About Book Crossing Dataset

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

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 collab

```
import pandas as pd
import numpy as np
import os
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

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

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

Execute the below cell to load the datasets

```
#Loading data
books = pd.read_csv("/content/drive/My Drive/20oct/books.csv", sep=";", error_bad_lines=False, er
books.columns = ['ISBN', 'bookTitle', 'bookAuthor', 'yearOfPublication', 'publisher', 'imageUrlS'

users = pd.read_csv('/content/drive/My Drive/20oct/users.csv', sep=';', error_bad_lines=False, er
users.columns = ['userID', 'Location', 'Age']

ratings = pd.read_csv('/content/drive/My Drive/20oct/ratings.csv', sep=';', error_bad_lines=False
ratings.columns = ['userID', 'ISBN', 'bookRating']
```

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

```
ratings
```

28	276754	0684867621	8
29	276755	0451166892	5
...	...	...	...
1149750	276690	0312970188	0
1149751	276690	0440439884	0
1149752	276690	0590453653	0
1149753	276690	0590453688	0
1149754	276690	0590455419	0
1149755	276690	0590464116	0
1149756	276690	0590581066	0
1149757	276690	0590907301	0
1149758	276697	8445072897	0
1149759	276704	0152022597	0
1149760	276704	0312873115	0
1149761	276704	0345386108	6
1149762	276704	0380796155	5
1149763	276704	0395404258	0
1149764	276704	0425060772	0
1149765	276704	0440206529	0
1149766	276704	0441007813	0
1149767	276704	0446353957	0
1149768	276704	0446605409	0
1149769	276704	059032120X	0
1149770	276704	0679752714	0
1149771	276704	0743211383	7
1149772	276704	080410526X	0
1149773	276704	0806917695	5
1149774	276704	0876044011	0
1149775	276704	1563526298	9
1149776	276706	0679447156	0
1149777	276709	0515107662	10
1149778	276721	0590442449	10
1149779	276723	05162443314	8

1149780 rows × 3 columns

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

```
(271360, 8)
```

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)
```

```
books.head()
```

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

Exploring books dataset

```
books.head()
```

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.a
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.a
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.a
		Flu: The Story of the Pandemic...	Gina Bari		Farrar Straus	

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

Barber

```
books.drop(['imageUrlS'], axis=1, inplace=True)
```

```
books.drop(['imageUrlM'], axis=1, inplace=True)
```

```
books.drop(['imageUrlL'], axis=1, inplace=True)
```

```
books.head()
```

	ISBN	bookTitle	bookAuthor	yearOfPublicat
0	0195153448	Classical Mythology	Mark P. O. Morford	2
1	0002005018	Clara Callan	Richard Bruce Wright	2
2	0060973129	Decision in Normandy	Carlo D'Este	1
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1

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',
```

As it can be seen from above that there are some incorrect entries in this field. It looks like Publisher names 'DK have been incorrectly 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 t questions.

Check the rows having 'DK Publishing Inc' as yearOfPublication

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

```
books = books[books.yearOfPublication != 'DK Publishing Inc']
```

```
books = books[books.yearOfPublication != 'Gallimard']
```

```
books
```

271337	0971854825	Guide to Living a G...	Marilyn Graman	20
271338	0316640786	Christie's Collectables: Blue and White China ...	Paul Tippett	19
271339	3257217323	Schmatz. Oder Die Sackgasse.	Hans Werner Kettenbach	20
271340	3596156904	Amok.	Emmanuel Carrere	20
271341	1874166633	Introducing Nietzsche (Foundations in Children...	Laurence Gane	19
271342	0130897930	Core Web Programming (2nd Edition)	Marty Hall	20
271343	020130998X	The Unified Modeling Language Reference Manual...	James Rumbaugh	19
271344	2268032019	Petite histoire de la dÃ?Â©sinformation	Vladimir Volkoff	19
271345	0684860112	Driving to Detroit: Memoirs of a Fast Woman	Lesley Hazleton	19
271346	0395264707	Dreamsnake	Vonda N. McIntyre	19
271347	3442150663	Der Mossad.	Victor Ostrovsky	20
271348	0231128444	Slow Food(The Case For Taste)	Carlo Petrini	20
271349	0520242335	Strong Democracy : Participatory Politics for ...	Benjamin R. Barber	20
271350	0762412119	Burpee Gardening Cyclopedia: A Concise, Up to ...	Allan Armitage	20
271351	1582380805	Tropical Rainforests: 230 Species in Full Colo...	Allen M., Ph.D. Young	20
271352	1845170423	Cocktail Classics	David Biggs	20
271353	014002803X	Anti Death League	Kingsley Amis	19
271354	0449906736	Flashpoints: Promise and Peril in a New World	Robin Wright	19
271355	0440400988	There's a Bat in Bunk Five	Paula Danziger	19
271356	0525447644	From One to One Hundred	Teri Sloat	19
271357	006008667X	Lily Dale : The True Story of the Town that Ta...	Christine Wicker	20
271358	0192126040	Republic (World's Classics)	Plato	19
271359	0767409752	A Guided Tour of Rene Descartes' Meditations o...	Christopher Biffle	20

271357 rows × 5 columns

