ML Project 6 - Book Rental Recommendation

February 16, 2023

1 ML Project 6 - Book Rental Recommendation

2 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

[1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

3 Read the books dataset and explore it

[2]:

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

[3]:

```
user = pd.read_csv('BX-Users.csv',encoding='latin-1')
```

[4]:

```
ratings = pd.read_csv('BX-Book-Ratings.csv',encoding='latin-1',nrows=10000)
```

In Ratings dataset - The first 10000 datasets is only read due to out of memory error

[5]:

```
user.head()
```

[5]:

```
  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
```

[6]:

```
user.isnull().sum()
```

[6]:

```
user_id          0
Location         1
Age         110763
dtype: int64
```

[7]:

```
user.shape
```

[7]:

```
(278859, 3)
```

[8]:

```
user.dtypes
```

[8]:

```
user_id      object
Location     object
Age         float64
dtype: object
```

[9]:

```
user.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278859 entries, 0 to 278858
Data columns (total 3 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   user_id   278859 non-null  object
 1   Location  278858 non-null  object
 2   Age       168096 non-null  float64
dtypes: float64(1), object(2)
memory usage: 6.4+ MB
```

[10]:

```
user.describe()
```

[10]:

```
                 Age
count  168096.000000
mean       34.751434
std        14.428097
min         0.000000
25%        24.000000
```

```
50%        32.000000
75%        44.000000
max       244.000000
```

4 Clean up NaN values

[11]:
```
user1 = user.dropna()
```

[12]:
```
user1.isnull().sum()
```

[12]:
```
user_id     0
Location    0
Age         0
dtype: int64
```

[13]:
```
book.head()
```

[13]:
```
        isbn                                         book_title  \
0  195153448                                Classical Mythology
1    2005018                                       Clara Callan
2   60973129                               Decision in Normandy
3  374157065  Flu: The Story of the Great Influenza Pandemic…
4  393045218                             The Mummies of Urumchi

            book_author year_of_publication                   publisher
0    Mark P. O. Morford                2002     Oxford University Press
1  Richard Bruce Wright                2001       HarperFlamingo Canada
2          Carlo D'Este                1991             HarperPerennial
3      Gina Bari Kolata                1999        Farrar Straus Giroux
4       E. J. W. Barber                1999  W. W. Norton &amp; Company
```

[14]:
```
book.shape
```

[14]:
```
(271379, 5)
```

[15]:
```
book.dtypes
```

[15]:
```
isbn                   object
book_title             object
book_author            object
year_of_publication    object
publisher              object
dtype: object
```

[16]:
```
book.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
```

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

[17]:
```
             isbn      book_title      book_author year_of_publication  \
count      271379          271379           271378              271379
unique     271379          242150           102042                 202
top     195153448  Selected Poems  Agatha Christie                2002
freq            1              27              632               17145

        publisher
count      271377
unique      16823
top     Harlequin
freq         7535
```

[18]:
```
book.isnull().sum()
```

[18]:
```
isbn                   0
book_title             0
book_author            1
year_of_publication    0
publisher              2
dtype: int64
```

[19]:
```
book1 = book.dropna()
```

[20]:
```
book1.isnull().sum()
```

[20]:
```
isbn                   0
book_title             0
book_author            0
year_of_publication    0
publisher              0
dtype: int64
```

5 Read the data where ratings are given by users

```
[21]: ratings.head()
```

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

```
[22]: ratings.shape
```

```
[22]: (10000, 3)
```

```
[23]: ratings.dtypes
```

```
[23]: user_id     int64
      isbn       object
      rating      int64
      dtype: object
```

```
[24]: ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   user_id  10000 non-null  int64
 1   isbn     10000 non-null  object
 2   rating   10000 non-null  int64
dtypes: int64(2), object(1)
memory usage: 234.5+ KB
```

```
[25]: ratings.describe()
```

```
[25]:             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
```

```
[26]: ratings.isnull().sum()
```

```
[26]: user_id    0
      isbn       0
      rating     0
      dtype: int64
```

```
[27]: df = pd.merge(ratings,book,on='isbn')
```

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

```
[28]:    user_id        isbn  rating             book_title      book_author  \
      0   276725  034545104X       0  Flesh Tones: A Novel       M. J. Rose
      1   276726   155061224       5      Rites of Passage       Judith Rae
      2   276727   446520802       0          The Notebook  Nicholas Sparks
      3   278418   446520802       0          The Notebook  Nicholas Sparks
      4   276729  052165615X       3        Help!: Level 1    Philip Prowse

        year_of_publication                   publisher
      0                2002            Ballantine Books
      1                2001                      Heinle
      2                1996                Warner Books
      3                1996                Warner Books
      4                1999  Cambridge University Press
```

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

```
[29]: n_users = df['user_id'].nunique()
      print('Number of Unique User :',n_users)
```

```
Number of Unique User : 828
```

```
[30]: n_books = df['isbn'].nunique()
      print('Number of Unique Books :',n_books)
```

```
Number of Unique Books : 8051
```

7 Convert ISBN variables to numeric numbers in the correct order

```
[31]: isbn_list = df['isbn'].unique()
```

```
[32]: print('Length of isbn list :',len(isbn_list))
```

```
Length of isbn list : 8051
```

```
[33]: def get_isbn_numeric_id(isbn):
          itemindex = np.where(isbn_list==isbn)
          return itemindex[0][0]
```

8 Convert the user_id variable to numeric numbers in the correct order

[34]:
```
userid_list = df['user_id'].unique()
```

[35]:
```
print('Length of User id list :',len(userid_list))
```

```
Length of User id list : 828
```

[36]:
```
def get_user_id_numeric_id(user_id):
    itemindex = np.where(userid_list==user_id)
    return itemindex[0][0]
```

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

[37]:
```
df['user_id_order'] = df['user_id'].apply(get_user_id_numeric_id)
```

[38]:
```
df['isbn_id'] = df['isbn'].apply(get_isbn_numeric_id)
```

[39]:
```
df.head()
```

[39]:
```
   user_id        isbn  rating             book_title      book_author  \
0   276725  034545104X       0  Flesh Tones: A Novel       M. J. Rose
1   276726   155061224       5      Rites of Passage       Judith Rae
2   276727   446520802       0          The Notebook  Nicholas Sparks
3   278418   446520802       0          The Notebook  Nicholas Sparks
4   276729  052165615X       3        Help!: Level 1    Philip Prowse

  year_of_publication                   publisher  user_id_order  isbn_id
0                2002            Ballantine Books              0        0
1                2001                      Heinle              1        1
2                1996                Warner Books              2        2
3                1996                Warner Books              3        2
4                1999  Cambridge University Press              4        3
```

10 Re-index the columns to build a matrix

[40]:
```
new_col_order =␣
 ↪['user_id_order','isbn_id','rating','book_title','book_author','year_of_publication','publi
```

[41]:
```
df = df.reindex(columns=new_col_order)
df.head()
```

[41]:
```
   user_id_order  isbn_id  rating            book_title      book_author  \
0              0        0       0  Flesh Tones: A Novel       M. J. Rose
```

```
1              1        1       5      Rites of Passage       Judith Rae
2              2        2       0          The Notebook  Nicholas Sparks
3              3        2       0          The Notebook  Nicholas Sparks
4              4        3       3        Help!: Level 1    Philip Prowse

  year_of_publication                   publisher        isbn  user_id
0                2002            Ballantine Books  034545104X   276725
1                2001                      Heinle   155061224   276726
2                1996                Warner Books   446520802   276727
3                1996                Warner Books   446520802   278418
4                1999  Cambridge University Press  052165615X   276729
```

11 Split your data into two sets (training and testing)

[42]:

```python
from sklearn.model_selection import train_test_split
```

[43]:

```python
train_data,test_data = train_test_split(df,test_size=0.30)
```

[44]:

```python
train_data_matrix = np.zeros((n_users,n_books))
```

[45]:

```python
for line in train_data.itertuples():
    train_data_matrix[line[1]-1,line[2]-1] = line[3]
```

[46]:

```python
test_data_matrix = np.zeros((n_users,n_books))
```

[47]:

```python
for line in test_data.itertuples():
    test_data_matrix[line[1]-1,line[2]-1] = line[3]
```

[48]:

```python
from sklearn.metrics.pairwise import pairwise_distances
```

[49]:

```python
user_similarity = pairwise_distances(train_data_matrix,metric='cosine')
item_similarity = pairwise_distances(train_data_matrix.T,metric='cosine')
```

[50]:

```python
user_similarity
```

[50]:

```
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.]])
```

[51]:

```python
item_similarity
```

[51]:

```
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.]])
```

12 Make predictions based on user and item variables

[52]:

```
def predict(ratings, similarity, type='user'):
    if type == 'user':
        mean_user_rating = ratings.mean(axis=1)
        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
```

[53]:

```
item_prediction = predict(train_data_matrix,item_similarity,type='item')
```

[54]:

```
user_prediction = predict(train_data_matrix,user_similarity,type='user')
```

13 Use RMSE to evaluate the predictions

[55]:

```
from sklearn.metrics import mean_squared_error
import math
```

[56]:

```
def rmse(prediction, ground_truth):
    prediction = prediction[ground_truth.nonzero()].flatten()
    ground_truth = ground_truth[ground_truth.nonzero()].flatten()
    return np.sqrt(mean_squared_error(prediction,ground_truth))
```

[57]:

```
print('User-based collaborative filtering RMSE :
 ↪',rmse(user_prediction,test_data_matrix))
```

```
User-based collaborative filtering RMSE : 7.620289671855066
```

[58]:

```
print('Item-based collaborative filtering RMSE :
 ↪',rmse(item_prediction,test_data_matrix))
```

```
Item-based collaborative filtering RMSE : 7.619726712253677
```

13.0.1 Both the approach yield almost has the same results

[ ]:

```
```