Book Rental Recommendation

April 2, 2023

[1]:
```
#Import required libraries
import numpy as np
import pandas as pd
```

[2]:
```
#Reading the data
df_user = pd.read_csv('BX-Users.csv',encoding='latin-1')
```

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

[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

[4]:
```
df_user.isnull().any()
```

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

[5]:
```
df_user.isnull().any()
```

[5]:
```
user_id     False
Location     True
Age          True
dtype: bool
```

[7]:
```
df_user1=df_user.dropna()
```

[8]:
```
df_user1.isnull().any()
```

[8]:
```
user_id     False
Location    False
Age         False
```

```
dtype: bool
```

[12]:
```
df_books = pd.read_csv('BX-Books.csv', encoding='Latin')
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3063:
DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or set
low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

[14]:
```
df_books.head()
```

[14]:

	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

[15]:
```
df = pd.read_csv('BX-Book-Ratings.csv',encoding='latin-1',nrows=10000)
```

[16]:
```
df.head()
```

[16]:

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

[17]:
```
df.describe()
```

[17]:

	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
75%	278418.000000	4.000000
max	278854.000000	10.000000

[18]:
```
df = pd.merge(df,df_books,on='isbn')
df.head()
```

[18]:

	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

[19]:
```
#Code for checking 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
```

[20]:
```
#Convert and print length of isbn list
isbn_list = df.isbn.unique()
print(" Length 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]
```

```
 Length of isbn List: 8051
```

[21]:
```
#Convert and print length of user_id list
userid_list = df.user_id.unique()
print(" Length of user_id List:", len(userid_list))
def get_user_id_numeric_id(user_id):
    #print ("  isbn is:" , isbn)
    itemindex = np.where(userid_list==user_id)
    return itemindex[0][0]
```

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

[22]:
```
df['user_id_order'] = df['user_id'].apply(get_user_id_numeric_id)
```

[23]:
```
df['isbn_id'] = df['isbn'].apply(get_isbn_numeric_id)
df.head()
```

[23]:

	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

[24]:
```
#Reindexing the columns
new_col_order = ['user_id_order', 'isbn_id', 'rating', 'book_title',␣
 ↪'book_author','year_of_publication','publisher','isbn','user_id']
df = df.reindex(columns= new_col_order)
df.head()
```

[24]:

	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

[25]:
```
#Importing train_test_split model for splittig the data into train and test set
from sklearn.model_selection import train_test_split
train_data, test_data = train_test_split(df, test_size=0.30)
```

[26]:
```
#Create user-book matrix for training
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]

#Create user-book matrix for testing
```

```
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]
```

[27]:
```
#Importing pairwise_distances function
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')
```

[28]:
```
user_similarity
```

[28]:
```
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.]])
```

[29]:
```
#Defining custom function to make predictions
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)]).T
    elif type == 'item':
        pred = ratings.dot(similarity) / np.array([np.abs(similarity).
 ↪sum(axis=1)])
    return pred
```

[30]:
```
item_prediction = predict(train_data_matrix, item_similarity, type='item')
user_prediction = predict(train_data_matrix, user_similarity, type='user')
```

[31]:
```
#Importing RMSE function
from sklearn.metrics import mean_squared_error
from math import sqrt

#Defining custom function to filter out elements with ground_truth.nonzero
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))
```

[32]:
```
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: 7.571145377216248
Item-based CF RMSE: 7.570417466503438
```

[ ]: