Recommed Book Rent

August 30, 2023

1 Book Rental Recommendation .

Course-end Project 4 # 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.

[1]:

```
import  pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
from sklearn import preprocessing
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import PolynomialFeatures
```

2 Following operations should be performed:

3 • Read the books dataset and explore it

[2]:

```
book_df =pd.read_csv('BX-Books.csv', encoding ='latin')
book_df.head()
```

[2]:

```
        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
```

[3]:

```
ratings_df =pd.read_csv('BX-Book-Ratings.csv',encoding ='latin')
ratings_df.head()
```

[3]:

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

[4]:

```
user_df =pd.read_csv('BX-Users.csv',encoding ='latin')
user_df.head()
```

[4]:

```
  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
```

[5]:

```
recommend_df =pd.read_csv('Recommend.csv',encoding='latin')
recommend_df.head()
```

[5]:

```
   196  242  3  881250949
0  186  302  3  891717742
```

```
1   22  377  1  878887116
2  244   51  2  880606923
3  166  346  1  886397596
4  298  474  4  884182806
```

[6]:

```
print( "the shape of book_df{}".format(book_df.shape))
print( "the shape of book_ratings_df{}".format(ratings_df.shape))
print( "the shape of user_df{}".format(user_df.shape))
```

```
the shape of book_df(271379, 5)
the shape of book_ratings_df(1048575, 3)
the shape of user_df(278859, 3)
```

[7]:

```
# lets find the columns
print( "the shape of bx_book_df{}".format(book_df.columns))
print( "the shape of bx_book_ratings_df{}".format(ratings_df.columns))
print( "the shape of bx_user_df{}".format(user_df.columns))
```

```
the shape of bx_book_dfIndex(['isbn', 'book_title', 'book_author',
'year_of_publication',
       'publisher'],
      dtype='object')
the shape of bx_book_ratings_dfIndex(['user_id', 'isbn', 'rating'],
dtype='object')
the shape of bx_user_dfIndex(['user_id', 'Location', 'Age'], dtype='object')
```

[8]:

```
# lets find the nan value with its percentage for each book, ratings,user book.
percentage_book = (book_df.isna().sum(axis=0)/book_df.shape[0])*100
percentage_book
```

[8]:

```
isbn                   0.000000
book_title             0.000000
book_author            0.000368
year_of_publication    0.000000
publisher              0.000737
dtype: float64
```

[9]:

```
book_df.isnull().sum(axis=0)
```

[9]:

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

[10]:

```
book_df.isnull().sum(axis=0).value_counts()
```

```
[10]: 0    3
      2    1
      1    1
      dtype: int64
```

```
[11]: ratings_df.isnull().sum(axis=0)
```

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

```
[12]: user_df.isnull().sum(axis=0)
```

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

```
[13]: user_df.isnull().sum(axis=0).value_counts()
```

```
[13]: 110763    1
      1         1
      0         1
      dtype: int64
```

Interpretation: we have seen that in the book dataset we have book_author=1 and publisher=2 as a NAN values and in the user dataset we have Location= 1 ,age= 110763 NAN values.

4 • Clean up NaN values.

```
[14]: user_df['Location'].unique()
```

```
[14]: array(['nyc, new york, usa', 'stockton, california, usa',
             'moscow, yukon territory, russia', …,
             'sergnano, lombardia, italy', 'stranraer, n/a, united kingdom',
             'tacoma, washington, united kingdom'], dtype=object)
```

```
[15]: book_df= book_df.dropna()
```

```
[16]: book_df.isna().sum(axis=0)
```

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

```
[17]: user_df =user_df.dropna()
```

```
[18]: user_df.isna().sum(axis=0)
```

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

```
[19]: ratings_df.isnull().sum(axis=0)
```

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

Interpretation: we have cleaned the NAN values by droping all the NAN values.

5 • Read the data where ratings are given by users

```
[20]: book_df.shape
```

```
[20]: (271376, 5)
```

```
[21]: book_df.columns
```

```
[21]: Index(['isbn', 'book_title', 'book_author', 'year_of_publication',
             'publisher'],
            dtype='object')
```

```
[22]: user_df.shape
```

```
[22]: (168096, 3)
```

```
[23]: user_df.columns
```

```
[23]: Index(['user_id', 'Location', 'Age'], dtype='object')
```

```
[24]: ratings_df.shape
```

```
[24]: (1048575, 3)
```

[25]:
```
ratings_df.columns
```

[25]:
```
Index(['user_id', 'isbn', 'rating'], dtype='object')
```

[26]:
```
ratings_df.describe()
```

[26]:
```
            user_id        rating
count  1.048575e+06  1.048575e+06
mean   1.285089e+05  2.879907e+00
std    7.421876e+04  3.857870e+00
min    2.000000e+00  0.000000e+00
25%    6.339400e+04  0.000000e+00
50%    1.288350e+05  0.000000e+00
75%    1.927790e+05  7.000000e+00
max    2.788540e+05  1.000000e+01
```

we can see we cannot clear with the dataset values ,hence,lets recall rating datasets and call 10000 rows to rechack the description of the data.

[27]:
```
ratings_df.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
```

[28]:
```
ratings1_df =pd.read_csv('BX-Book-Ratings.csv',encoding ='latin', nrows=10000)
ratings1_df.head()
```

[28]:
```
   user_id        isbn  rating
0   276725  034545104X       0
1   276726   155061224       5
2   276727   446520802       0
3   276729  052165615X       3
4   276729   521795028       6
```

[29]:
```
ratings1_df.describe()
```

[29]:
```
             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
```

Interpretation: we have seen that the ratings and user id count is 1048575 values including int64, object and float as well. but in calling the rating1 data we have seen that count is 10000 standard deviation is aprox 56000 and min value =2 and max= 278854. hence we have seen that calling all the values has made memory crashed to read the data. we will stick to work with 10000 dataset.

6 lets merge the new rating1df and book dataset and then re merge for our final data set with the user dataset.

[30]:
```
# lets first merge the  book and rating data.
final_df =pd.merge(ratings1_df,book_df, on='isbn')
final_df
```

[30]:
```
      user_id        isbn  rating                 book_title  \
0      276725  034545104X       0         Flesh Tones: A Novel
1      276726   155061224       5             Rites of Passage
2      276727   446520802       0                 The Notebook
3      278418   446520802       0                 The Notebook
4      276729  052165615X       3               Help!: Level 1
…         …          …       …                          …
8696      243   385720106       7           A Map of the World
8697      243   425092917       0       The Accidental Tourist
8698      243   425098834       0        If Morning Ever Comes
8699      243   425163407       9           Unnatural Exposure
8700      243   425164403       0   Only Love (Magical Love)

                   book_author year_of_publication  \
0                   M. J. Rose                2002
1                  Judith Rae                 2001
2              Nicholas Sparks                1996
3              Nicholas Sparks                1996
4               Philip Prowse                 1999
…                         …                 …
8696            Jane Hamilton                 1999
8697              Anne Tyler                  1994
8698              Anne Tyler                  1983
8699 Patricia Daniels Cornwell                1998
8700              Erich Segal                 1998

                    publisher
```

```
0                Ballantine Books
1                          Heinle
2                    Warner Books
3                    Warner Books
4     Cambridge University Press
…                             …
8696       Anchor Books/Doubleday
8697     Berkley Publishing Group
8698     Berkley Publishing Group
8699     Berkley Publishing Group
8700     Berkley Publishing Group

[8701 rows x 7 columns]
```

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

[31]:

```
 # check for the unique id list and isbn(books):
print("The length of unique number of user  is {}".format(len(final_df.user_id.
 ↪unique()))); print("The length of unique number of books is {}".
 ↪format(len(final_df.isbn.unique())))
```

```
The length of unique number of user  is 828
The length of unique number of books is 8051
```

[32]:

```
final_book_df =pd.merge(final_df,user_df, on='user_id')
final_book_df.head()
```

[32]:

```
  user_id        isbn  rating                          book_title  \
0      99   451166892       3                The Pillars of the Earth
1      99   786868716       0  The Five People You Meet in Heaven
2      99  067976397X       0          Corelli's Mandolin : A Novel
3      99   312252617       8                          Fast Women
4      99   312261594       8                 Female Intelligence

          book_author year_of_publication          publisher  \
0          Ken Follett                1996        Signet Book
1          Mitch Albom                2003           Hyperion
2  LOUIS DE BERNIERES                1995            Vintage
3      Jennifer Crusie                2001  St. Martin's Press
4          Jane Heller                2001  St. Martin's Press

                 Location   Age
0  franktown, colorado, usa  42.0
1  franktown, colorado, usa  42.0
2  franktown, colorado, usa  42.0
```

```
3  franktown, colorado, usa  42.0
4  franktown, colorado, usa  42.0
```

Interpretation: we have noticed that in the above data final_book_df there will be no impact on location and age column hence we will be continuing to take the final dataset as final_df

[33]:
```
print("the shape of the final book data", final_book_df.shape)
print("the columns in the final book dataset",final_book_df.columns)
```

```
the shape of the final book data (136, 9)
the columns in the final book dataset Index(['user_id', 'isbn', 'rating',
'book_title', 'book_author',
       'year_of_publication', 'publisher', 'Location', 'Age'],
      dtype='object')
```

[34]:
```
 print("The length of unique number  user  is {}".format(len(final_book_df.
↪user_id.unique()))); print("The length of unique  number of books is {}".
↪format(len(final_book_df.isbn.unique())))
```

```
The length of unique number  user  is 43
The length of unique  number of books is 134
```

[35]:
```
final_book_df.describe()
```

[35]:
```
           rating         Age
count  136.000000  136.000000
mean     4.492647   36.044118
std      4.113206   12.004856
min      0.000000   14.000000
25%      0.000000   27.000000
50%      6.000000   37.000000
75%      8.000000   42.000000
max     10.000000   62.000000
```

Interpretation: we have seen that the final book dataset the rating and user id count is 682099.

[36]:
```
book_df.describe()
```

[36]:
```
              isbn      book_title      book_author year_of_publication  \
count       271376          271376           271376              271376
unique      271376          242148           102041                 202
top     156649222X  Selected Poems  Agatha Christie                2002
freq             1              27              632               17144

        publisher
count      271376
unique      16822
top     Harlequin
```

```
freq         7535
```

Interpretation : in the final data the user and number of books is around 43 and 134 and in the ratings dataset the number of user and number of books are 828 and 8051 respectivvely who rated the books.

8 • Convert ISBN variables to numeric numbers in the correct order

[37]:

```
final_df.isbn
```

[37]:

```
0       034545104X
1        155061224
2        446520802
3        446520802
4       052165615X
           …
8696     385720106
8697     425092917
8698     425098834
8699     425163407
8700     425164403
Name: isbn, Length: 8701, dtype: object
```

we have noticed that in the numeric value some parts of values has string value attached and the dtype is object, lets convert it into numeric fully.

[38]:

```
isbn_list = final_df.isbn.unique()
isbn_list
```

[38]:

```
array(['034545104X', '155061224', '446520802', …, '425098834',
       '425163407', '425164403'], dtype=object)
```

[39]:

```
print("the length of number of  book", len(isbn_list))

def isbn_numeric(isbn):
    # print("isbn:",isbn)
    isbn_index =np.where(isbn_list==isbn)
    return isbn_index[0][0]      #This line returns the index of the matched␣
 ↪ISBN in the isbn_list.
                                 #It can be used to map a given ISBN to its␣
 ↪index within the list of unique ISBNs.
```

```
the length of number of  book 8051
```

Overall, the code is creating a function (get_isbn_numeric_id) that takes an ISBN as input and returns its index within the list of unique ISBNs extracted from the df_final DataFrame.

9 • Convert ISBN to the ordered list, i.e., from 0…n-1

[40]:
```
final_df['isbn_id']=final_df['isbn'].apply(isbn_numeric)
final_df.head()
```

[40]:
```
   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  isbn_id
0                2002            Ballantine Books        0
1                2001                      Heinle        1
2                1996                Warner Books        2
3                1996                Warner Books        2
4                1999  Cambridge University Press        3
```

10 • Convert the user_id variable to numeric numbers in the correct order

[41]:
```
user_id_list =final_df.user_id.unique()
```

[42]:
```
# similarly creates the function for user_id to get converted into numeric data
print("the number of user",len(user_id_list))
def user_is_numeric(user_id):
    user_id_index= np.where(user_id_list==user_id)
    return user_id_index[0][0]
```

```
the number of user 828
```

11 • Convert user_id to the ordered list, i.e., from 0…n-1

[43]:
```
final_df['user_id_order'] = final_df['user_id'].apply(user_is_numeric)
```

[44]:
```
final_df.head()
```

[44]:
```
   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  isbn_id  user_id_order
0                2002            Ballantine Books        0              0
1                2001                      Heinle        1              1
2                1996                Warner Books        2              2
3                1996                Warner Books        2              3
4                1999  Cambridge University Press        3              4
```

Interpretation: now we can user user_id_order and isbn_id for model prediction.

