C3389C AY2018 TERM 4
Coursework Submission

Personal Details	
Name	**LIM YUAN HER**
Admin No.	**17060167**

Compliance Statement	
Plagiarism I declare that this report is my original work. I understand that if I am suspected of plagiarism, my enrolment in the programme may be terminated.	✓
Retention of Backup Copy I declare that I have a back-up electronic copy of this report for immediate submission.	✓
Signature	
IMPORTANT: Non-compliance to these clauses will result in unconditional rejection of your submission	

Table of Contents

Table of Figures

List of Tables

1 Introduction

1.1 Background

This report details the development of a recommender system application based on the Book Crossing Dataset [1].

The Book-Crossing dataset consists of 3 tables:

- BX-Users

  Contains the users.

- BX-Books

  Contains the books and include identification information e.g. ISBN and content-based information e.g. Book-Title, Book-Author, Year-Of-Publication, Publisher etc. obtained from Amazon Web Services. URLs linking to cover images are included in small (Image-URL-S), medium (Image-URL-M), and large (Image-URL-L) choices.

- BX-Book-Ratings

  Contains both implicit (denoted by 0 rating) and explicit (on a scale from 1-10) book rating information..

1.2 Objectives

The objective of this recommender system application as follows:

1. Collect relevant preferences/ratings information from a new user.
2. Based on user information from the preceding step, recommend a set of 5 items.
3. Requests user feedback/rating on the items recommended in the preceding step.
4. Generates a comparison of predicted vs actual user rating on the recommended items in the preceding steps

2 Features

The features of the recommender system application are:

1. Add new user with attributes e.g. Location, Age.
2. Add book ratings for user based on pre-defined list of books.
3. Get a list of book recommendations for user.
4. Update recommender model(s) based on user selection

The figure below illustrates the program flow of the recommender:

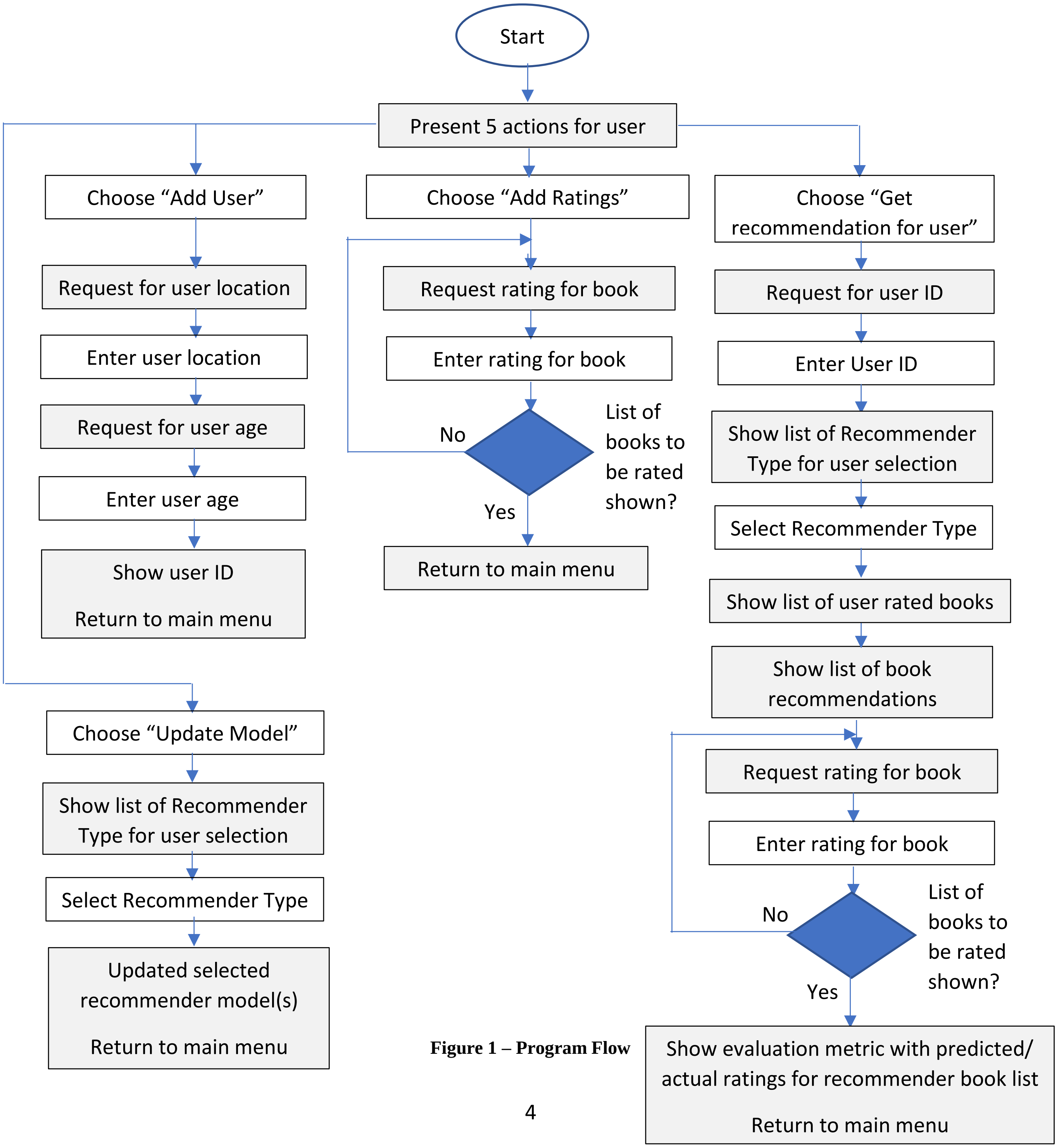

Start
Present 5 actions for user
Choose “Add User”
Request for user location
Enter user location
Request for user age
Enter user age
Show user ID
Return to main menu
Choose “Add Ratings”
Request rating for book
Enter rating for book
List of books to be rated shown?
No
Yes
Return to main menu
Choose “Get recommendation for user”
Request for user ID
Enter User ID
Show list of Recommender Type for user selection
Select Recommender Type
Show list of user rated books
Show list of book recommendations
Request rating for book
Enter rating for book
List of books to be rated shown?
No
Yes
Show evaluation metric with predicted/ actual ratings for recommender book list
Return to main menu
Choose “Update Model”
Show list of Recommender Type for user selection
Select Recommender Type
Updated selected recommender model(s)
Return to main menu

Figure 1 – Program Flow

3 Implementation

3.1 Dataset Preparation

Due to system hardware limitation, the original BX-Ratings dataset with **1,149,780 records** is trimmed down to a manageable size suitable for a CPU-powered execution environment using the following steps:

1. Filter to include only records with book ISBNs in Books dataset
2. Filter to include only records with user ID in Users dataset
3. Filter to include only explicit ratings i.e. rating from 1 to 10
4. Filter to include ony records with top 120 users with the highest number of book ratings
5. Filter to include ony records with top 120 books with the highest number of user ratings

The resulting dataset consists of **1,052** records with **117 users** and **120 books**

Refer to Appendix 1 for the Jupyter Notebook output for dataset trimming.

3.2 Recommender Algorithms

The 6 recommender algorithms were implemented with the Python programming language using open-source libraries e.g. numpy, pandas, scipy, sklearn, tensorflow etc. and can be broadly classified in 3 different categories, namely content-based, collaborative-filtering and matrix factorization based:

3.2.1 Content-Based

Content-based recommendation is implemented using TF-IDF (term frequency inverse document frequency), which uses the *TfidfVectorizer* class from the *sklearn.feature_extraction.text* package to tokenize the book titles into a feature set of keywords from which is computed the similarity of books based on the book title. Based on this, the top 2 most similar books for each book rated at least 5.0 by the user are extracted as a list and ordered by occurrence frequency in the list (in descending order). The top 5 items from the ordered list is then presented as recommendations for the user.

3.2.2 Collaborative Filtering

Two types of collaborative filtering mechanisms are implemented, namely user-based and item-based. User-based collaborative filtering recommends items by finding similar users to the user of interest whilst item-based collaborative filtering recommends items by finding similar items to the items liked by the user of interest. The similarity between users/ books are measured using distance measures e.g. cosine simliarity, adjusted cosine similarity, correlation, or euclidean distance. Predicted ratings are computed by taking the weighted average of ratings with similarities as weights normalized by the number of users/ books with a bias offset applied using the user's mean rating for all previously rated books. Books that have yet to be rated by the user are then ordered by their predicted rating and the top 5 itesm are presented as recommendations for the user.

3.2.3 Matrix Factorization

This family of recommender algorithm decomposes the user-book interaction matrix into two lower-rank matrices to discover the latent features, which when combined (by taking the dot product), will recover the original user-book interaction. A few variations of matrix factorization are available, and in this recommender system application, both SVD and SGD types are implemented.

In both variants, books that have yet to be rated by the user are ordered by their predicted ratings (in descending order) and the top 5 itesm are presented as recommendations for the user.

3.2.3.1 SVD (Singular Value Decomposition)

SVD leverages a latent factor model to capture the relationship between users and books. In this recommender system application, this is performed by using the *svds* class from the *scipy.sparse.linalg* package to derive 2 matrices (user-factor and book-factor) that maps the users/books to latent factors that can be used to predict ratings for books that the user has not rated yet.

The *svds* class requires input parameter for the number of singular values and vectors to compute and this is the number of terms required to capture at least 90% of the information. After a trial-and-error analysis, it is observed that a value of 100 is sufficient to fulfill this requirement.

3.2.3.2 SGD (Stochastic Gradient Descent)

SGD treats this as an optimization problem and measures how good the predicted rating matches the actual user given rating. Ratings for unrated items are predicted by minimizing the regularized squared error with respect to latent user/ item feature matrices.

The SGD model takes 4 input parameters and after a trial-and-error analysis, the optimal parameter values selected and brief description of each parameter are summarized in the table below:

s/n	Parameter	Description	Value
1	k	Number of factors (latent dimensions)	100
2	alpha	Learning Rate	0.1
3	beta	Regularization parameter	0.01
4	iterations	Number of iterations of the SGD procedure	15

Table 1 – SGD Model Parametsrs

3.2.4 Neural Network

This algorithm make recommendations by learning embeddings of items using a neural network. Neural network embeddings are low-dimensional and learned i.e. similar entities are placed closer to one another in the embedding space. This implementation uses the embedding layer functionality of Keras to define 2 embedding layers, one for users, and one for books. It uses *mean squared error* as the loss function and *Stochastic Gradient Descent* (learning rate of 0.08, momentum of 0.9) as the optimizer. The predicted ratings are then calculated by taking the dot product of both embedding layrers. Books that have yet to be rated by the user are then ordered by their predicted ratings (in descending order) and presented as recommendations for the user.

3.3 Recommender Model Persistence

The recommender model outputs are saved to disk when ratings have been added for a new user or when manual update requests are made. The purpose of saving the model outputs is for subsequent quick loading when requests for user recommendations are made instead of having to perform the time-consuming similarity/ predicted rating recalculations again. The figure below illustrates this process:

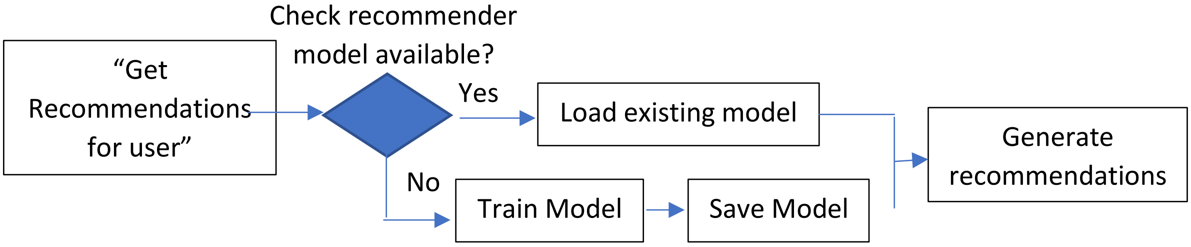


Figure 2 – Generate Recommendations Process

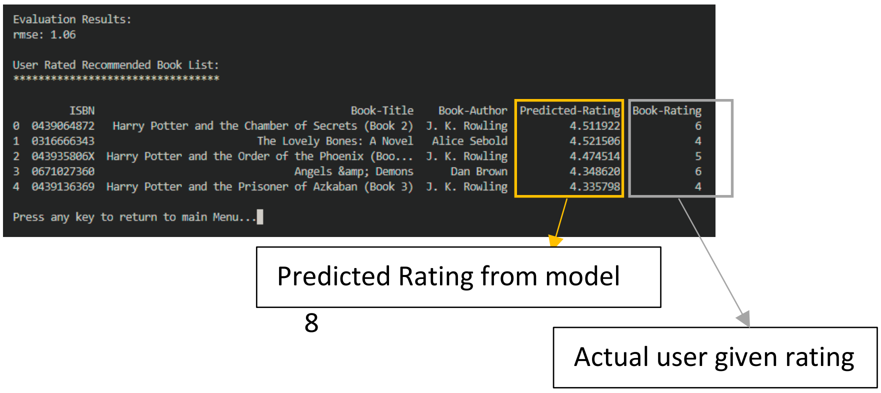
The table below summarizes the model saved filename and stored information:

s/n	Recommender Type	Model Filename	Model Info
1	TF-IDF	TfidfModel.pkl	Similarity
2	User-Based Collaborative Filtering	user_CFModel.pkl	Predicted Ratings
3	Item-Based Collaborative Filtering	item_CFModel.pkl	Predicted Ratings
4	SVD	SVDModel.pkl	Predicted Ratings
5	SGD	SGDModel.pkl	Predicted Ratings
6	Embedding (Neural Network)	EMBModel.h5	Keras model

Table 2 – Recommender Model Information

3.4 Evaluation

To compare the predicted and actual user rating on the recommended items, the **root mean squared error (RMSE)** metric, using the *mean_squared_error* class from *sklearn.metrics* package is used, taking the square root to produce the desired result.



```
Evaluation Results:
rmse: 1.06

User Rated Recommended Book List:
*********************************

        ISBN                                    Book-Title    Book-Author  Predicted-Rating  Book-Rating
0  0439064872   Harry Potter and the Chamber of Secrets (Book 2)  J. K. Rowling          4.511922            6
1  0316666343                        The Lovely Bones: A Novel   Alice Sebold          4.521506            4
2  043935806X  Harry Potter and the Order of the Phoenix (Boo...  J. K. Rowling          4.474514            5
3  0671027360                              Angels &amp; Demons      Dan Brown          4.348620            6
4  0439136369  Harry Potter and the Prisoner of Azkaban (Book 3)  J. K. Rowling          4.335798            4

Press any key to return to main Menu...
```

4 Recommender System Application

4.1 Objective

The recommender system application is designed as a console application that allows the user to perform the following functions:

1. Add a new user (by providing location and age information). The user id for the new uesr will be displayed.
2. Get ratings from the new user for 15 pre-determined books as listed below:
   1) The Da Vinci Code (0385504209)
   2) The Firm (044021145X)
   3) The Pelican Brief (0440214041)
   4) Interview with the Vampire (0345337662)
   5) Harry Potter and the Sorcerer's Stone (059035342X)
   6) Jurassic Park (0345370775)
   7) Life of Pi (0156027321)
   8) To Kill a Mockingbird (0446310786)
   9) Red Dragon (0440206154)
   10) Fahrenheit 451 (0345342968)
   11) Lord of the Flies (0399501487)
   12) Silence of the Lambs (0312924585)
   13) The Hobbit (0345339681)
   14) The English Patient (0679745203)
   15) The Green Mile (0671041789)

   The pre-determined book-list is selected based on familiarity of the book title rather than selecting the most popular books The user is not required to give a rating for all the books listed and can exit at any stage of the book listing.
3. Get top 5 book recommendations for the new user based on their selection of recommender model as listed below:
   1) Content-Based (TF-IDF)
   2) User-based Collaborative-Filtering
   3) Item-based Collaborative-Filtering
   4) SVD-based (Singular Value Decomposition)
   5) SGD-based (Stochastic Gradient Descent)
   6) Embeddings (Neural Network-based)

   The user is also given an option to provide ratings for the 5 recommended books and the system will generate a comparison of predicted vs actual user rating on these recommended books.

4. Manually update all recommender models (as listed above) with latest user-book information.
5. Exit the application.

The typical workflow would be to add a new user, followed by providing ratings for the 15 pre-determined books from the new user, and then get recommendations for this new user based on their selection of recommender model. This translates to options (1), (2), (3) in the main menu screen.

4.2 Application Requirements

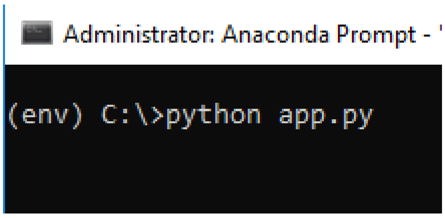
The recommender system application requires the following libraries to be installed:

- numpy
- pandas
- sklearn
- scipy
- tensorflow

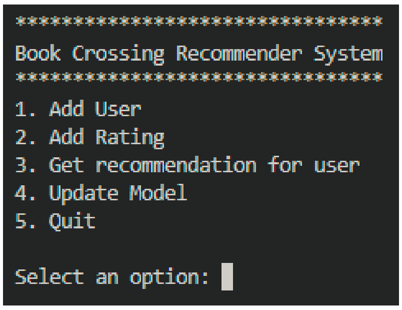
These libraries can be installed by executing the command “pip install <libraryname>” at the command prompt. Alternatively, the required libraries can be installed by executing the command “pip install -r requirements.txt” at the command prompt where the “requirements.txt” file is located.

4.3 Start Program

To start the recommender system application, navigate to the base directory where the “app.py” file is located e.g. “C:\BCRec\app.py”. Then, start the program by typing “python app.py” in the command prompt:



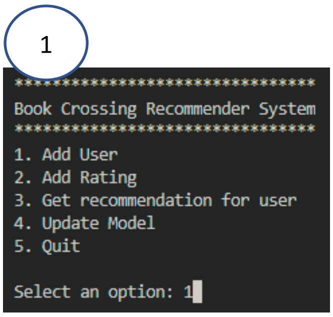
```
(env) C:\>python app.py
```



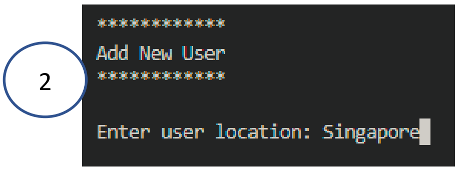
```
*********************************
Book Crossing Recommender System
*********************************
1. Add User
2. Add Rating
3. Get recommendation for user
4. Update Model
5. Quit

Select an option:
```

The main menu will be displayed.

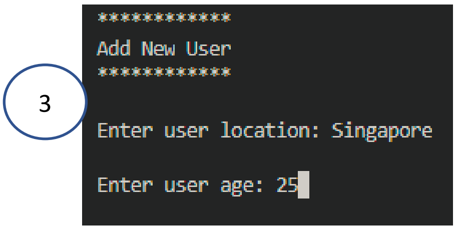
4.4 Add User



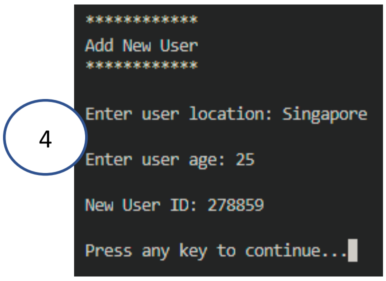
```
*********************************
Book Crossing Recommender System
*********************************
1. Add User
2. Add Rating
3. Get recommendation for user
4. Update Model
5. Quit

Select an option: 1
```



```
************
Add New User
************

Enter user location: Singapore
```



```
************
Add New User
************

Enter user location: Singapore

Enter user age: 25
```



```
************
Add New User
************

Enter user location: Singapore

Enter user age: 25

New User ID: 278859

Press any key to continue...
```

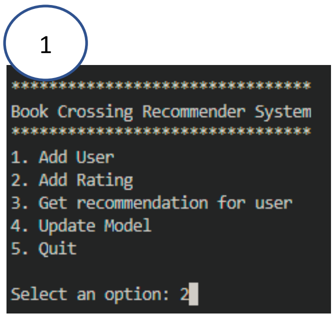
(1) In the main menu screen, type 1 at the prompt and press the enter key. The "Add New User" screen will be displayed:

(2) & (3) Enter the user location and age information at the prompts as shown in (2) and (3).

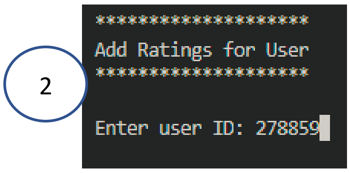
(4) The user ID for the newly added user will be displayed as shown in (4).

4.5 Add Ratings

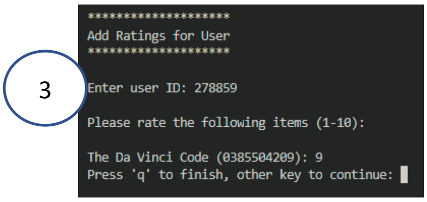
(1) In the main menu screen, type 2 at the prompt and press the enter key. The "Add Ratings for user" screen will be displayed.



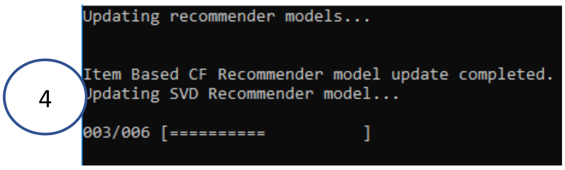
```
*********************************
Book Crossing Recommender System
*********************************
1. Add User
2. Add Rating
3. Get recommendation for user
4. Update Model
5. Quit

Select an option: 2
```



```
********************
Add Ratings for User
********************

Enter user ID: 278859
```



```
********************
Add Ratings for User
********************

Enter user ID: 278859

Please rate the following items (1-10):

The Da Vinci Code (0385504209): 9
Press 'q' to finish, other key to continue:
```



```
Updating recommender models...

Item Based CF Recommender model update completed.
Updating SVD Recommender model...

003/006 [==========          ]
```

(2) Enter the user ID.

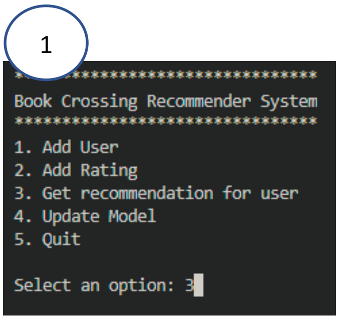
(3) Enter the rating on a scale from 1 to 10 for each book displayed.

(4) After entering the rating for the last book, the recommender models for the various algorithms will start updating with the progress shown.

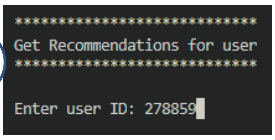
(5) After all the recommender models have been successfully updated, the main menu will be displayed.

4.6 Get Recommendations

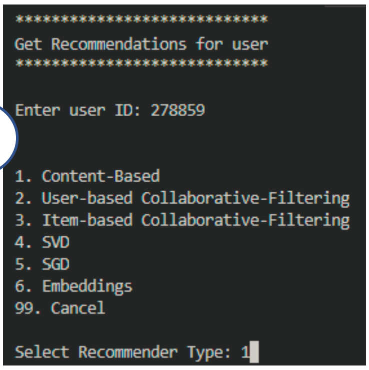
(1) In the main menu screen, type 3 at the prompt and press the enter key. The “Get Recommendations for user’ screen will be displayed.



```
********************************
Book Crossing Recommender System
********************************
1. Add User
2. Add Rating
3. Get recommendation for user
4. Update Model
5. Quit

Select an option: 3
```



```
****************************
Get Recommendations for user
****************************

Enter user ID: 278859
```

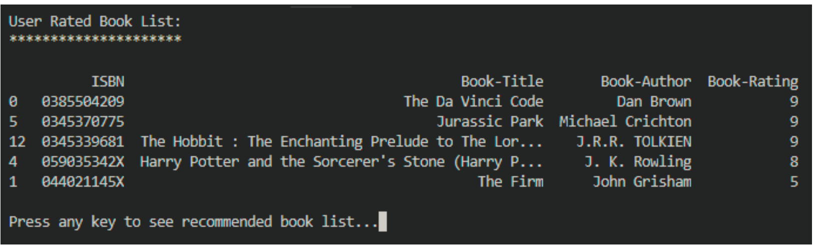


```
****************************
Get Recommendations for user
****************************

Enter user ID: 278859

1. Content-Based
2. User-based Collaborative-Filtering
3. Item-based Collaborative-Filtering
4. SVD
5. SGD
6. Embeddings
99. Cancel

Select Recommender Type: 1
```

(2) Enter the user ID.

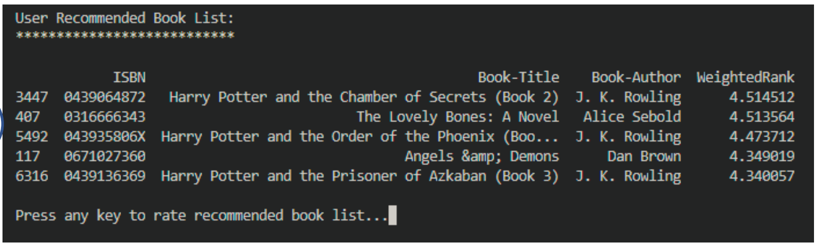
(3) Enter number corresponding to the recommender type e.g. 1 for content-based.

(4) The top 5 books with the highest ratings by the user will be displayed:



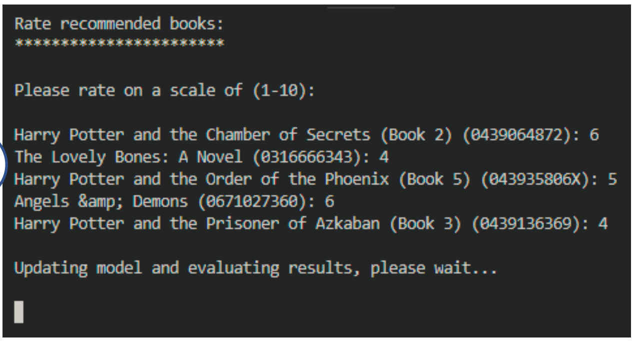
```
User Rated Book List:
*********************

          ISBN                                       Book-Title       Book-Author  Book-Rating
0   0385504209                                The Da Vinci Code         Dan Brown            9
5   0345370775                                    Jurassic Park  Michael Crichton            9
12  0345339681  The Hobbit : The Enchanting Prelude to The Lor...      J.R.R. TOLKIEN            9
4   059035342X  Harry Potter and the Sorcerer's Stone (Harry P...    J. K. Rowling            8
1   044021145X                                         The Firm      John Grisham            5

Press any key to see recommended book list...
```

(5) The top 5 recommended books forthe user based on the selected recommender type will be displayed:



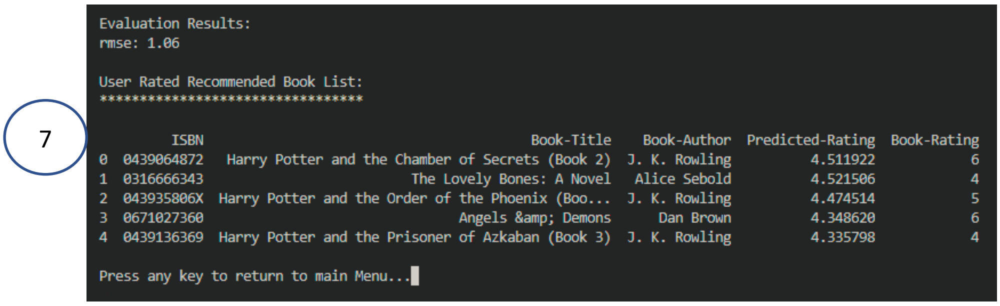
```
User Recommended Book List:
***************************

            ISBN                                         Book-Title    Book-Author  WeightedRank
3447  0439064872   Harry Potter and the Chamber of Secrets (Book 2)  J. K. Rowling      4.514512
407   0316666343                          The Lovely Bones: A Novel   Alice Sebold      4.513564
5492  043935806X  Harry Potter and the Order of the Phoenix (Boo...  J. K. Rowling      4.473712
117   0671027360                               Angels &amp; Demons      Dan Brown      4.349019
6316  0439136369  Harry Potter and the Prisoner of Azkaban (Book 3)  J. K. Rowling      4.340057

Press any key to rate recommended book list...
```

(6) The user will be asked to rate the recommended books. Enter the rating on a scale of 1 to 10 for each of the recommended books displayed:



```
Rate recommended books:
***********************

Please rate on a scale of (1-10):

Harry Potter and the Chamber of Secrets (Book 2) (0439064872): 6
The Lovely Bones: A Novel (0316666343): 4
Harry Potter and the Order of the Phoenix (Book 5) (043935806X): 5
Angels &amp; Demons (0671027360): 6
Harry Potter and the Prisoner of Azkaban (Book 3) (0439136369): 4

Updating model and evaluating results, please wait...
```

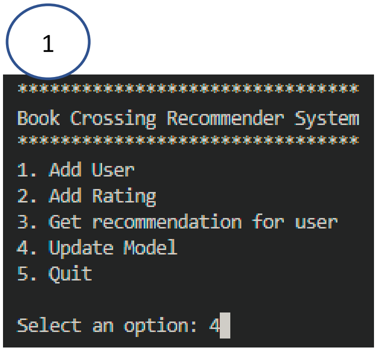
(7) The RMSE evaluation metric will be displayed along with the predicted and actual ratings given:



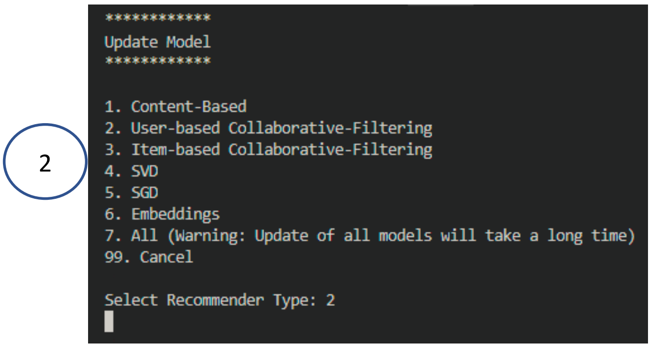
```
Evaluation Results:
rmse: 1.06

User Rated Recommended Book List:
*********************************


         ISBN                                      Book-Title    Book-Author  Predicted-Rating  Book-Rating
0  0439064872   Harry Potter and the Chamber of Secrets (Book 2)  J. K. Rowling          4.511922            6
1  0316666343                        The Lovely Bones: A Novel   Alice Sebold          4.521506            4
2  043935806X  Harry Potter and the Order of the Phoenix (Boo...  J. K. Rowling          4.474514            5
3  0671027360                              Angels &amp; Demons       Dan Brown          4.348620            6
4  0439136369  Harry Potter and the Prisoner of Azkaban (Book 3)  J. K. Rowling          4.335798            4

Press any key to return to main Menu...
```

4.7 Update Model

In the main menu screen, type 4 at the prompt and press the enter key. The "Update Model" screen will be displayed.



```
**********************************
Book Crossing Recommender System
**********************************
1. Add User
2. Add Rating
3. Get recommendation for user
4. Update Model
5. Quit

Select an option: 4
```



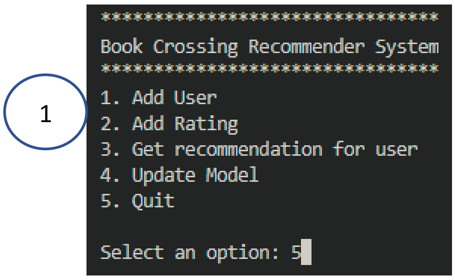
```
************
Update Model
************

1. Content-Based
2. User-based Collaborative-Filtering
3. Item-based Collaborative-Filtering
4. SVD
5. SGD
6. Embeddings
7. All (Warning: Update of all models will take a long time)
99. Cancel

Select Recommender Type: 2
```

(2) Enter number corresponding to the recommender type e.g. 2 for user-based collaborative filtering.

(3) After the recommender model has been successfully updated, the main menu will be displayed.

4.8 Exit

In the main menu screen, type 5 at the prompt and press the enter key. The program will be terminated.

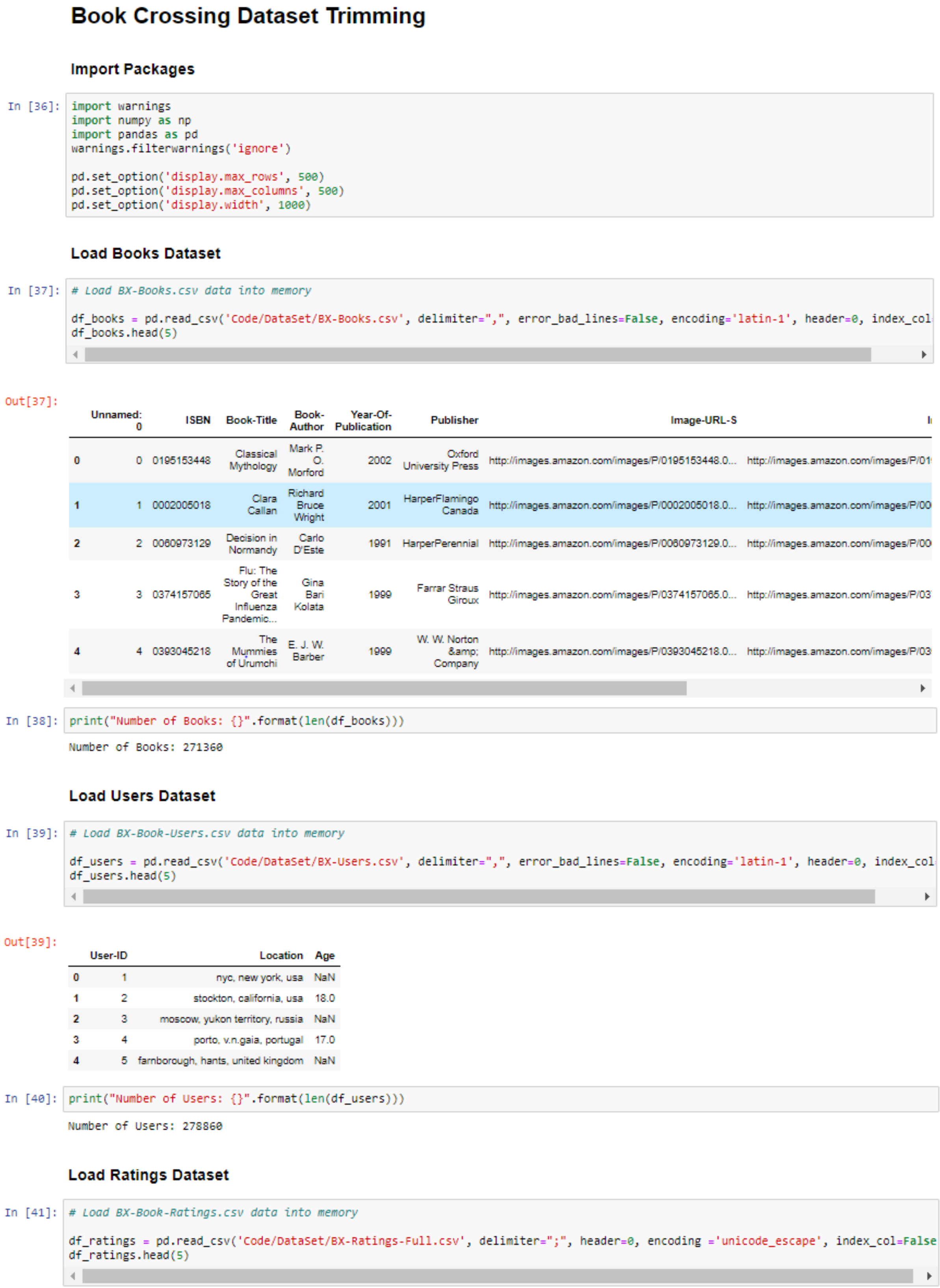


```
**********************************
Book Crossing Recommender System
**********************************
1. Add User
2. Add Rating
3. Get recommendation for user
4. Update Model
5. Quit

Select an option: 5
```

5 References

1. Book Crossing Dataset, *http://www2.informatik.uni-freiburg.de/~cziegler/BX/*
2. Improving Recommendation Lists Through Topic Diversification, Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, Georg Lausen; *Proceedings of the 14th International World Wide Web Conference (WWW '05), May 10-14, 2005, Chiba, Japan*.

6 Appendix 1



Book Crossing Dataset Trimming

Import Packages

In [36]:
```
import warnings
import numpy as np
import pandas as pd
warnings.filterwarnings('ignore')

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

Load Books Dataset

In [37]:
```
# Load BX-Books.csv data into memory

df_books = pd.read_csv('Code/DataSet/BX-Books.csv', delimiter=",", error_bad_lines=False, encoding='latin-1', header=0, index_col
df_books.head(5)
```

Out[37]:

	Unnamed: 0	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	I
0	0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/images/P/01
1	1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/images/P/00
2	2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/00
3	3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.com/images/P/03
4	4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton &amp; Company	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.com/images/P/03

In [38]:
```
print("Number of Books: {}".format(len(df_books)))
```

```
Number of Books: 271360
```

Load Users Dataset

In [39]:
```
# Load BX-Book-Users.csv data into memory

df_users = pd.read_csv('Code/DataSet/BX-Users.csv', delimiter=",", error_bad_lines=False, encoding='latin-1', header=0, index_col
df_users.head(5)
```

Out[39]:

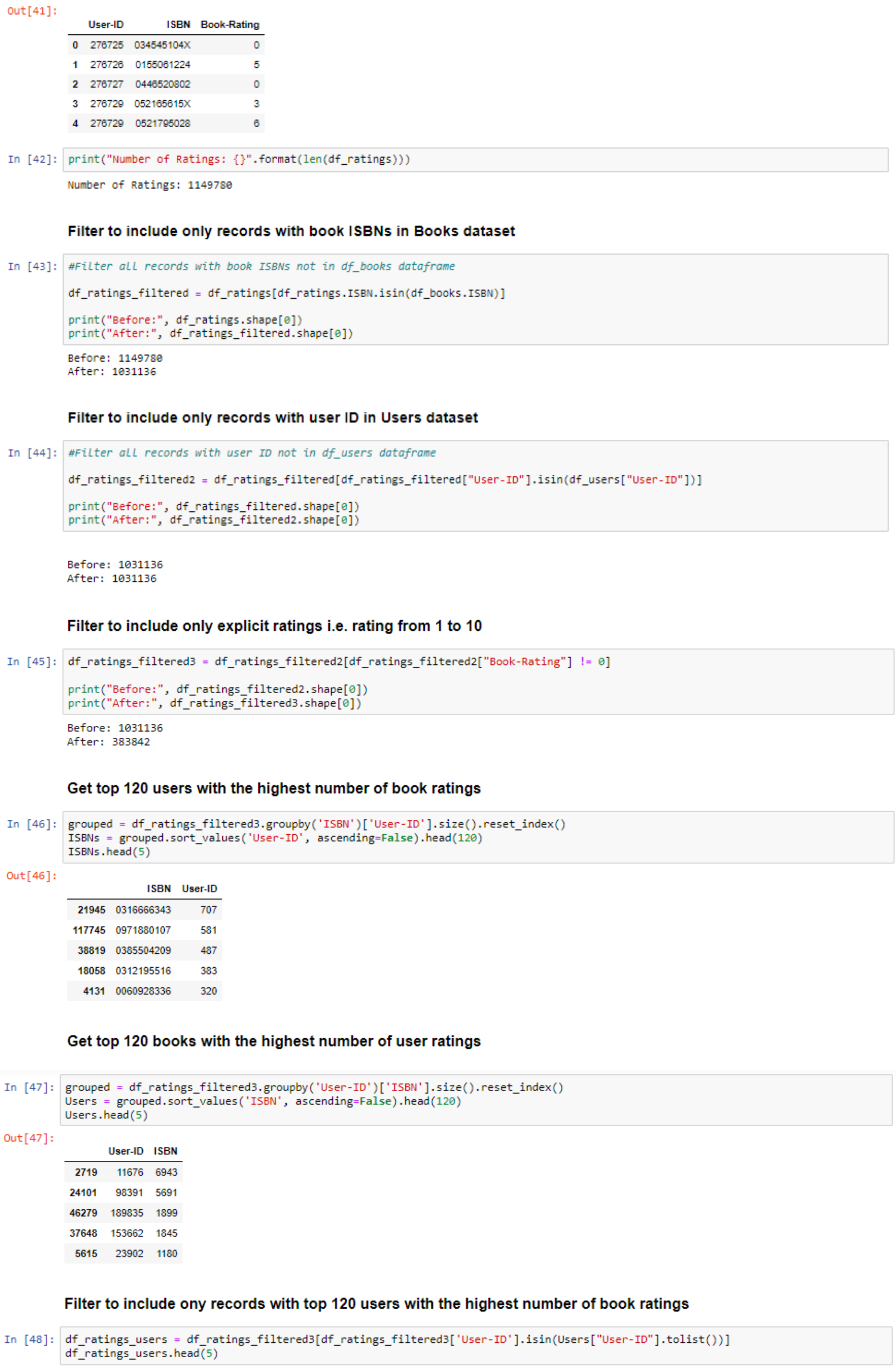
	User-ID	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN

In [40]:
```
print("Number of Users: {}".format(len(df_users)))
```

```
Number of Users: 278860
```

Load Ratings Dataset

In [41]:
```
# Load BX-Book-Ratings.csv data into memory

df_ratings = pd.read_csv('Code/DataSet/BX-Ratings-Full.csv', delimiter=";", header=0, encoding ='unicode_escape', index_col=False
df_ratings.head(5)
```



Out[41]:

	User-ID	ISBN	Book-Rating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6

In [42]:
```
print("Number of Ratings: {}".format(len(df_ratings)))
```

```
Number of Ratings: 1149780
```

Filter to include only records with book ISBNs in Books dataset

In [43]:
```
#Filter all records with book ISBNs not in df_books dataframe

df_ratings_filtered = df_ratings[df_ratings.ISBN.isin(df_books.ISBN)]

print("Before:", df_ratings.shape[0])
print("After:", df_ratings_filtered.shape[0])
```

```
Before: 1149780
After: 1031136
```

Filter to include only records with user ID in Users dataset

In [44]:
```
#Filter all records with user ID not in df_users dataframe

df_ratings_filtered2 = df_ratings_filtered[df_ratings_filtered["User-ID"].isin(df_users["User-ID"])]

print("Before:", df_ratings_filtered.shape[0])
print("After:", df_ratings_filtered2.shape[0])
```

```
Before: 1031136
After: 1031136
```

Filter to include only explicit ratings i.e. rating from 1 to 10

In [45]:
```
df_ratings_filtered3 = df_ratings_filtered2[df_ratings_filtered2["Book-Rating"] != 0]

print("Before:", df_ratings_filtered2.shape[0])
print("After:", df_ratings_filtered3.shape[0])
```

```
Before: 1031136
After: 383842
```

Get top 120 users with the highest number of book ratings

In [46]:
```
grouped = df_ratings_filtered3.groupby('ISBN')['User-ID'].size().reset_index()
ISBNs = grouped.sort_values('User-ID', ascending=False).head(120)
ISBNs.head(5)
```

Out[46]:

	ISBN	User-ID
21945	0316666343	707
117745	0971880107	581
38819	0385504209	487
18058	0312195516	383
4131	0060928336	320

Get top 120 books with the highest number of user ratings

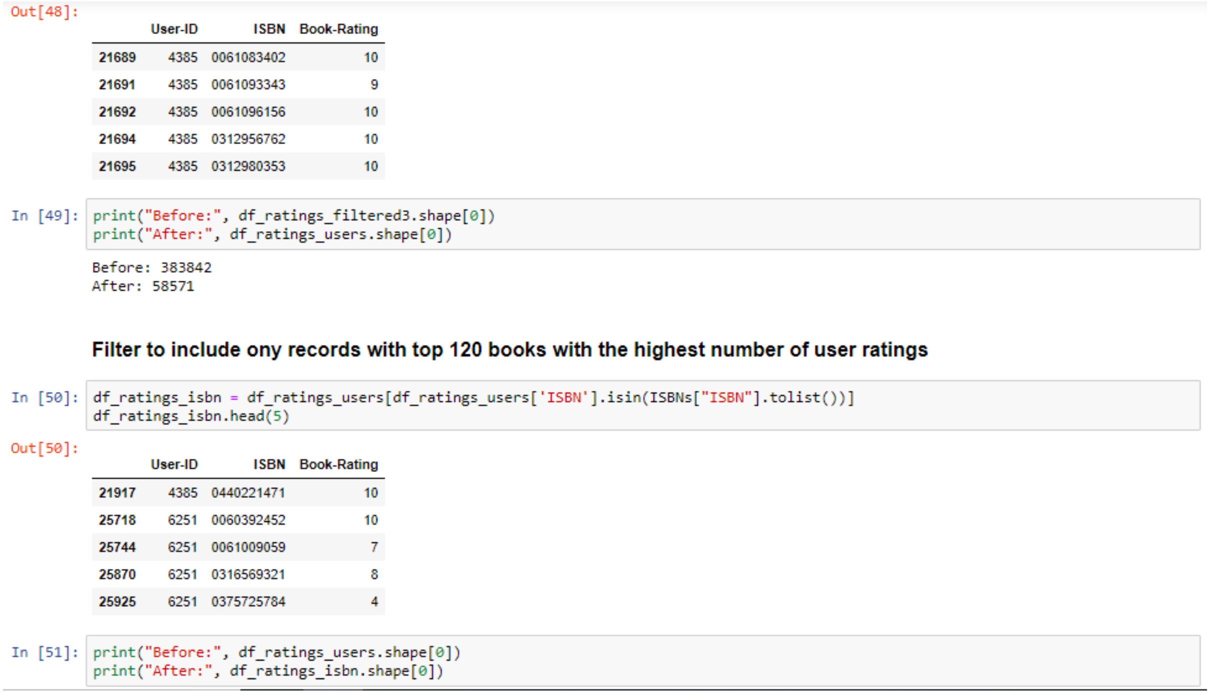
In [47]:
```
grouped = df_ratings_filtered3.groupby('User-ID')['ISBN'].size().reset_index()
Users = grouped.sort_values('ISBN', ascending=False).head(120)
Users.head(5)
```

Out[47]:

	User-ID	ISBN
2719	11676	6943
24101	98391	5691
46279	189835	1899
37648	153662	1845
5615	23902	1180

Filter to include ony records with top 120 users with the highest number of book ratings

In [48]:
```
df_ratings_users = df_ratings_filtered3[df_ratings_filtered3['User-ID'].isin(Users["User-ID"].tolist())]
df_ratings_users.head(5)
```



Out[48]:

	User-ID	ISBN	Book-Rating
21689	4385	0061083402	10
21691	4385	0061093343	9
21692	4385	0061096156	10
21694	4385	0312956762	10
21695	4385	0312980353	10

In [49]:
```
print("Before:", df_ratings_filtered3.shape[0])
print("After:", df_ratings_users.shape[0])
```

```
Before: 383842
After: 58571
```

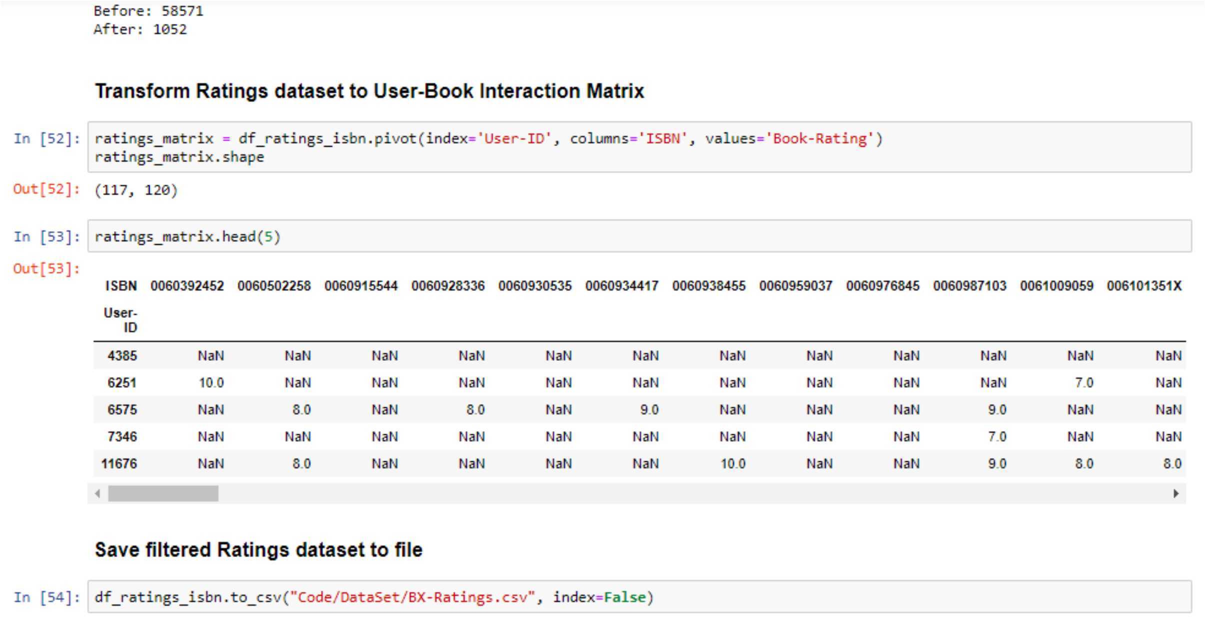
Filter to include ony records with top 120 books with the highest number of user ratings

In [50]:
```
df_ratings_isbn = df_ratings_users[df_ratings_users['ISBN'].isin(ISBNs["ISBN"].tolist())]
df_ratings_isbn.head(5)
```

Out[50]:

	User-ID	ISBN	Book-Rating
21917	4385	0440221471	10
25718	6251	0060392452	10
25744	6251	0061009059	7
25870	6251	0316569321	8
25925	6251	0375725784	4

In [51]:
```
print("Before:", df_ratings_users.shape[0])
print("After:", df_ratings_isbn.shape[0])
```



```
Before: 58571
After: 1052
```

Transform Ratings dataset to User-Book Interaction Matrix

In [52]:
```
ratings_matrix = df_ratings_isbn.pivot(index='User-ID', columns='ISBN', values='Book-Rating')
ratings_matrix.shape
```

Out[52]:
```
(117, 120)
```

In [53]:
```
ratings_matrix.head(5)
```

Out[53]:

ISBN	0060392452	0060502258	0060915544	0060928336	0060930535	0060934417	0060938455	0060959037	0060976845	0060987103	0061009059	006101351X
User-ID												
4385	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6251	10.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7.0	NaN
6575	NaN	8.0	NaN	8.0	NaN	9.0	NaN	NaN	NaN	9.0	NaN	NaN
7346	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7.0	NaN	NaN
11676	NaN	8.0	NaN	NaN	NaN	NaN	10.0	NaN	NaN	9.0	8.0	8.0

Save filtered Ratings dataset to file

In [54]:
```
df_ratings_isbn.to_csv("Code/DataSet/BX-Ratings.csv", index=False)
```