```
books.dtypes
```

Change the datatype of yearOfPublication to 'int'

```
books.dtypes
```

```
ISBN                    object
bookTitle               object
bookAuthor              object
yearOfPublication       object
```

```
books["yearOfPublication"] = pd.to_numeric(books["yearOfPublication"])
```

```
books.dtypes
```

```
ISBN                    object
bookTitle               object
bookAuthor              object
yearOfPublication        int64
publisher               object
dtype: object
```

Drop NaNs in `'publisher'` column

```
books= books.dropna()
```

```
books.shape
```

```
(271354, 5)
```

Exploring Users dataset

```
users
```

28	29	cuernavaca, alabama, mexico	19.0
29	30	anchorage, alaska, usa	24.0
...	...	...	...
278828	278829	boise, idaho, usa	NaN
278829	278830	springield, virginia, usa	28.0
278830	278831	anchorage, alaska, usa	NaN
278831	278832	new smyrna beach, florida, usa	62.0
278832	278833	hanoi, australian capital territory, vietnam	25.0
278833	278834	essen, england, germany	NaN
278834	278835	karachi, sindh, pakistan	18.0
278835	278836	des moines, washington, usa	47.0
278836	278837	taiyuan, shanxi, china	NaN
278837	278838	massillon, ohio, usa	15.0
278838	278839	austin, texas, usa	NaN
278839	278840	encinitas, california, usa	45.0
278840	278841	llangollen, denbighshire county, united kingdom.	NaN
278841	278842	perth, western australia, australia	NaN
278842	278843	pismo beach, california, usa	28.0
278843	278844	st. paul, minnesota, usa	28.0
278844	278845	järvenpää, uusimaa, finland	NaN
278845	278846	toronto, ontario, canada	23.0
278846	278847	brooklyn, new york, usa	NaN
278847	278848	köln, nordrhein-westfalen, germany	NaN
278848	278849	georgetown, ontario, canada	23.0
278849	278850	sergnano, lombardia, italy	NaN
278850	278851	dallas, texas, usa	33.0
278851	278852	brisbane, queensland, australia	32.0
278852	278853	stranraer, n/a, united kingdom	17.0
278853	278854	portland, oregon, usa	NaN
278854	278855	tacoma, washington, united kingdom	50.0
278855	278856	brampton, ontario, canada	NaN
278856	278857	knoxville, tennessee, usa	NaN
278857	278858	dublin, n/a, ireland	NaN

278858 rows × 3 columns

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

```
(278858, 3)
```

	userID	Location	Age
0	1	nyc, new york, usa	NaN

Get all unique values in ascending order for column Age

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-107-9aa6453dffb5> in <module>()
----> 1 sorted=  sorted(users.Age.unique())

TypeError: 'list' object is not callable
```

SEARCH STACK OVERFLOW

```
sorted
```

```
        [nan,
         0.0,
         1.0,
         2.0,
         3.0,
         4.0,
         5.0,
         6.0,
         7.0,
         8.0,
         9.0,
         10.0,
         11.0,
         12.0,
         13.0,
         14.0,
         15.0,
         16.0,
         17.0,
         18.0,
         19.0,
         20.0,
         21.0,
         22.0,
         23.0,
         24.0,
         25.0,
         26.0,
         27.0,
         28.0,
         29.0,
         30.0,
         31.0,
         32.0,
         33.0,
         34.0,
         35.0,
         36.0,
         37.0,
         38.0,
         39.0,
         40.0,
         41.0,
         42.0,
         43.0,
         44.0,
         45.0,
         46.0,
         47.0,
         48.0,
         49.0,
         50.0,
         51.0,
         52.0,
         53.0,
         54.0,
         55.0,
         56.0,
         57.0,
         58.0,
         59.0,
         60.0,
         61.0,
         62.0,
         63.0,
         64.0,
         65.0,
         66.0,
         67.0,
         68.0,
         69.0,
         70.0,
         71.0,
         72.0,
         73.0,
         74.0,
         75.0,
         76.0,
         77.0,
         78.0,
         79.0,
         80.0,
         81.0,
         82.0,
         83.0,
         84.0,
         85.0,
         86.0,
         87.0,
         88.0,
         89.0,
         90.0,
         91.0,
         92.0,
         93.0,
         94.0,
         95.0,
         96.0,
         97.0,
         98.0,
         99.0,
         100.0,
         101.0,
         102.0,
         103.0,
         104.0,
         105.0,
         106.0,
         107.0,
         108.0,
         109.0,
         110.0,
         111.0,
         113.0,
         114.0,
         115.0,
         116.0,
         118.0,
         119.0,
         123.0,
         124.0,
         127.0,
         128.0,
         132.0,
         133.0,
         136.0,
         137.0,
         138.0,
         140.0,
         141.0,
         143.0,
         146.0,
         147.0,
         148.0,
         151.0,
         152.0,
         156.0,
         157.0,
         159.0,
         162.0,
         168.0,
         172.0,
         175.0,
         183.0,
         186.0,
         189.0,
         199.0,
         200.0,
         201.0,
         204.0,
         207.0,
         208.0,
         209.0,
         210.0,
         212.0,
         219.0,
         220.0,
         223.0,
         226.0,
         228.0,
         229.0,
         230.0,
         231.0,
         237.0,
         239.0,
         244.0]
```

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 replac

```
users.loc[(users.Age>90) | (users.Age<5), 'Age'  ]=np.NaN
```

Replace null values in column `Age` with mean

```
users.Age=users.Age.fillna(users.Age.mean())
```

Change the datatype of `Age` to `int`

```
users.Age=users.Age.astype(np.int32)
```

Exploring the Ratings Dataset

check the shape

```
ratings.shape
```

```
(1149780, 3)
```

```
books.shape
```

```
(271354, 5)
```

```
users.shape
```

```
(278858, 3)
```

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

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

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

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

```
ratings_new=ratings[ratings.ISBN.isin(books.ISBN)]
```

```
ratings_new=ratings_new[ratings_new.userID.isin(users.userID)]
```

```
ratings.shape
```

```
(1149780, 3)
```

```
ratings_new.shape
```

```
(1031129, 3)
```

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

```
ratings.bookRating.unique()
```

```
array([ 0,  5,  3,  6,  8,  7, 10,  9,  4,  1,  2])
```

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

```
ratings_explicit=ratings_new[ratings_new.bookRating !=0]
ratings_implicit=ratings_new[ratings_new.bookRating ==0]
```

```
ratings_explicit
```

```
      3   276729   052165615X    3
      4   276729   0521795028    6
      8   276744   038550120X    7
     16   276747   0060517794    9
     19   276747   0671537458    9
     20   276747   0679776818    8
     21   276747   0943066433    7
     23   276747   1885408226    7
     24   276748   0747558167    6
     27   276751   3596218098    8
     28   276754   0684867621    8
     29   276755   0451166892    5
     33   276762   0380711524    5
     44   276762   3453092007    8
     59   276772   0553572369    7
     61   276772   3499230933   10
     62   276772   3596151465   10
     66   276774   3442136644    9
     77   276786   8437606322    8
     81   276786   8478442588    6
     83   276788   0345443683    8
     84   276788   043935806X    7
     85   276788   055310666X   10
     86   276796   0330332775    5
     88   276798   0006379702    5
     90   276798   3442131340    7
     97   276798   3548603203    6
    105   276800   1562827898    7
    109   276804   0440498058    8
    ...      ...          ...  ...
1149685   276688   0394748646   10
1149690   276688   0425156737    2
1149697   276688   044661193X    7
1149703   276688   0553074938    8
```

1149703	276688	0553074938	8
1149704	276688	0553074946	9
1149705	276688	0553088467	8
1149709	276688	0553089234	10
1149711	276688	0553566040	6
1149713	276688	0553572512	7
1149714	276688	0553575090	7
1149715	276688	0553575104	6
1149717	276688	0671015591	2
1149719	276688	0671563149	6
1149728	276688	0684195569	10
1149729	276688	0684804484	10
1149738	276688	0688156134	8
1149739	276688	0743202694	10
1149741	276688	0786011157	7
1149743	276688	0836218655	10
1149744	276688	0836236688	10
1149745	276688	0892966548	10
1149746	276688	1551669315	6
1149747	276688	1575660792	7
1149761	276704	0345386108	6
1149762	276704	0380796155	5
1149771	276704	0743211383	7
1149773	276704	0806917695	5
1149775	276704	1563526298	9
1149777	276709	0515107662	10
1149778	276721	0590442449	10

383838 rows × 3 columns

```
users_exp_ratings=users[users.userID.isin(ratings_explicit.userID)]
```

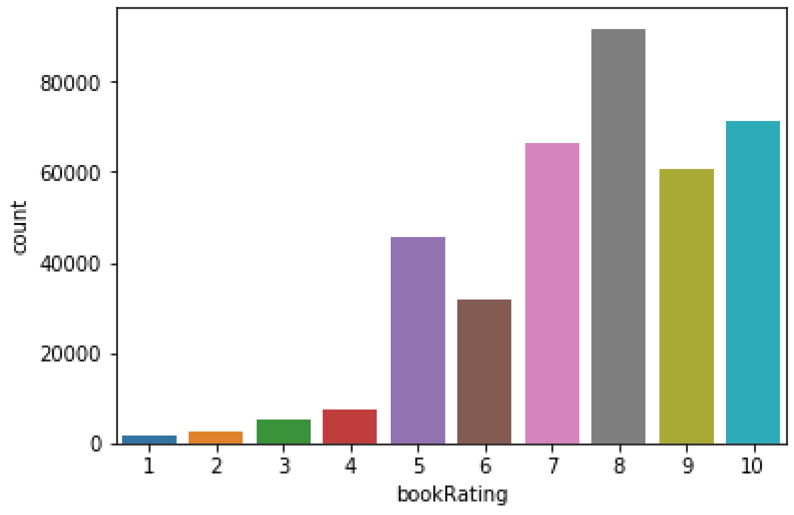
```
users_imp_ratings=users[users.userID.isin(ratings_implicit.userID)]
```

```
users_exp_ratings.shape
```

```
(68091, 3)
```

Find out which rating has been given highest number of times

```
sns.countplot(data=ratings_explicit , x='bookRating')
plt.show()
```



Collaborative Filtering Based Recommendation Systems

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

```
counts1=ratings_explicit['userID'].value_counts()
```

```
ratings_explicit=ratings_explicit[ratings_explicit['userID'].isin(counts1[counts1 >=100 ].index)]
```

```
ratings_explicit
```

```
   1458  277427  003008685X   8
   1461  277427  0060006641  10
   1465  277427  0060542128   7
   1474  277427  0061009059   9
   1477  277427  0062507109   8
   1483  277427  0132220598   8
   1488  277427  0140283374   6
   1490  277427  014039026X   8
   1491  277427  0140390715   7
   1494  277427  0141439742   8
   1497  277427  0152050167  10
   1501  277427  0201000822  10
   1506  277427  0310435706  10
   1509  277427  0312944691   8
   1522  277427  0316776963   8
   1543  277427  0345413903  10
   1554  277427  0375408886   9
   1560  277427  0375751513   9
   1564  277427  0380702843   8
   1570  277427  0380791978   9
   1571  277427  038081904X   9
   1578  277427  0385424736   9
   1581  277427  0385486804   9
   1583  277427  0385503857   9
   1584  277427  0385504209   8
   1586  277427  039304016X   8
   1591  277427  0394738136   7
   1592  277427  0394862147   7
   1599  277427  0399149562   7
    ...     ...         ...  ...
1147394  275970  0877017883  10
1147406  275970  0896086011  10
1147411  275970  0915230151   7
1147419  275970  0945774109  10
```