12 • Re-index the columns to build a matrix

[45]:
```
# lets re index the column for building the matrix before that lets cal the␣
 ↪columns
final_df.columns
```

[45]:
```
Index(['user_id', 'isbn', 'rating', 'book_title', 'book_author',
       'year_of_publication', 'publisher', 'isbn_id', 'user_id_order'],
      dtype='object')
```

[46]:
```
# lets ordered it accordingly:
cols=['user_id_order','isbn_id',␣
 ↪'rating','book_title','book_author','year_of_publication',␣
 ↪'publisher','user_id','isbn']
final_df =final_df.reindex(columns=cols)
final_df.head()
```

[46]:
```
   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  user_id        isbn
0                2002             Ballantine Books   276725  034545104X
1                2001                       Heinle   276726   155061224
2                1996                 Warner Books   276727   446520802
3                1996                 Warner Books   278418   446520802
4                1999   Cambridge University Press   276729  052165615X
```

Now it will be easy for us to view the data and connecting it.

13 • Split your data into two sets (training and testing)

[47]:
```
from sklearn.model_selection import train_test_split
```

[48]:
```
train_data, test_data =train_test_split(final_df,random_state=7,train_size=0.8)
```

[49]:
```
train_data.columns
```

[49]:
```
Index(['user_id_order', 'isbn_id', 'rating', 'book_title', 'book_author',
       'year_of_publication', 'publisher', 'user_id', 'isbn'],
      dtype='object')
```

[50]:
```
print("the datset in train data is{}".format(train_data.shape))
print("the datset in test data is {}".format(test_data.shape))
```

```
the datset in train data is(6960, 9)
the datset in test data is (1741, 9)
```

14 Approach for Recommendation Book:

15 a) User-based nearest-neighbor collaborative filtering:

The system finds out the users who have the same sort of taste of books rading and similarity between users is computed based upon the rating behavior.

16 b) Item-based nearest-neighbor collaborative filtering:

The system checks the items that are similar to the items the user bought. The similarity between different items is computed based on the items and not the users for the prediction.

[51]:
```
n_user= final_df.user_id.nunique()
n_books =final_df.isbn.nunique()
```

```
print("Numbr of Users"+str(n_user))
print("Number of books:"+str(n_books))
train_matrix= np.zeros((n_user, n_books))
for line in train_data.itertuples():
    train_matrix[line[1]-1,line[2]-1]=line[3]


# Create user-book matrix for testing
test_matrix = np.zeros((n_user,n_books))
for line in test_data.itertuples():
    test_matrix[line[1]-1, line[2]-1] = line[3]
```

```
Numbr of Users828
Number of books:8051
```

17 • Make predictions based on user and item variables

[52]:

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

[53]:

```
# colaborative user based recommendation system
user_similarity =pairwise_distances(train_matrix, metric='cosine')
user_similarity
```

[53]:

```
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.]])
```

[54]:

```
user_similarity.shape
```

[54]:

```
(828, 828)
```

[55]:

```
# item based collaborative recommendation system
item_similarity =pairwise_distances(train_matrix.T,metric='cosine')
item_similarity
```

[55]:

```
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.]])
```

[56]:
```
item_similarity.shape
```

[56]:
```
(8051, 8051)
```

18 Make Prediction

[57]:
```
def prediction(ratings, similarity,type='user'):
    if type== 'user':
        mean_user =ratings.mean(axis=1)
        rating_diff =(ratings-mean_user[:,np.newaxis])
        pred= mean_user[:,np.newaxis]+similarity.dot(rating_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
```

[58]:
```
test_matrix.shape
```

[58]:
```
(828, 8051)
```

[59]:
```
user_prediction= prediction(train_matrix,user_similarity, type='user')
user_prediction
```

[59]:
```
array([[-0.0013735 , -0.0013735 ,  0.00225407, …, -0.0013735 ,
        -0.0013735 , -0.0013735 ],
       [ 0.00405066, -0.00199529,  0.00163228, …, -0.00199529,
        -0.00199529, -0.00199529],
       [ 0.06511313,  0.05906554,  0.06269409, …,  0.05906554,
         0.05906554,  0.05906554],
       …,
       [ 0.00405066, -0.00199529,  0.00163228, …, -0.00199529,
        -0.00199529, -0.00199529],
       [ 0.00405066, -0.00199529,  0.00163228, …, -0.00199529,
        -0.00199529, -0.00199529],
       [ 0.00405066, -0.00199529,  0.00163228, …, -0.00199529,
        -0.00199529, -0.00199529]])
```

[60]:
```
item_similarity.shape
```

[60]:
```
(8051, 8051)
```

[61]:

```
item_prediction =prediction(train_matrix,item_similarity,type='item')
item_prediction
```

[61]:

```
array([[0.        , 0.00062112, 0.0006212 , …, 0.00062112, 0.00062112,
         0.00062112],
        [0.        , 0.        , 0.        , …, 0.        , 0.        ,
         0.        ],
        [0.06099379, 0.06099379, 0.06100137, …, 0.06099379, 0.06099379,
         0.06099379],
        …,
        [0.        , 0.        , 0.        , …, 0.        , 0.        ,
         0.        ],
        [0.        , 0.        , 0.        , …, 0.        , 0.        ,
         0.        ],
        [0.        , 0.        , 0.        , …, 0.        , 0.        ,
         0.        ]])
```

19 • Use RMSE to evaluate the predictions

[64]:

```
# 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))
```

[65]:

```
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: 7.818113790950216
Item-based CF RMSE: 7.817194775495292
```

20 Conclusion:

we have seen that recommendation system by using collaborative filtering user based and item based has an accuracy of model pridiction for the book name according to rating is 78%.

[109]:

```
final_df.shape
```

[109]:

```
(8701, 9)
```

```
[110]: final_df['user_id'].value_counts()
```

```
[110]: 278418    3997
       277427     490
       277639     265
       278188     194
       277478     187
                  …
       277827       1
       277819       1
       277811       1
       277803       1
       278528       1
       Name: user_id, Length: 828, dtype: int64
```

```
[ ]:
```

```
[ ]:
```

```
[114]: # Extra Work:
       #we can presume the recoomendation code:
       # using collaborative filtering method
       # how many user rated more than 10
       x =final_df.groupby('user_id').count()['rating']>10
       x[x]# count the index place where it is true
```

```
[114]: user_id
       8          True
       99         True
       242        True
       243        True
       276762     True
                  …
       278633     True
       278637     True
       278771     True
       278843     True
       278851     True
       Name: rating, Length: 77, dtype: bool
```

Interpretation: we can check there are books whose ratings are above 200.

```
[115]: # how many books are there which are rated by 828 user?
       required_user =x[x].index
       required_user
```

```
[115]: Int64Index([     8,     99,    242,    243, 276762, 276798, 276822, 276828,
            276847, 276856, 276859, 276866, 276925, 276929, 276939, 276954,
            276964, 276984, 276994, 277042, 277051, 277157, 277168, 277171,
            277187, 277195, 277196, 277203, 277378, 277427, 277439, 277466,
            277478, 277523, 277629, 277639, 277662, 277681, 277710, 277711,
            277744, 277879, 277882, 277901, 277922, 277928, 277929, 277937,
            277945, 277954, 277965, 277982, 277984, 278002, 278026, 278137,
            278144, 278188, 278194, 278202, 278221, 278314, 278346, 278356,
            278390, 278418, 278506, 278522, 278535, 278554, 278563, 278582,
            278633, 278637, 278771, 278843, 278851],
           dtype='int64', name='user_id')
```

```
[116]: final_df[final_df['user_id'].isin(required_user)]
```

```
[116]:       user_id_order  isbn_id  rating                book_title  \
       3                 3        2       0              The Notebook
       8                 3        6       0           A Painted House
       10                8        7       0                 Lightning
       12                9        8       8       Manhattan Hunt Club
       15                3       10       0                Night Sins
       …             …      …     …                       …
       8696             96     8046       7        A Map of the World
       8697             96     8047       0    The Accidental Tourist
       8698             96     8048       0     If Morning Ever Comes
       8699             96     8049       9        Unnatural Exposure
       8700             96     8050       0  Only Love (Magical Love)

                          book_author year_of_publication                 publisher  \
       3              Nicholas Sparks                1996              Warner Books
       8                 JOHN GRISHAM                2001                 Doubleday
       10              Dean R. Koontz                1996  Berkley Publishing Group
       12                   JOHN SAUL                2002          Ballantine Books
       15                   TAMI HOAG                1995                    Bantam
       …                         …                 …                       …
       8696             Jane Hamilton                1999    Anchor Books/Doubleday
       8697                Anne Tyler                1994  Berkley Publishing Group
       8698                Anne Tyler                1983  Berkley Publishing Group
       8699  Patricia Daniels Cornwell                1998  Berkley Publishing Group
       8700               Erich Segal                1998  Berkley Publishing Group

             user_id        isbn
       3      278418   446520802
       8      278418  038550120X
       10     277427   425115801
       12     278026   449006522
       15     278418  055356451X
       …       …         …
```

```
8696       243   385720106
8697       243   425092917
8698       243   425098834
8699       243   425163407
8700       243   425164403

[7240 rows x 9 columns]
```

