PG AIML - Machine Learning

Course-end 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.

Problem Statement:
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.

Domain: Technology

Dataset description:
Dataset name: **BX-Users.csv, BX-Books.csv, BX-Book-Ratings.csv**

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.

Steps to perform:

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

Solution

Import Libraries

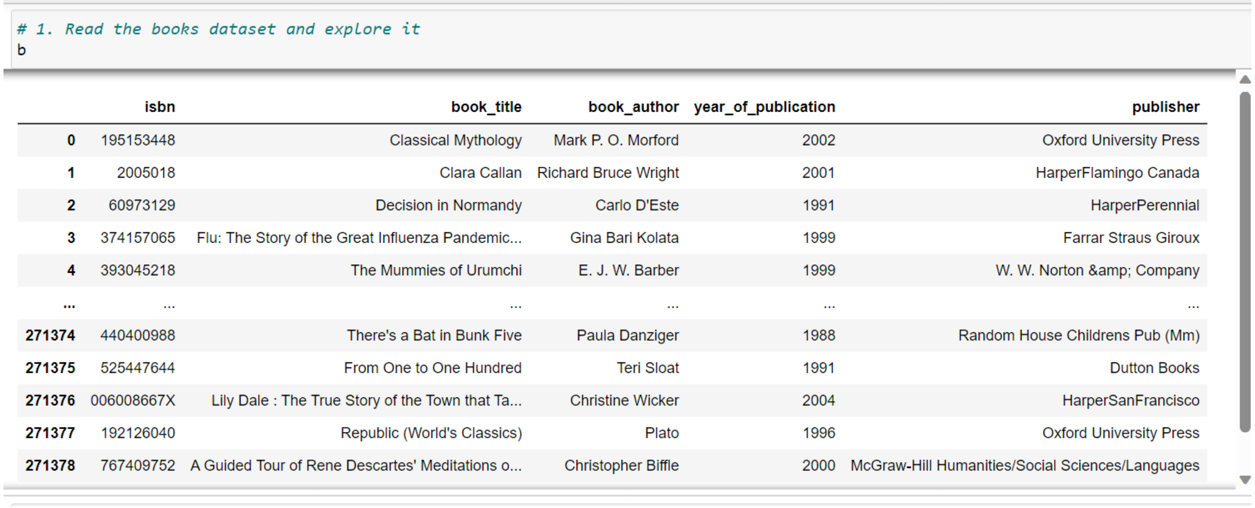
```
import os
import pandas as pd
import numpy as np
```

Load the data file

```
for each in os.listdir():
    print(each)

rating = "BX-Book-Ratings.csv"
book = "BX-Books.csv"
users = "BX-Users.csv"

r = pd.read_csv(rating,encoding="latin-1")
b = pd.read_csv(book,encoding="latin-1")
u = pd.read_csv(users,encoding="latin-1")
```

1. Read the books dataset and explore it

```
b = pd.read_csv(book,encoding="latin-1")
```



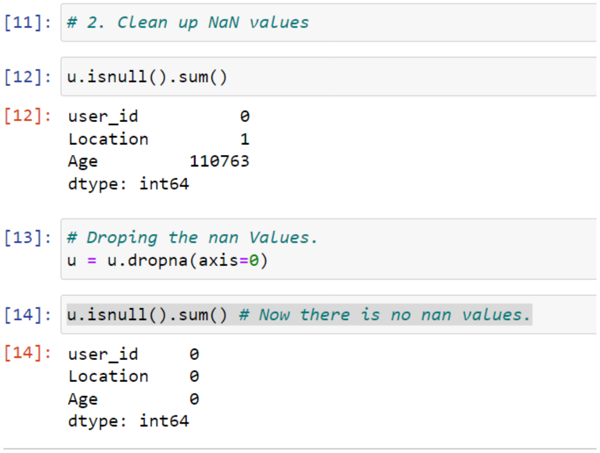
```
# 1. Read the books dataset and explore it
b
```

	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 &amp; Company
...	...	...	...	...	...
271374	440400988	There's a Bat in Bunk Five	Paula Danziger	1988	Random House Childrens Pub (Mm)
271375	525447644	From One to One Hundred	Teri Sloat	1991	Dutton Books
271376	006008667X	Lily Dale : The True Story of the Town that Ta...	Christine Wicker	2004	HarperSanFrancisco
271377	192126040	Republic (World's Classics)	Plato	1996	Oxford University Press
271378	767409752	A Guided Tour of Rene Descartes' Meditations o...	Christopher Biffle	2000	McGraw-Hill Humanities/Social Sciences/Languages

```
b.shape

b.info()

b.describe().T
```

2. Clean up NaN values

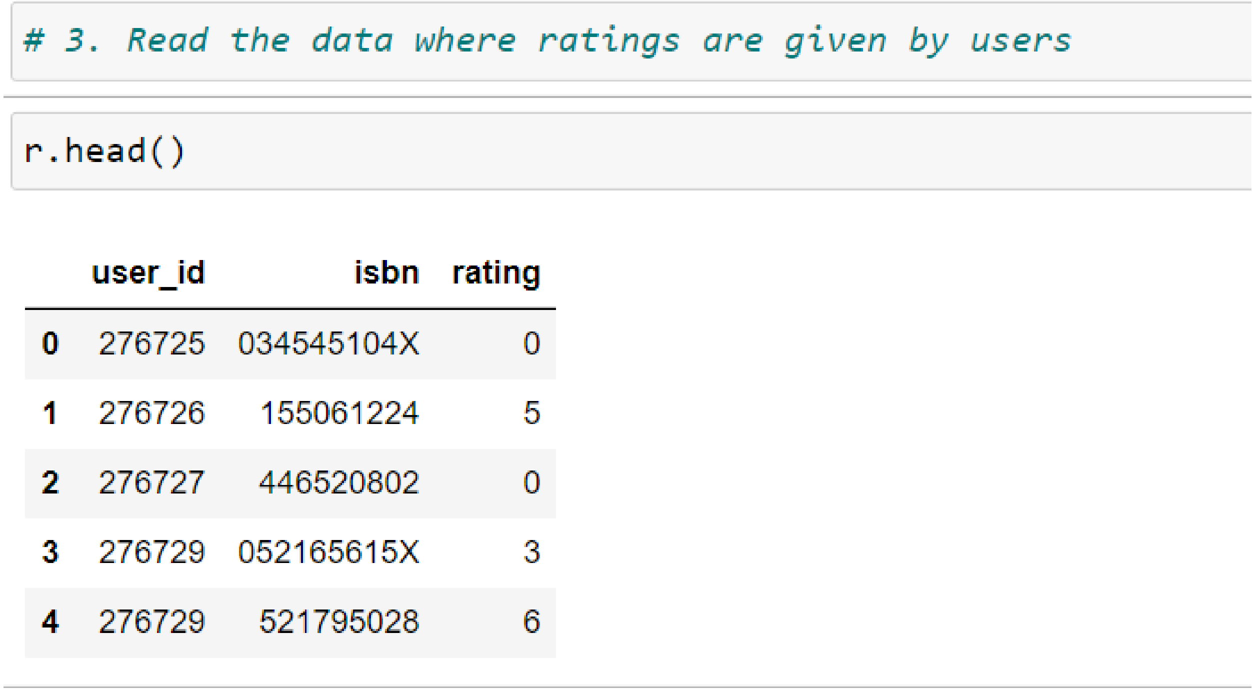
```
u.isnull().sum()

# Droping the nan Values.
u = u.dropna(axis=0)

u.isnull().sum() # Now there is no nan values.
```



```
[11]: # 2. Clean up NaN values

[12]: u.isnull().sum()

[12]: user_id          0
      Location         1
      Age         110763
      dtype: int64

[13]: # Droping the nan Values.
      u = u.dropna(axis=0)

[14]: u.isnull().sum() # Now there is no nan values.

[14]: user_id     0
      Location    0
      Age         0
      dtype: int64
```

3. Read the data where ratings are given by users

```
r.head()
```



```
# 3. Read the data where ratings are given by users
```

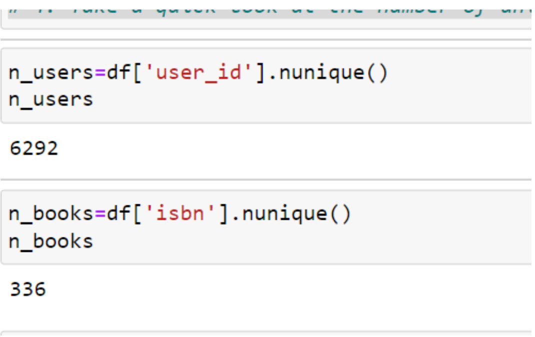
```
r.head()
```

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

```
# merging the two data set books and rating.

df = pd.merge(r,b,on='isbn')
df=df.head(10000)
```

