books.describe()
```

Out[9]:

	isbn	book_title	book_author	year_of_publication	publisher
count	271379	271379	271378	271379	271377
unique	271379	242150	102042	202	16823
top	1879591022	Selected Poems	Agatha Christie	2002	Harlequin
freq	1	27	632	17145	7535

Now read the data where the ratings are given by users. You will read only first 10K to avoid Out of memory problem

In [10]:
```
df =pd.read_csv('BX-Book-Ratings.csv', encoding='latin-1',nrows=10000)
```

In [11]:
```
df.head()
```

Out[11]:

	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 [12]:
```
df.describe()
```

Out[12]:

	user_id	rating
count	10000.000000	10000.000000
mean	265844.379600	1.974700
std	56937.189618	3.424884
min	2.000000	0.000000
25%	277478.000000	0.000000
50%	278418.000000	0.000000

	user_id	rating
75%	278418.000000	4.000000
max	278854.000000	10.000000

Merge dataframes. For all practical purposes User Master Data is not required. So, Ignore dataframe df_user

In [13]:
```
df = pd.merge(df,df_books,on='isbn')
df.head()
```

Out[13]:

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
1	276726	155061224	5	Rites of Passage	Judith Rae	2001	Heinle
2	276727	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books
3	278418	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books
4	276729	052165615X	3	Help!: Level 1	Philip Prowse	1999	Cambridge University Press

In [14]:
```
#Checking for the number of unique users and books

n_users = df.user_id.nunique()
n_books = df.isbn.nunique()

print('Num. of Users: '+str(n_users))
print('Num. of books: '+str(n_books))
```

```
Num. of Users: 828
Num. of books: 8051
```

In [15]:
```
isbn_list = df.isbn.unique()
print('Lenght of isbn list: ', len(isbn_list))
def get_isbn_numeric_id(isbn):
    #print("  isbn is:" ,isbn)
    itemindex = np.where(isbn_list==isbn)
    return itemindex[0][0]
```

```
Lenght of isbn list:  8051
```

In [16]:
```
userid_list = df.user_id.unique()
print("Length of user_id List:", len(userid_list))
def get_user_id_numeric_id(user_id):
```

```
    itemindex = np.where(userid_list==user_id)
    return itemindex[0][0]
```

```
Length of user_id List: 828
```

Convert both user_id and isbn to ordered list i.e. from 0 ...n-1

In [17]:
```
df['user_id_order']= df['user_id'].apply(get_user_id_numeric_id)
```

In [18]:
```
df['isbn_id'] = df['isbn'].apply(get_isbn_numeric_id)
df.head()
```

Out[18]:

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher	user_id_order	isbn_id
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	0	0
1	276726	155061224	5	Rites of Passage	Judith Rae	2001	Heinle	1	1
2	276727	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books	2	2
3	278418	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books	3	2
4	276729	052165615X	3	Help!: Level 1	Philip Prowse	1999	Cambridge University Press	4	3

Re-index columns to build matrix later on

In [19]:
```
new_col_order = ['user_id_order', 'isbn_id', 'rating','book_title','book_author','year_of_publication','publisher','isbn'
df = df.reindex(columns= new_col_order)
df.head()
```

Out[19]:

	user_id_order	isbn_id	rating	book_title	book_author	year_of_publication	publisher	isbn	user_id
0	0	0	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	034545104X	276725
1	1	1	5	Rites of Passage	Judith Rae	2001	Heinle	155061224	276726
2	2	2	0	The Notebook	Nicholas Sparks	1996	Warner Books	446520802	276727
3	3	2	0	The Notebook	Nicholas Sparks	1996	Warner Books	446520802	278418
4	4	3	3	Help!: Level 1	Philip Prowse	1999	Cambridge University Press	052165615X	276729

Train Test Split

Recommendation Systems, due to the difficulty to be eveluated, we split the data in two sets but do not perform the classic X_train,X_test,y_train,y_test split. Instead, we just segment the data into two sets of data

In [20]:
```
from sklearn.model_selection import train_test_split
train_data, test_data = train_test_split(df, test_size=0.30)
```

Approach: You Will Use Memory-Based Collaborative Filtering

Memory-Based Collaborative Filtering approaches can be divided into two main sections: user-item filtering and item-item filtering

Item-Item Collaborative Filtering: “Users who liked this item also liked …” User-Item Collaborative Filtering: “Users who are similar to you also liked …”

In both cases we create a matrix built from the entire dataset

The training matrix contains 70% of the raitings and the testing 30% of the ratings

In [23]:
```
#Create two user-book matrices, one for training and another for testing

train_data_matrix = np.zeros((n_users, n_books))
for line in train_data.itertuples():
    train_data_matrix[line[1]-1, line[2]-1]= line[3]

test_data_matrix = np.zeros((n_users, n_books))
for line in test_data.itertuples():
    test_data_matrix[line[1]-1, line[2]-1]= line[3]
```

Now we use pairwise_distances function from sklearn to calculate the cosine similarity. The output will range from 0 to 1 since the rating are all positive

In [26]:
```
from sklearn.metrics.pairwise import pairwise_distances
user_similarity = pairwise_distances(train_data_matrix, metric='cosine')
item_similarity = pairwise_distances(train_data_matrix.T, metric='cosine')
```

In [27]:
```
user_similarity
```

Out[27]:
```
array([[0., 1., 1., ..., 1., 1., 1.],
       [1., 0., 1., ..., 1., 1., 1.],
       [1., 1., 0., ..., 1., 1., 1.],
       ...,
       [1., 1., 1., ..., 0., 1., 1.],
```

```
       [1., 1., 1., ..., 1., 0., 1.],
       [1., 1., 1., ..., 1., 1., 0.]])
```

Next, Predictions

In [28]:
```python
def predict(ratings,similarity, type='user'):
    if type =='user':
        mean_user_rating = ratings.mean(axis=1)
        #You use np.newaxis so that mean_user_rating has same format as ratings
        ratings_diff = (ratings - mean_user_rating[:,np.newaxis])
        pred = mean_user_rating[:, np.newaxis] + similarity.dot(ratings_diff) / np.array([np.abs(similarity).sum(axis=1)]
    elif type =='item':
        pred = ratings.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
    return pred
```

In [29]:
```python
item_prediction = predict(train_data_matrix, item_similarity, type='item')
user_prediction = predict(train_data_matrix, user_similarity, type='user')
```

Evaluation

The evaluation metric we will use is Root Mean Square Error (RMSE)

Since we only want to consider predicted ratings that are in the test dataset, you filter out all other elements in the prediction matrix with: prediction[ground_truth.nonzero()]

In [31]:
```python
from sklearn.metrics import mean_squared_error
from math import sqrt
def rmse(prediction,ground_truth):
    prediction = prediction[ground_truth.nonzero()].flatten()
    ground_truth = ground_truth[ground_truth.nonzero()].flatten()
    return sqrt(mean_squared_error(prediction,ground_truth))
```

In [32]:
```python
print('User-based CF RMSE: ' +str(rmse(user_prediction, test_data_matrix)))
print('Item-based CF RMSE: ' +str(rmse(item_prediction, test_data_matrix)))
```

```
User-based CF RMSE: 7.676789006692209
Item-based CF RMSE: 7.676234404874078
```

Both give almosth the same result

In [ ]: