About Book Crossing Dataset

This dataset has been compiled by Cai-Nicolas Ziegler in 2004, and it comprises of three tables for users, books anc
scale from 1-10 (higher values denoting higher appreciation) and implicit rating is expressed by 0.

Reference: http://www2.informatik.uni-freiburg.de/~cziegler/BX/

Objective

This project entails building a Book Recommender System for users based on user-based and item-based collabora

Execute the below cell to load the datasets

```
import io
import pandas as pd
import numpy as np
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.moun
```

```
path1 = "/content/drive/My Drive/Residency 5 -External Lab/books.csv"
path2 = "/content/drive/My Drive/Residency 5 -External Lab/ratings.csv"
path3 = "/content/drive/My Drive/Residency 5 -External Lab/users.csv"
```

```
#Loading data
books1 = pd.read_csv(path1, sep=";", error_bad_lines=False, encoding="latin-1")
books1.columns = ['ISBN', 'bookTitle', 'bookAuthor', 'yearOfPublication', 'publisher', 'imageUrlS',
```

```
b'Skipping line 6452: expected 8 fields, saw 9\nSkipping line 43667: expected 8 fields,
b'Skipping line 92038: expected 8 fields, saw 9\nSkipping line 104319: expected 8 fields
b'Skipping line 144058: expected 8 fields, saw 9\nSkipping line 150789: expected 8 field
b'Skipping line 209388: expected 8 fields, saw 9\nSkipping line 220626: expected 8 field
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarni
  interactivity=interactivity, compiler=compiler, result=result)
```

```
#Loading data
users1 = pd.read_csv(path3, sep=";", error_bad_lines=False, encoding="latin-1")
users1.columns = ['userID', 'Location', 'Age']
```

```
ratings1 = pd.read_csv(path2, sep=";", error_bad_lines=False, encoding="latin-1")
ratings1.columns = ['userID', 'ISBN', 'bookRating']
```

```
books = books1.copy(deep=True)
```

```
users = users1.copy(deep=True)

ratings = ratings1.copy(deep=True)
```

Check no.of records and features given in each dataset

```
print(books.shape)
```

```
(271360, 8)
```

```
print(ratings.shape)
```

```
(1149780, 3)
```

```
print(users.shape)
```

```
(278858, 3)
```

Exploring books dataset

```
books.head()
```

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amaz
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amaz
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amaz
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amaz
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton &amp; Company	http://images.amaz

Drop last three columns containing image URLs which will not be required for analysis

```
books.drop(columns=['imageUrlS','imageUrlM','imageUrlL'],inplace=True)

books.head()
```

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial

yearOfPublication

Check unique values of yearOfPublication

```
books['yearOfPublication'].unique()
```

```
array([2002, 2001, 1991, 1999, 2000, 1993, 1996, 1988, 2004, 1998, 1994,
       2003, 1997, 1983, 1979, 1995, 1982, 1985, 1992, 1986, 1978, 1980,
       1952, 1987, 1990, 1981, 1989, 1984, 0, 1968, 1961, 1958, 1974,
       1976, 1971, 1977, 1975, 1965, 1941, 1970, 1962, 1973, 1972, 1960,
       1966, 1920, 1956, 1959, 1953, 1951, 1942, 1963, 1964, 1969, 1954,
       1950, 1967, 2005, 1957, 1940, 1937, 1955, 1946, 1936, 1930, 2011,
       1925, 1948, 1943, 1947, 1945, 1923, 2020, 1939, 1926, 1938, 2030,
       1911, 1904, 1949, 1932, 1928, 1929, 1927, 1931, 1914, 2050, 1934,
       1910, 1933, 1902, 1924, 1921, 1900, 2038, 2026, 1944, 1917, 1901,
       2010, 1908, 1906, 1935, 1806, 2021, '2000', '1995', '1999', '2004',
       '2003', '1990', '1994', '1986', '1989', '2002', '1981', '1993',
       '1983', '1982', '1976', '1991', '1977', '1998', '1992', '1996',
       '0', '1997', '2001', '1974', '1968', '1987', '1984', '1988',
       '1963', '1956', '1970', '1985', '1978', '1973', '1980', '1979',
       '1975', '1969', '1961', '1965', '1939', '1958', '1950', '1953',
       '1966', '1971', '1959', '1972', '1955', '1957', '1945', '1960',
       '1967', '1932', '1924', '1964', '2012', '1911', '1927', '1948',
       '1962', '2006', '1952', '1940', '1951', '1931', '1954', '2005',
       '1930', '1941', '1944', 'DK Publishing Inc', '1943', '1938',
       '1900', '1942', '1923', '1920', '1933', 'Gallimard', '1909',
       '1946', '2008', '1378', '2030', '1936', '1947', '2011', '2020',
       '1919', '1949', '1922', '1897', '2024', '1376', '1926', '2037'],
      dtype=object)
```

As it can be seen from above that there are some incorrect entries in this field. It looks like Publisher names 'DK Pub
loaded as yearOfPublication in dataset due to some errors in csv file.

Also some of the entries are strings and same years have been entered as numbers in some places. We will try to fi

Check the rows having 'DK Publishing Inc' as yearOfPublication

```
books[(books['yearOfPublication']=='Gallimard') ^ (books['yearOfPublication']=='DK Publishing Inc')]
```

	ISBN	bookTitle	bookAuthor	yearOfPublication	
209538	078946697X	DK Readers: Creating the X-Men, How It All Beg...	2000	DK Publishing Inc	http://images.amazon.com/
		Peuple du ciel,			

Drop the rows having `'DK Publishing Inc'` and `'Gallimard'` as yearOfPublication

```
books.drop([books.index[209538] , books.index[220731], books.index[221678]],inplace=True)
```

```
## Checking the dropped records

books[(books['yearOfPublication']=='Gallimard') ^ (books['yearOfPublication']=='DK Publishing Inc')]
```

ISBN	bookTitle	bookAuthor	yearOfPublication	publisher

```
print(books1.shape)
print(books.shape)
```

```
(271360, 8)
(271357, 5)
```

Change the datatype of yearOfPublication to 'int'

```
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 271357 entries, 0 to 271359
Data columns (total 5 columns):
ISBN                 271357 non-null object
bookTitle            271357 non-null object
bookAuthor           271356 non-null object
yearOfPublication    271357 non-null object
publisher            271355 non-null object
dtypes: object(5)
memory usage: 12.4+ MB
```

```
books['yearOfPublication'] = books['yearOfPublication'].astype(np.int32)
```

```
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 271357 entries, 0 to 271359
Data columns (total 5 columns):
ISBN                 271357 non-null object
bookTitle            271357 non-null object
bookAuthor           271356 non-null object
yearOfPublication    271357 non-null int32
publisher            271355 non-null object
dtypes: int32(1), object(4)
memory usage: 11.4+ MB
```

```
books.dtypes
```

```
ISBN                 object
bookTitle            object
bookAuthor           object
yearOfPublication     int32
publisher            object
dtype: object
```

```
books['yearOfPublication'].unique()
```

```
array([2002, 2001, 1991, 1999, 2000, 1993, 1996, 1988, 2004, 1998, 1994,
       2003, 1997, 1983, 1979, 1995, 1982, 1985, 1992, 1986, 1978, 1980,
       1952, 1987, 1990, 1981, 1989, 1984,    0, 1968, 1961, 1958, 1974,
       1976, 1971, 1977, 1975, 1965, 1941, 1970, 1962, 1973, 1972, 1960,
       1966, 1920, 1956, 1959, 1953, 1951, 1942, 1963, 1964, 1969, 1954,
       1950, 1967, 2005, 1957, 1940, 1937, 1955, 1946, 1936, 1930, 2011,
       1925, 1948, 1943, 1947, 1945, 1923, 2020, 1939, 1926, 1938, 2030,
       1911, 1904, 1949, 1932, 1928, 1929, 1927, 1931, 1914, 2050, 1934,
       1910, 1933, 1902, 1924, 1921, 1900, 2038, 2026, 1944, 1917, 1901,
       2010, 1908, 1906, 1935, 1806, 2021, 2012, 2006, 1909, 2008, 1378,
       1919, 1922, 1897, 2024, 1376, 2037])
```

Drop NaNs in `'publisher'` column

```
books['publisher'].unique()
```

```
array(['Oxford University Press', 'HarperFlamingo Canada',
       'HarperPerennial', ..., 'Tempo', 'Life Works Books', 'Connaught'],
      dtype=object)
```

```
books.dropna(subset=['publisher'],inplace=True)
```

```
## Checking the dropped records

books[(books['publisher']==np.NaN) ]
```

```
    ISBN  bookTitle  bookAuthor  yearOfPublication  publisher
```

```
print(books1.shape)
print(books.shape)
```

```
(271360, 8)
(271355, 5)
```

Exploring Users dataset

```
print(users.shape)
print(users1.shape)
users.head()
```

```
(278858, 3)
(278858, 3)
```

	userID	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

Get all unique values in ascending order for column `Age`

```
pd.DataFrame({'Age':users['Age'].unique()}).sort_values(by='Age',ascending=True)
```

```
...    ...
157  157.0
135  159.0
120  162.0
133  168.0
115  172.0
114  175.0
150  183.0
136  186.0
164  189.0
131  199.0
140  200.0
 95  201.0
151  204.0
144  207.0
155  208.0
116  209.0
129  210.0
117  212.0
110  219.0
161  220.0
153  223.0
142  226.0
149  228.0
145  229.0
 87  230.0
 71  231.0
118  237.0
 88  239.0
101  244.0
  0    NaN

166 rows × 1 columns
```

Age column has some invalid entries like nan, 0 and very high values like 100 and above

Values below 5 and above 90 do not make much sense for our book rating case...hence replace th

```
users.head()
```

	userID	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

```
users['Age'].replace(users[(users['Age']<5)^(users['Age']>90)]['Age'],np.NaN,inplace=True)
```

```
users[(users['Age']<5)^(users['Age']>90)]['Age']
```

```
Series([], Name: Age, dtype: float64)
```

Replace null values in column Age with mean

```
users['Age'].head()
```

```
0     NaN
1    18.0
2     NaN
3    17.0
4     NaN
Name: Age, dtype: float64
```

```
users['Age'].mean()
```

```
34.72384041634689
```

```
users['Age'].replace(np.NaN,users['Age'].mean(),inplace=True)
```

```
users['Age'].head()
```

```
0    34.72384
1    18.00000
2    34.72384
3    17.00000
4    34.72384
Name: Age, dtype: float64
```

Change the datatype of `Age` to `int`

```
users['Age'] =users['Age'].astype(np.int64)
```

```
users['Age'].dtypes
```

```
dtype('int64')
```

```
print(sorted(users.Age.unique()))
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
```

Exploring the Ratings Dataset

check the shape

```
print(ratings.shape)
print(ratings1.shape)
ratings.head()
```

```
(1149780, 3)
(1149780, 3)
```

	userID	ISBN	bookRating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6

```
n_users = users.shape[0]
n_books = books.shape[0]
```

```
print(n_users,n_books)
```

```
278858 271355
```

```
ratings.head(5)
```

	userID	ISBN	bookRating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6

Ratings dataset should have books only which exist in our books dataset. Drop the remaining row

```
ratings.head()
```

	userID	ISBN	bookRating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6

```
books.head()
```

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial

```
users.head()
```

	userID	Location	Age
0	1	nyc, new york, usa	34
1	2	stockton, california, usa	18
2	3	moscow, yukon territory, russia	34
3	4	porto, v.n.gaia, portugal	17
4	5	farnborough, hants, united kingdom	34

```
ratings_books = pd.merge(ratings,books,on='ISBN',how='inner')
```

```
print(books.shape)
print(ratings.shape)
print(ratings_books.shape)
```

```
(271355, 5)
(1149780, 3)
(1031130, 7)
```

```
ratings_books.head()
```

	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	publisher
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
1	2313	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
2	6543	034545104X	0	Flesh Tones: A	M. J. Rose	2002	Ballantine

Ratings dataset should have ratings from users which exist in users dataset. Drop the remaining r

```
ratings_books_users_df = pd.merge(ratings_books,users,on='userID',how='inner')
```

```
ratings_books_users_df.shape
```

```
(1031130, 9)
```

```
print(books.shape)
print(ratings.shape)
print(ratings_books.shape)
```

```
(271355, 5)
(1149780, 3)
(1031130, 7)
```

```
ratings_books_users_df.head()
```

	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	publish
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballanti Boo
1	2313	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballanti Boo
				Ender's Game (Ender	Orson Scott		

```
ratings_books_users_df_cpy = ratings_books_users_df.copy(deep=True)
```

Consider only ratings from 1-10 and leave 0s in column `bookRating`

```
ratings_books_users_df = ratings_books_users_df[ratings_books_users_df['bookRating']!=0]
```

```
ratings_books_users_df.head()
```

	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	publ
1	2313	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine
2	2313	0812533550	9	Ender's Game (Ender Wiggins Saga (Paperback))	Orson Scott Card	1986	Tor
3	2313	0679745580	8	In Cold Blood (Vintage International)	TRUMAN CAPOTE	1994	V

```
ratings_books_users_df_cpy[ratings_books_users_df_cpy['bookRating']==0]['bookRating'].count()
```

```
647291
```

```
print(ratings_books_users_df_cpy.shape)
```

```
(1031130, 9)
```

```
print(ratings_books_users_df.shape)
```

```
(383839, 9)
```

```
## 383839+647291 = 1031130
```

Find out which rating has been given highest number of times

```
ratings_books_users_df.columns
```

```
Index(['userID', 'ISBN', 'bookRating', 'bookTitle', 'bookAuthor',
       'yearOfPublication', 'publisher', 'Location', 'Age'],
      dtype='object')
```

```
ratings_books_users_df.groupby('bookRating').count().sort_values(by='userID',ascending=False)
## rating 8 is given highest number of times - 91804
```

bookRating	userID	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher	Location
8	91804	91804	91804	91803	91804	91804	91804
10	71225	71225	71225	71225	71225	71225	71225
7	66401	66401	66401	66401	66401	66401	66401
9	60776	60776	60776	60776	60776	60776	60776
5	45355	45355	45355	45355	45355	45355	45355
6	31687	31687	31687	31687	31687	31687	31687
4	7617	7617	7617	7617	7617	7617	7617
3	5118	5118	5118	5118	5118	5118	5118
2	2375	2375	2375	2375	2375	2375	2375
1	1481	1481	1481	1481	1481	1481	1481

Collaborative Filtering Based Recommendation Systems

For more accurate results only consider users who have rated atleast 100 books

```
ratings_books_users_df.columns
```

```
Index(['userID', 'ISBN', 'bookRating', 'bookTitle', 'bookAuthor',
       'yearOfPublication', 'publisher', 'Location', 'Age'],
      dtype='object')
```

```
user_grp = ratings_books_users_df.groupby(['userID']).count().sort_values(by=[ 'ISBN', 'bookRating',
       'yearOfPublication', 'publisher', 'Location', 'Age'])
```

```
user_grp.head()
```

	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	publisher	Location
userID							
9	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1

```
userid = user_grp[user_grp['ISBN']>99].index
```

```
len(userid)
```

```
449
```

```
ratings_books_users100_df = ratings_books_users_df.loc[ratings_books_users_df['userID'].isin(userid)
```

```
ratings_books_users100_df.shape
```

```
(103269, 9)
```

```
ratings_books_users_df.shape
```

```
(383839, 9)
```

```
ratings_books_users100_df.head()
```

	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	publisher
43	6543	0446605484	10	Roses Are Red (Alex Cross Novels)	James Patterson	2001	Warner Vision
47	6543	0805062971	8	Fight Club	Chuck Palahniuk	1999	Owl Books

Generating ratings matrix from explicit ratings

Note: since NaNs cannot be handled by training algorithms, replace these by 0, which indicates absence of ratings

```
ratings_books_users100_df.isna().sum()
```

```
userID               0
ISBN                 0
bookRating           0
bookTitle            0
bookAuthor           0
yearOfPublication    0
publisher            0
Location             0
Age                  0
dtype: int64
```

```
ratings_books_users100_df.fillna(0,inplace=True)
```

Generate the predicted ratings using SVD with no.of singular values to be 50

```
pip install surprise
```

```
Requirement already satisfied: surprise in /usr/local/lib/python3.6/dist-packages (0.1)
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (fr
```

```
from collections import defaultdict
from surprise import SVD
from surprise import Dataset
```

```
from sklearn.model_selection import train_test_split

trainDF, tempDF = train_test_split(ratings_books_users100_df, test_size = 0.2, random_state = 100)
```

```
print(trainDF.shape, tempDF.shape)
```

```
(82615, 9) (20654, 9)
```

```
trainDF.head()
```

	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	p
425933	150979	0679460152	9	The Blackstone Chronicles	John Saul	1997	
278665	60244	0393049566	7	Socrates Cafe: A Fresh Taste of Philosophy	Christopher Phillips	2001	
8536	98391	067104222X	9	Dangerous Dilemmas	Evelyn Palfrey	2001	

```
tempDF.head()
```

	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	publ
446631	197659	0842304673	7	The Complete Book of Zingers	Croft M. Pentz	1990	Ty H Publ
135668	184299	0345358791	8	2061: Odyssey Three	Arthur C. Clarke	1991	De I
84242	115003	1400031354	9	Tears of the Giraffe (No.1 Ladies	Alexander McCall	2002	A

```
testDF = tempDF.copy()
```

```
tempDF['bookRating'] = np.nan
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexi
  """Entry point for launching an IPython kernel.
```

```
tempDF.head()
```

	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	publ
446631	197659	0842304673	NaN	The Complete Book of Zingers	Croft M. Pentz	1990	Ty I Publ
135668	184299	0345358791	NaN	2061: Odyssey Three	Arthur C. Clarke	1991	De I
84242	115003	1400031354	NaN	Tears of the Giraffe (No.1 Ladies	Alexander McCall	2002	A

```
testDF = testDF.dropna()
```

```
testDF.head()
```

	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	publ
446631	197659	0842304673	7	The Complete Book of Zingers	Croft M. Pentz	1990	Ty I Publ
135668	184299	0345358791	8	2061: Odyssey Three	Arthur C. Clarke	1991	De I
84242	115003	1400031354	9	Tears of the Giraffe (No.1 Ladies	Alexander McCall	2002	A

```
rtings = pd.concat([trainDF, tempDF]).reset_index()
```

```
rtings.sample(10)
```

	index	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublicatio
27770	786233	123094	0385299397	8.0	Childhood Rising: The Astrology of Your Mother...	Michael Lutin	199
4490	31861	11676	0671748742	5.0	Left to Die	Dan Kurzman	199
34607	66539	275970	0061059072	9.0	The Last Continent (Discworld Novels (Paperback))	Terry Pratchett	200
97213	698388	212965	0821734989	NaN	Forbidden Ecstasy	Janelle Taylor	199
28525	602294	23902	0385490992	6.0	The Street Lawyer	John Grisham	199
84980	193656	153662	0380772574	NaN	Enchanted	Elizabeth Lowell	199
42051	166948	204864	0140186409	10.0	The Grapes of Wrath (20th	John Steinbeck	199

```
rtings.head()
```

	index	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication
0	425933	150979	0679460152	9.0	The Blackstone Chronicles	John Saul	1997
1	278665	60244	0393049566	7.0	Socrates Cafe: A Fresh Taste of Philosophy	Christopher Phillips	2001
2	8536	98391	067104222X	9.0	Dangerous Dilemmas	Evelyn Palfrey	2001

```
rtings['userID'].min()
```

```
2033
```

```
rtings = rtings.drop_duplicates()
rtings.shape
```

```
(103269, 10)
```

```
rtings[(rtings['userID']==2033)#&(rtings['ISBN']=='0393049566')
      ]
```

	index	userID	ISBN	bookRating	bookTitl
126	364083	2033	1891400495	10.0	A Simple Choice : A Practical Guide to Saving .
239	364038	2033	0882710583	8.0	Catholic Children's Bibl
571	363987	2033	0671025554	10.0	What's in a Nam
1424	363969	2033	0451458028	10.0	The Invisible Rin
1518	364004	2033	0716724022	7.0	Physical Chemistr
4143	364066	2033	0895779129	6.0	Foods That Harm, Foods That Heal: An A - . Gu.
4310	363927	2033	0590353403	9.0	Harry Potter and the Sorcerer's Stone (Book 1
6419	364010	2033	0786880007	8.0	Simplify Your Life : 100 Ways to Slow Down and.
7831	364052	2033	0886775639	8.0	Winds of Change (The Mage Winds, Book 2
8796	364063	2033	0886778603	10.0	The Children of Wrath (Renshai Chronicles
8804	364061	2033	0886777593	8.0	Prince of Demons (Renshai Chronicles
8813	364035	2033	0874936225	9.0	Pediatric Basic Life Support, 1997-199
9198	364014	2033	0812090381	9.0	Maine Coon Cats: Everything About Purchase Ca.
10046	363916	2033	0316779032	10.0	The Discipline Book: How to Have a Better Beha.
10249	364041	2033	0886773784	10.0	Arrows of the Queen (The Heralds of Valdemar,.
10373	364042	2033	0886774004	10.0	Arrow's Fall (The Heralds of Valdemar, Book 3
10719	363934	2033	0060248025	10.0	Falling U
10799	363948	2033	0192800493	6.0	The Oxford Companion to Ches
11929	363968	2033	0451457781	8.0	Treachery and Treaso
15529	363933	2033	0030020786	7.0	Principles of Instrumental Analysi

```
17678  363944  2033  0133502813  10.0                                          Chemistr
19801  364040  2033  0886773776  10.0       Arrow's Flight (The Heralds of Valdemar, Book 2
20090  363920  2033  0553573136   8.0                                         Couplehoo
21601  364058  2033  0886777151   9.0       Prince of Demons (Renshai Chronicles/Micke
                                                                                   Zu.
23702  364032  2033  0812575717  10.0                                     Ender's Shadov
24789  363896  2033  0812533550  10.0     Ender's Game (Ender Wiggins Saga (Paperback)
24980  363905  2033  0812532635   8.0         The Ships of Earth : Homecoming: Volume
                                                                                (Hom.
26189  364062  2033  0886777739  10.0              Oathblood (Vows and Honor, Book 3
26445  364050  2033  0886775205   9.0              The Western Wizard (Renshai Trilogy
26663  363989  2033  0671524313   6.0               The Girlfriends' Guide to Pregnanc
  ...     ...   ...         ...   ...                                                .
83508  363899  2033  0765342987   NaN                                         Kushiel's Da
84631  364044  2033  0886774144   NaN            The Oathbound (Vows and Honor, Book 1
84748  364018  2033  0812520157   NaN         Speaker for Dead Ender #2: Valorous (Ende
                                                                                 Wig.
84798  364025  2033  0812532961   NaN                 Earthfall (Homecoming (Paperback)
85716  363921  2033  0886774632   NaN                      By the Sword (Kerowyn's Tale
85765  363956  2033  0375815260   NaN                  Charlie and the Chocolate Factor
85910  363936  2033  0060256737   NaN                                    A Light in the Atti
86190  364024  2033  0812532619   NaN                                    The Call of Eart
86837  363999  2033  0698119509   NaN                                        Spindle's En
```

88298	364037	2033	0880887575	NaN	Flowers for My Friend (Peter Pauper Petite Ser
89016	363953	2033	0316542377	NaN	Toilet Learning : The Picture Book Technique f...
89287	364057	2033	0886777127	NaN	Storm Rising (Mage Storms Trilogy
89816	364067	2033	0912500522	NaN	Mothering Your Nursing Toddle
90889	363990	2033	0671676253	NaN	The ABYS
91961	363957	2033	0380775123	NaN	Tiger Burning Brigh
93660	363946	2033	0142500135	NaN	Treasure at the Heart of the Tanglewoo
94240	363965	2033	0451456718	NaN	Daughter of the Blood (Black Jewels Trilogy
94423	364019	2033	0812521358	NaN	Hart's Hop
95332	364070	2033	0920668372	NaN	Love You Foreve
95979	364029	2033	0812533658	NaN	The Changed Man (Changed Mar
96577	364055	2033	0886776589	NaN	Beyond Ragnarok (Renshai Chronicles, Vol 1
97036	363996	2033	0688161162	NaN	Easy to Love, Difficult to Discipline: The Sev...
97650	364078	2033	1559274581	NaN	James Herriot's Animal Storie
97800	364021	2033	0812523679	NaN	Monkey Sonata
98260	363939	2033	0061020427	NaN	Sword and Shadow (Sword in Exile, Book 3
99019	364003	2033	0716723980	NaN	Inorganic Chemistr
99579	363961	2033	0399523308	NaN	The Girlfriends' Guide to Surviving the First ...
100294	363942	2033	0061056294	NaN	Kingmaker's Sword (The Rune Blade Trilogy Boo...
100378	363938	2033	0061020419	NaN	King of Shadows (Sword in Exile, Book 2
101007	363930	2033	0836218256	NaN	Something Under the Bed Is Droolin

129 rows × 10 columns

```
R_df = rtings.pivot(index = 'userID', columns = 'ISBN', values = 'bookRating').fillna(0)
```

```
R_df.head()
```

ISBN	0000913154	0001046438	000104687X	0001047213	0001047973	000104799X	00010480
userID							
2033	0.0	0.0	0.0	0.0	0.0	0.0	C
2110	0.0	0.0	0.0	0.0	0.0	0.0	C
2276	0.0	0.0	0.0	0.0	0.0	0.0	C
4017	0.0	0.0	0.0	0.0	0.0	0.0	C
4385	0.0	0.0	0.0	0.0	0.0	0.0	C

5 rows × 66572 columns

```
R_df.index
```

```
Int64Index([  2033,   2110,   2276,   4017,   4385,   5582,   6242,   6251,
              6543,   6575,
            ...
            269566, 270713, 271448, 271705, 273113, 274061, 274301, 275970,
            277427, 278418],
           dtype='int64', name='userID', length=449)
```

```
R_df.get_value(2033,'0451457781')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarning: get_value
  """Entry point for launching an IPython kernel.
8.0
```

```
R_df.shape
```

```
(449, 66572)
```

```
rtings['userID'].nunique()
```

```
449
```

```
rtings['ISBN'].nunique()
```

```
66572
```

```
from scipy.sparse.linalg import svds
```

```
U, sigma, Vt = svds(R_df, k = 50)
```

```
sigma
```

```
array([131.07954208, 132.44479902, 132.61470995, 133.96010817,
       134.94232624, 136.38117803, 137.0634911 , 138.04647807,
       140.45935247, 141.29908114, 142.26811037, 143.88305269,
       144.27243066, 144.93753168, 149.39109893, 149.62291223,
       149.94512384, 152.15710138, 152.98116567, 154.23600256,
       155.64958852, 156.98587955, 158.30450983, 161.41139495,
       164.36235669, 164.60938522, 166.22369888, 168.8872909 ,
       173.19509942, 174.99507662, 176.37245022, 178.41205733,
       180.20327794, 181.26833216, 184.19621481, 186.26397001,
       190.17666439, 194.12064112, 202.52424067, 206.23585733,
       210.1876945 , 219.80287636, 223.09823012, 232.70628393,
       237.36014895, 252.56483856, 257.35846413, 338.84909015,
       567.12180411, 605.76299262])
```

```
sigma = np.diag(sigma)
```

```
sigma
```

```
array([[131.07954208,   0.        ,   0.        , ...,   0.        ,
          0.        ,   0.        ],
       [  0.        , 132.44479902,   0.        , ...,   0.        ,
          0.        ,   0.        ],
       [  0.        ,   0.        , 132.61470995, ...,   0.        ,
          0.        ,   0.        ],
       ...,
       [  0.        ,   0.        ,   0.        , ..., 338.84909015,
          0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        , ...,   0.        ,
        567.12180411,   0.        ],
       [  0.        ,   0.        ,   0.        , ...,   0.        ,
          0.        , 605.76299262]])
```

```
all_users_predicted_ratings = np.dot(np.dot(U, sigma), Vt)
```

```
preds_df = pd.DataFrame(all_users_predicted_ratings, columns = R_df.columns)
```

```
# preds_df.index.values
```

```
user_ids = R_df.index
```

```
pred_ids = preds_df.index.values
```

```
user_pred_map = dict(zip(user_ids,pred_ids))


################################### RECOMMENDATION ALGORITHM ##########################################


def recommend_movies(userID,Recommendation_count):
  user_ids = R_df.index
  pred_ids = preds_df.index.values
  user_pred_map = dict(zip(user_ids,pred_ids))


  for user_ids, pred_row_number in user_pred_map.items():
    if user_ids == userID:
      pred_row_number
      # print(pred_row_number)

      user_pred_total = preds_df.loc[pred_row_number,:].sort_values(ascending=False)

      sorted_user_predictions = pd.DataFrame({'Predicted_Ratings':preds_df.loc[user_row_number].sort
      ## sorted_user_predictions.head()

      sorted_user_predictions.reset_index(inplace=True)
      ## sorted_user_predictions.head()

      ##  sorted_user_predictions.shape
      sorted_user_predictions_all = pd.merge(sorted_user_predictions,rtings,on='ISBN',how='inner')

      ##  sorted_user_predictions_all.shape
      ## Total_Predictions = sorted_user_predictions_all.shape[0]
      ## Total_Predictions

      Books_Rated = sorted_user_predictions_all[sorted_user_predictions_all['userID']==user_ids].dro

      Books_Not_Rated = sorted_user_predictions_all[((sorted_user_predictions_all['userID']==user_id
      ##  Books_Not_Rated.shape[0]

      Books_Not_Rated_Unique = Books_Not_Rated[['Predicted_Ratings','bookTitle','bookAuthor', 'yearO

      ## Recommendation_count

      Recommendation_result = Books_Not_Rated_Unique.sort_values(by='Predicted_Ratings',ascending=Fa

      print('UserID:- {0} , has already rated {1} books.'.format(user_ids, Books_Rated.shape[0]))
      print('Recommending the highest {0} predicted different ratings books not already rated by use

      return(Recommendation_result)
```

```
np.array(sorted(rtings['userID'].unique()))
```

```
array([  2033,   2110,   2276,   4017,   4385,   5582,   6242,   6251,
         6543,   6575,   7286,   7346,   8067,   8245,   8681,   8890,
        10560,  11676,  11993,  12538,  12824,  12982,  13552,  13850,
        14422,  15408,  15418,  16634,  16795,  16966,  17950,  19085,
        21014,  23768,  23872,  23902,  25409,  25601,  25981,  26535,
        26544,  26583,  28591,  28634,  29259,  30276,  30511,  30711,
        30735,  30810,  31315,  31556,  31826,  32773,  33145,  35433,
        35836,  35857,  35859,  36299,  36554,  36606,  36609,  36836,
        36907,  37644,  37712,  37950,  38023,  38273,  38281,  39281,
        39467,  40889,  40943,  43246,  43910,  46398,  47316,  48025,
        48494,  49144,  49889,  51883,  52199,  52350,  52584,  52614,
        52917,  53220,  55187,  55490,  55492,  56271,  56399,  56447,
        56554,  56959,  59172,  60244,  60337,  60707,  63714,  63956,
        65258,  66942,  67840,  68555,  69078,  69389,  69697,  70415,
        70594,  70666,  72352,  73681,  75591,  75819,  76151,  76223,
        76499,  76626,  78553,  78783,  78834,  78973,  79441,  81492,
        81560,  83287,  83637,  83671,  85526,  85656,  86189,  86947,
        87141,  87555,  88283,  88677,  88693,  88733,  89602,  91113,
        92652,  92810,  93047,  93363,  93629,  94242,  94347,  94853,
        94951,  95010,  95359,  95902,  95932,  96448,  97754,  97874,
        98391,  98758, 100459, 100906, 101209, 101606, 101851, 102359,
       102647, 102702, 102967, 104399, 104636, 105028, 105517, 105979,
       106007, 107784, 107951, 109574, 109901, 109955, 110483, 110912,
       110934, 110973, 112001, 113270, 113519, 114368, 114868, 114988,
       115002, 115003, 116599, 117384, 120565, 122429, 122793, 123094,
       123608, 123883, 123981, 125519, 125774, 126492, 126736, 127200,
       127359, 128835, 129074, 129716, 129851, 130554, 130571, 132492,
       132836, 133747, 134434, 135149, 135265, 136010, 136139, 136348,
       136382, 138578, 138844, 140000, 140358, 141902, 142524, 143175,
       143253, 143415, 145449, 146113, 146348, 147847, 148199, 148258,
       148744, 148966, 149907, 149908, 150979, 153662, 156150, 156269,
       156300, 156467, 157247, 157273, 158226, 158295, 158433, 159506,
       160295, 162052, 162639, 162738, 163759, 163761, 163804, 163973,
       164096, 164323, 164533, 164828, 164905, 165308, 165319, 165758,
       166123, 166596, 168047, 168245, 169682, 170513, 170634, 171118,
       172030, 172742, 172888, 173291, 173415, 174304, 174892, 177072,
       177432, 177458, 178522, 179718, 179978, 180378, 180651, 181176,
       182085, 182086, 182993, 183958, 183995, 184299, 184532, 185233,
       185384, 187145, 187256, 187517, 189139, 189334, 189835, 189973,
       190708, 190925, 193458, 193560, 193898, 194600, 196077, 196160,
       196502, 197659, 199416, 200226, 201290, 203240, 204864, 205735,
       205943, 206534, 207782, 208406, 208671, 209516, 210485, 211426,
       211919, 212965, 214786, 216012, 216444, 216683, 217106, 217318,
       217740, 218552, 218608, 219546, 219683, 222204, 222296, 223087,
       223501, 224349, 224525, 224646, 224764, 225087, 225199, 225232,
       225595, 225763, 226965, 227250, 227447, 227520, 227705, 229011,
       229329, 229551, 229741, 230522, 231210, 232131, 232945, 233911,
       234359, 234828, 235105, 235282, 235935, 236058, 236283, 236340,
       236757, 236948, 239584, 239594, 240144, 240403, 240543, 240567,
       240568, 241198, 241666, 241980, 242006, 242083, 242409, 242465,
       244627, 244685, 245410, 245827, 246311, 247429, 247447, 248718,
       249894, 250405, 250709, 251394, 251843, 251844, 252695, 252820,
       254206, 254465, 254899, 255489, 257204, 258152, 258185, 258534,
       261105, 261829, 262998, 264031, 264082, 264321, 264525, 265115,
       265313, 265889, 266056, 266226, 268110, 268300, 268932, 269566,
       270713, 271448, 271705, 273113, 274061, 274301, 275970, 277427,
       278418])
```

```
#x = recommend_movies(2033,12)
#x.index
#y = recommend_movies(278418,12)
#y.index

#print(x.index.intersection(y.index),len(x.index.intersection(y.index)))
```

```
recommend_movies(2110,12)
```

```
UserID:- 2110 , has already rated 85 books.
Recommending the highest 12 predicted different ratings books not already rated by user
```

	Predicted_Ratings	bookTitle	bookAuthor	yearOfPu
0	0.285622	Purity in Death	J.D. Robb	
15	0.253016	Face the Fire (Three Sisters Island Trilogy)	Nora Roberts	
29	0.236891	Dance upon the Air (Three Sisters Island Trilogy)	Nora Roberts	
48	0.212176	Jewels of the Sun (Irish Trilogy)	Nora Roberts	
63	0.206198	Heart of the Sea (Irish Trilogy)	Nora Roberts	
81	0.204699	The Lovely Bones: A Novel	Alice Sebold	
152	0.196262	Tears of the Moon (Irish Trilogy)	Nora Roberts	
169	0.188376	Witness in Death (Eve Dallas Mysteries (Paperb...	J. D. Robb	
180	0.183306	Ceremony in Death (Eve Dallas Mysteries (Paper...	J. D. Robb	
188	0.178672	Summer Pleasures	Nora Roberts	
196	0.178214	Heaven and Earth (Three Sisters Island Trilogy)	Nora Roberts	
208	0.177991	Carolina Moon	Nora Roberts	

```
recommend_movies(2033,12)
```