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

```
n_users=df['user_id'].nunique()
n_users

n_books=df['isbn'].nunique()
n_books
```



```
# 4. Take a quick look at the number of uni
```

```
n_users=df['user_id'].nunique()
n_users
```

```
6292
```

```
n_books=df['isbn'].nunique()
n_books
```

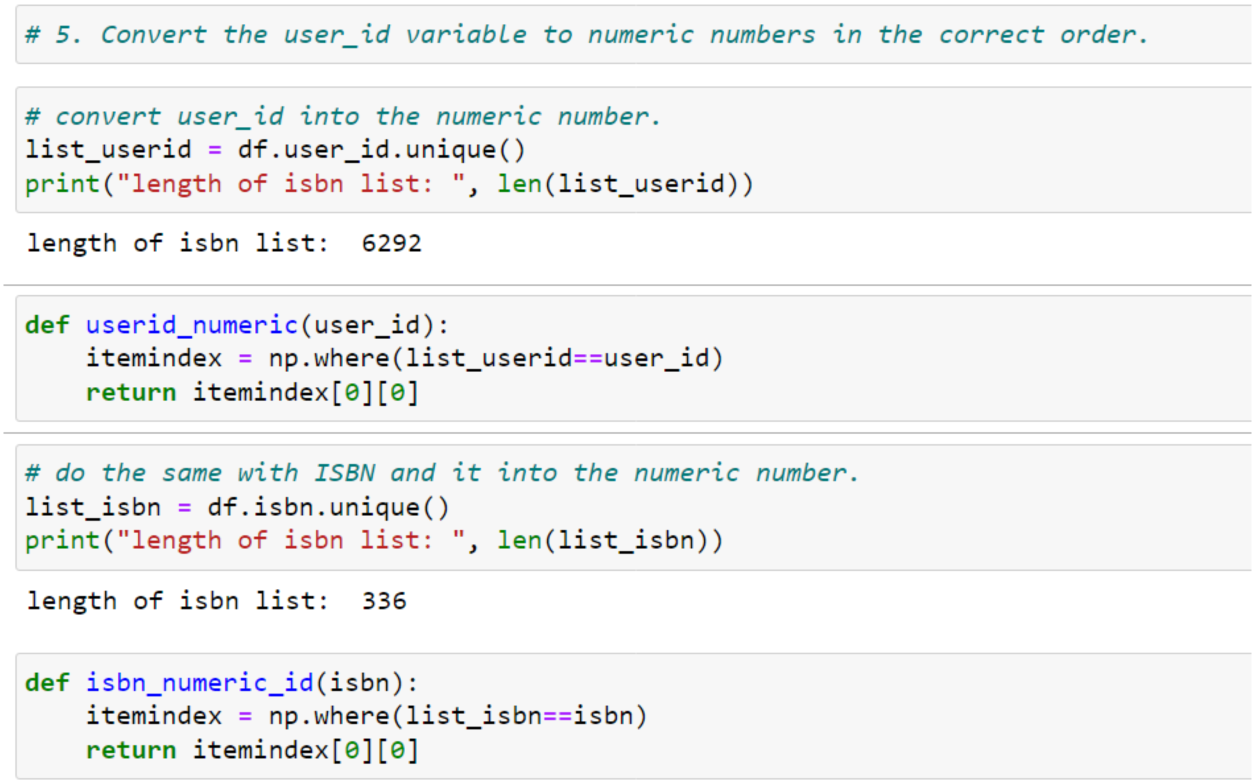
```
336
```

5. Convert ISBN variables to numeric numbers in the correct order

```
list_isbn = df.isbn.unique()
print("length of isbn list: ", len(list_isbn))


def isbn_numeric_id(isbn):
    itemindex = np.where(list_isbn==isbn)
    return itemindex[0][0]
```

6.Convert the user_id variable to numeric numbers in the correct order.

```
# convert user_id into the numeric number.
list_userid = df.user_id.unique()
print("length of isbn list: ", len(list_userid))


def userid_numeric(user_id):
    itemindex = np.where(list_userid==user_id)
    return itemindex[0][0]


# do the same with ISBN and it into the numeric number.
list_isbn = df.isbn.unique()
print("length of isbn list: ", len(list_isbn))


def isbn_numeric_id(isbn):
    itemindex = np.where(list_isbn==isbn)
    return itemindex[0][0]
```



```
# 5. Convert the user_id variable to numeric numbers in the correct order.
```

```
# convert user_id into the numeric number.
list_userid = df.user_id.unique()
print("length of isbn list: ", len(list_userid))
```

```
length of isbn list:  6292
```

```
def userid_numeric(user_id):
    itemindex = np.where(list_userid==user_id)
    return itemindex[0][0]
```

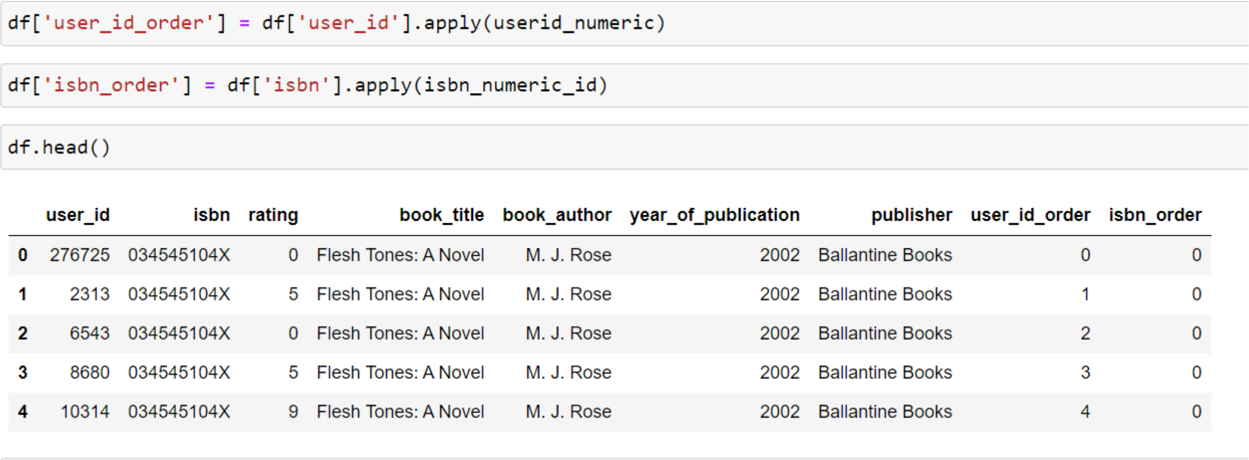
```
# do the same with ISBN and it into the numeric number.
list_isbn = df.isbn.unique()
print("length of isbn list: ", len(list_isbn))
```

```
length of isbn list:  336
```

```
def isbn_numeric_id(isbn):
    itemindex = np.where(list_isbn==isbn)
    return itemindex[0][0]
```

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

```
df['user_id_order'] = df['user_id'].apply(userid_numeric)
df['isbn_order'] = df['isbn'].apply(isbn_numeric_id)
df.head()
```



```
df['user_id_order'] = df['user_id'].apply(userid_numeric)
```

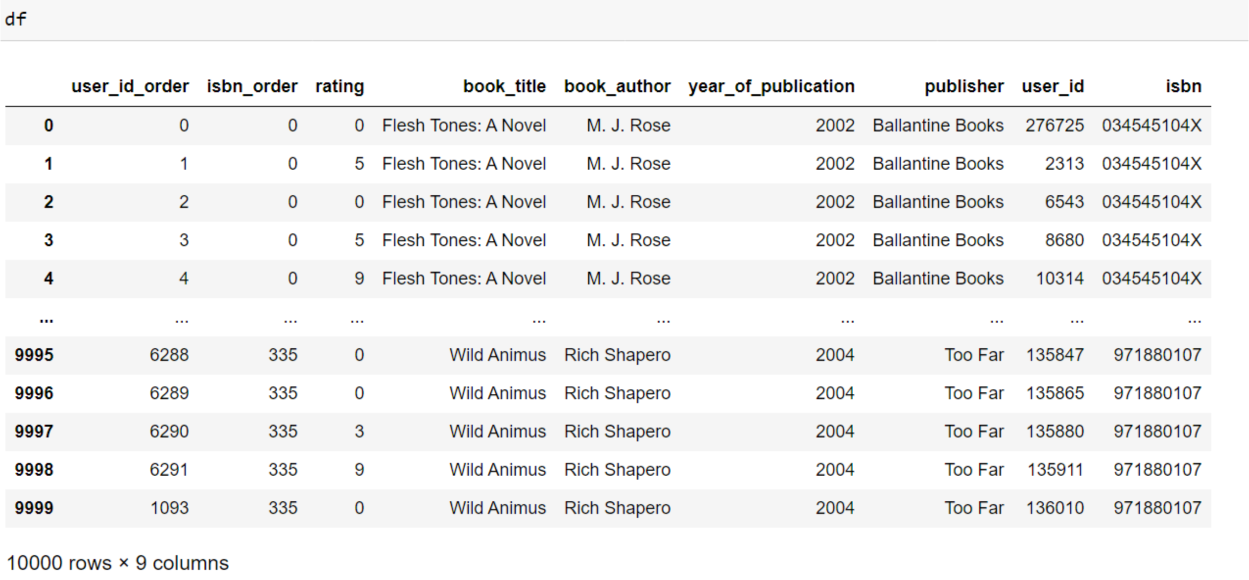
```
df['isbn_order'] = df['isbn'].apply(isbn_numeric_id)
```

```
df.head()
```

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher	user_id_order	isbn_order
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	0	0
1	2313	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	1	0
2	6543	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	2	0
3	8680	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	3	0
4	10314	034545104X	9	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	4	0

8. Re-index the columns to build a matrix

```
ordered_col = ['user_id_order', 'isbn_order', 'rating',
'book_title', 'book_author', 'year_of_publication','publisher',
               'user_id', 'isbn']
df = df.reindex(columns = ordered_col)

df
```



```
df
```

	user_id_order	isbn_order	rating	book_title	book_author	year_of_publication	publisher	user_id	isbn
0	0	0	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	276725	034545104X
1	1	0	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	2313	034545104X
2	2	0	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	6543	034545104X
3	3	0	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	8680	034545104X
4	4	0	9	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	10314	034545104X
...	...	...	...	...	...	...	...	...	...
9995	6288	335	0	Wild Animus	Rich Shapero	2004	Too Far	135847	971880107
9996	6289	335	0	Wild Animus	Rich Shapero	2004	Too Far	135865	971880107
9997	6290	335	3	Wild Animus	Rich Shapero	2004	Too Far	135880	971880107
9998	6291	335	9	Wild Animus	Rich Shapero	2004	Too Far	135911	971880107
9999	1093	335	0	Wild Animus	Rich Shapero	2004	Too Far	136010	971880107

10000 rows × 9 columns

Jjljj

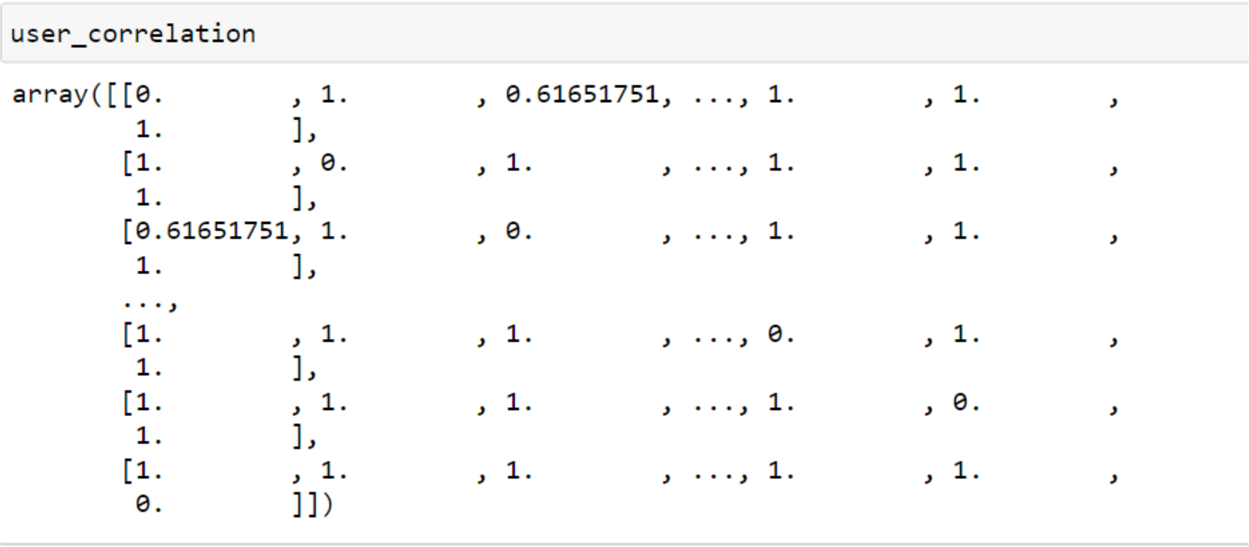
9. Split your data into two sets (training and testing)

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=.30, random_state =
10)
```

10. Make predictions based on user and item variables.

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

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

from sklearn.metrics.pairwise import pairwise_distances
user_correlation = pairwise_distances(train_matrix, metric=
'cosine')
```

```
item_correlation = pairwise_distances(train_matrix.T, metric=
'cosine')

user_correlation
```



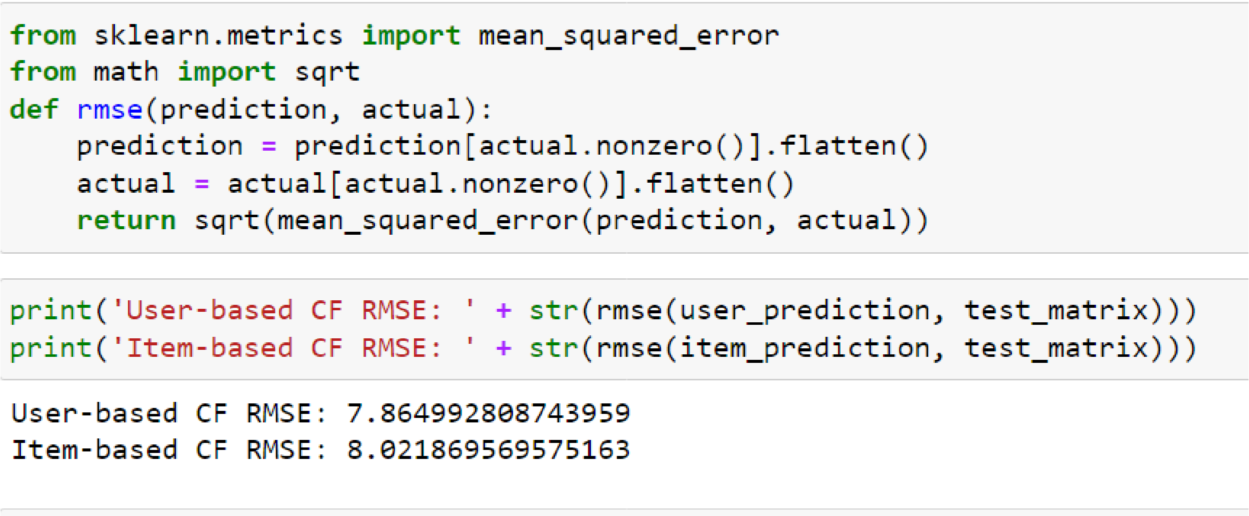
```
user_correlation
```

```
array([[0.        , 1.        , 0.61651751, ..., 1.        , 1.        ,
        1.        ],
       [1.        , 0.        , 1.        , ..., 1.        , 1.        ,
        1.        ],
       [0.61651751, 1.        , 0.        , ..., 1.        , 1.        ,
        1.        ],
       ...,
       [1.        , 1.        , 1.        , ..., 0.        , 1.        ,
        1.        ],
       [1.        , 1.        , 1.        , ..., 1.        , 0.        ,
        1.        ],
       [1.        , 1.        , 1.        , ..., 1.        , 1.        ,
        0.        ]])
```

```
def predict(ratings,correlation, type= 'user'):
    if type == 'user':
        mean_user_rating = ratings.mean(axis=1)
        rating_diff = (ratings - mean_user_rating[:,np.newaxis])
        pred = mean_user_rating[:, np.newaxis] +
correlation.dot(rating_diff) /
np.array([np.abs(correlation).sum(axis=1)]).T
    elif type == 'item':
        pred = ratings.dot(correlation) /
np.array([np.abs(correlation).sum(axis=1)])

    return pred


user_prediction = predict(train_matrix, user_correlation, type =
'user')
item_prediction = predict(train_matrix, item_correlation, type =
'item')

item_prediction.shape
```

11. Use RMSE to evaluate the predictions.

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

print('User-based CF RMSE: ' + str(rmse(user_prediction,
test_matrix)))
print('Item-based CF RMSE: ' + str(rmse(item_prediction,
test_matrix)))
```



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

```
print('User-based CF RMSE: ' + str(rmse(user_prediction, test_matrix)))
print('Item-based CF RMSE: ' + str(rmse(item_prediction, test_matrix)))
```

```
User-based CF RMSE: 7.864992808743959
Item-based CF RMSE: 8.021869569575163
```