[117]:
```
final_df['user_id'].nunique()
```

[117]:
```
828
```

[118]:
```
# lets filter the rating according the  name of the books and the user that␣
 ↪rated it?
filter_rating =final_df[final_df['user_id'].isin(required_user)]
```

[119]:
```
filter_rating.shape
```

[119]:
```
(7240, 9)
```

[120]:
```
filter_rating.head()
```

[120]:
```
    user_id_order  isbn_id  rating           book_title      book_author  \
3               3        2       0         The Notebook  Nicholas Sparks
8               3        6       0      A Painted House     JOHN GRISHAM
10              8        7       0            Lightning   Dean R. Koontz
12              9        8       8  Manhattan Hunt Club        JOHN SAUL
15              3       10       0           Night Sins        TAMI HOAG

   year_of_publication                 publisher  user_id        isbn
3                 1996              Warner Books   278418   446520802
8                 2001                 Doubleday   278418  038550120X
10                1996  Berkley Publishing Group   277427   425115801
12                2002           Ballantine Books   278026   449006522
15                1995                    Bantam   278418  055356451X
```

[121]:
```
filter_rating.describe()
```

[121]:
```
       user_id_order      isbn_id       rating        user_id
count    7240.000000  7240.000000  7240.000000    7240.000000
mean       56.077348  3985.956768     1.383978  274224.448481
std       107.459209  2230.542641     3.021188   32601.232483
min         3.000000     2.000000     0.000000       8.000000
25%         3.000000  2017.750000     0.000000  277639.000000
50%         3.000000  4093.500000     0.000000  278418.000000
75%        89.000000  5898.250000     0.000000  278418.000000
max       775.000000  8050.000000    10.000000  278851.000000
```

[122]:

```
# lets put this in the new variable called as y
y =filter_rating.groupby('book_title').count()['rating']
y[y]
```

[122]:

```
book_title
01-01-00: The Novel of the Millennium    1
01-01-00: The Novel of the Millennium    1
01-01-00: The Novel of the Millennium    1
01-01-00: The Novel of the Millennium    1
01-01-00: The Novel of the Millennium    1
                                        ..
01-01-00: The Novel of the Millennium    1
01-01-00: The Novel of the Millennium    1
01-01-00: The Novel of the Millennium    1
01-01-00: The Novel of the Millennium    1
01-01-00: The Novel of the Millennium    1
Name: rating, Length: 6760, dtype: int64
```

[123]:

```
y.value_counts()
```

[123]:

```
1     6394
2      295
3       49
4       13
5        5
7        1
10       1
6        1
9        1
Name: rating, dtype: int64
```

[124]:

```
famous_books= y[y].index
famous_books
```

[124]:

```
Index(['01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '100 Best-Loved Poems (Dover Thrift Editions)',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       …
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '100 Best-Loved Poems (Dover Thrift Editions)',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium',
       '01-01-00: The Novel of the Millennium'],
      dtype='object', name='book_title', length=6760)
```

[125]:
```
final_df['book_title']
```

[125]:
```
0            Flesh Tones: A Novel
1                Rites of Passage
2                    The Notebook
3                    The Notebook
4                  Help!: Level 1
                        …
8696             A Map of the World
8697         The Accidental Tourist
8698          If Morning Ever Comes
8699             Unnatural Exposure
8700       Only Love (Magical Love)
Name: book_title, Length: 8701, dtype: object
```

[126]:
```
final_rating=filter_rating[filter_rating['book_title'].isin(famous_books)]
final_rating
```

[126]:
```
      user_id_order  isbn_id  rating  \
915             202      749       5
1206            239      977       8
1914              8     1566       0
3165             95     2644       9
4368              3     3756       0
6020              3     5401       0
6564              3     5937       0
6573              3     5946       0
7480              3     6848       0
7800              3     7166       0

                                        book_title        book_author  \
915          01-01-00: The Novel of the Millennium       R. J. Pineiro
1206                                101 Dalmatians         Walt Disney
1914                             101 Great Resumes       Career Press
3165  100 Best-Loved Poems (Dover Thrift Editions)       Philip Smith
4368                                101 Dalmatians      Justine Korman
6020                  1001 Ways to Cut Your Expenses   Jonathan D. Pond
```

```
6564                                 101 Bug Jokes       Lisa Eisenberg
6573                                 101 Pet Jokes          Phil Hirsch
7480                     100 Days of Fun at School  Janet Palazzo Craig
7800      101 Best Home-Based Businesses for Women    Priscilla Y. Huff

     year_of_publication                        publisher  user_id        isbn
915                 1999                   Tor Books (Mm)   277168   812568710
1206                1995              Stoddart+publishing   277203   717284832
1914                1995                  Delmar Learning   277427  1564142019
3165                1995               Dover Publications   277965   486285537
4368                1996  Golden Books Publishing Company   278418   307001164
6020                1992          Dell Publishing Company   278418   440504953
6564                1986            Scholastic Paperbacks   278418   590332473
6573                1980                  Scholastic Inc.   278418   590371177
7480                1998             Troll Communications   278418   816745412
7800                1995                 Prima Lifestyles   278418  1559587032
```

```
[127]: # lets apply the collaborative filtering method by finding the pivot table
       pivot =final_rating.pivot_table(index='book_title',columns=␣
        ↪'user_id',values='rating')
       pivot
```

```
[127]: user_id                                        277168  277203  277427  277965  \
       book_title
       01-01-00: The Novel of the Millennium             5.0     NaN     NaN     NaN
       100 Best-Loved Poems (Dover Thrift Editions)      NaN     NaN     NaN     9.0
       100 Days of Fun at School                         NaN     NaN     NaN     NaN
       1001 Ways to Cut Your Expenses                    NaN     NaN     NaN     NaN
       101 Best Home-Based Businesses for Women          NaN     NaN     NaN     NaN
       101 Bug Jokes                                     NaN     NaN     NaN     NaN
       101 Dalmatians                                    NaN     8.0     NaN     NaN
       101 Great Resumes                                 NaN     NaN     0.0     NaN
       101 Pet Jokes                                     NaN     NaN     NaN     NaN

       user_id                                        278418
       book_title
       01-01-00: The Novel of the Millennium             NaN
       100 Best-Loved Poems (Dover Thrift Editions)      NaN
       100 Days of Fun at School                         0.0
       1001 Ways to Cut Your Expenses                    0.0
       101 Best Home-Based Businesses for Women          0.0
       101 Bug Jokes                                     0.0
       101 Dalmatians                                    0.0
       101 Great Resumes                                 NaN
       101 Pet Jokes                                     0.0
```

[132]:
```
pivot.fillna(0, inplace =True)
pivot
```

[132]:
```
user_id                                      277168  277203  277427  277965  \
book_title
01-01-00: The Novel of the Millennium           5.0     0.0     0.0     0.0
100 Best-Loved Poems (Dover Thrift Editions)    0.0     0.0     0.0     9.0
100 Days of Fun at School                       0.0     0.0     0.0     0.0
1001 Ways to Cut Your Expenses                  0.0     0.0     0.0     0.0
101 Best Home-Based Businesses for Women        0.0     0.0     0.0     0.0
101 Bug Jokes                                   0.0     0.0     0.0     0.0
101 Dalmatians                                  0.0     8.0     0.0     0.0
101 Great Resumes                               0.0     0.0     0.0     0.0
101 Pet Jokes                                   0.0     0.0     0.0     0.0

user_id                                      278418
book_title
01-01-00: The Novel of the Millennium           0.0
100 Best-Loved Poems (Dover Thrift Editions)    0.0
100 Days of Fun at School                       0.0
1001 Ways to Cut Your Expenses                  0.0
101 Best Home-Based Businesses for Women        0.0
101 Bug Jokes                                   0.0
101 Dalmatians                                  0.0
101 Great Resumes                               0.0
101 Pet Jokes                                   0.0
```

[133]:
```
pivot.shape
```

[133]:
```
(9, 5)
```

[135]:
```
# find the similarity using cosine_similarity pairwise distances
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.metrics.pairwise import cosine_similarity
```

[155]:
```
similar_score = cosine_similarity(pivot)
similar_score
```

[155]:
```
array([[1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 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., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

[156]:
```
pivot.index
```

[156]:
```
Index(['01-01-00: The Novel of the Millennium',
       '100 Best-Loved Poems (Dover Thrift Editions)',
       '100 Days of Fun at School', '1001 Ways to Cut Your Expenses',
       '101 Best Home-Based Businesses for Women', '101 Bug Jokes',
       '101 Dalmatians', '101 Great Resumes', '101 Pet Jokes'],
      dtype='object', name='book_title')
```

[157]:
```
pivot.index[0]
```

[157]:
```
'01-01-00: The Novel of the Millennium'
```

[158]:
```
# lets define a recoomendation book
def recommend(book_name):
    index= np.where(pivot.index==book_name)[0][0]
    similar_items =sorted(list(enumerate(similar_score[index])),key=lambda x:
 ↪x[1],reverse=True)[1:6]
    for i in similar_items:
        print(pivot.index[i[0]])
```

[159]:
```
recommend('01-01-00: The Novel of the Millennium')
```

```
100 Best-Loved Poems (Dover Thrift Editions)
100 Days of Fun at School
1001 Ways to Cut Your Expenses
101 Best Home-Based Businesses for Women
101 Bug Jokes
```

[163]:
```
book_df.head()
```

[163]:
```
        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
```

Interpretation: we have seen after applying the cosine similarity function between book title and user id with respect to ratings, we can ask the recoomendation wihich is similar to the novel '01-01-00: The Novel of the Millennium' and we the model had predicted certainly: Model Prediction:

100 Best-Loved Poems (Dover Thrift Editions) 100 Days of Fun at School 1001 Ways to Cut Your Expenses 101 Best Home-Based Businesses for Women 101 Bug Jokes

[174]:
```
# lets call the rating and user id alon with the book title
# we are using the same define class recommmend but with little modification:

def recommend(book_name):
    # fetch index using book_name
    index = np.where(pivot.index==book_name)[0][0]
    similar_items = sorted(list(enumerate(similar_score[index])),key = lambda x:
 ↪ x[1], reverse=True)[1:6]

    data=[]
    for i in similar_items:
        items=[]
        temp_df = book_df[book_df['book_title'] == pivot.index[i[0]]]
        items.extend(list(temp_df.drop_duplicates('book_title')['book_title'].
 ↪values))
        items.extend(list(temp_df.drop_duplicates('book_title')['book_author'].
 ↪values))
        items.extend(list(temp_df.drop_duplicates('book_title')['publisher'].
 ↪values))

        data.append(items)
    return data
```

[175]:
```
recommend('01-01-00: The Novel of the Millennium')
```

[175]:
```
[['100 Best-Loved Poems (Dover Thrift Editions)',
  'Philip Smith',
  'Dover Publications'],
 ['100 Days of Fun at School', 'Janet Palazzo Craig', 'Troll Communications'],
 ['1001 Ways to Cut Your Expenses',
  'Jonathan D. Pond',
  'Dell Publishing Company'],
 ['101 Best Home-Based Businesses for Women',
  'Priscilla Y. Huff',
  'Prima Lifestyles'],
 ['101 Bug Jokes', 'Lisa Eisenberg', 'Scholastic Paperbacks']]
```

21 Conclusion:

hence we noticed that recommendation of books according to user and ratings has been displayed above. Machine Learning Using recoomendation system techniques shows the accuracy of prediction novels is 78%. list of novels, publisher and authors predicted accordingly by using cosine_similarity function and user based and item based collaborative filtering techniques.

```
[ ]:
```