```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
book_rat=pd.read_csv('/content/drive/My Drive/book_ratings.dat',sep='\t')
book_info = pd.read_csv('/content/drive/My Drive/bo_info.txt',sep='\t')
user_info = pd.read_csv('/content/drive/My Drive/users_info.dat',sep='\t')
```

Double-click (or enter) to edit

```
book_info.info()

book_info.columns = book_info.columns.str.strip().str.lower().str.replace('-', '_')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17384 entries, 0 to 17383
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Book_ID              17384 non-null  int64 
 1   ISBN                 17384 non-null  object
 2   Book-Title           17384 non-null  object
 3   Book-Author          17383 non-null  object
 4   Year-Of-Publication  17384 non-null  object
 5   Publisher            17384 non-null  object
 6   Image-URL-S          17384 non-null  object
 7   Image-URL-M          17382 non-null  object
 8   Image-URL-L          17383 non-null  object
 9   Unnamed: 9           973 non-null    object
 10  Unnamed: 10          36 non-null     object
 11  Unnamed: 11          5 non-null      object
 12  Unnamed: 12          1 non-null      object
dtypes: int64(1), object(12)
memory usage: 1.7+ MB
```

```
book_info.isna().sum()
```

```
book_id                    0
isbn                       0
book_title                 0
book_author                1
year_of_publication        0
publisher                  0
image_url_s                0
image_url_m                2
image_url_l                1
unnamed: 9             16411
unnamed: 10            17348
unnamed: 11            17379
```

```
unnamed: 12           17383
dtype: int64
```

```
book_info=book_info.iloc[:,:9]
```

```
book_info=book_info.dropna()
```

```
book_info.isna().sum() #all Null values removed
```

```
book_id                0
isbn                   0
book_title             0
book_author            0
year_of_publication    0
publisher              0
image_url_s            0
image_url_m            0
image_url_l            0
dtype: int64
```

```
book_info.head(5)
```

	book_id	isbn	book_title	book_author	year_of_publication	publisher	image_url_s	image_url_m	
0	1	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/0060973129.0...	http://images.a
1	2	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton &amp	Company	http://images.amazon.com/images/P/0393045218.0...	http://images.a
2	3	425176428	What If?: The World's Foremost Military Histor...	Robert Cowley	2000	Berkley Publishing Group	http://images.amazon.com/images/P/0425176428.0...	http://images.amazon.com/images/P/0425176428.0...	http://images.a
3	4	452264464	Beloved (Plume Contemporary Fiction)	Toni Morrison	1994	Plume	http://images.amazon.com/images/P/0452264464.0...	http://images.amazon.com/images/P/0452264464.0...	http://images.a
4	5	609804618	Our Dumb Century: The Onion Presents 100 Years...	The Onion	1999	Three Rivers Press	http://images.amazon.com/images/P/0609804618.0...	http://images.amazon.com/images/P/0609804618.0...	http://images.a

```
book_info['image_url_l'][2]
```

```
'http://images.amazon.com/images/P/0425176428.01.LZZZZZZZ.jpg'
```

```
book_info.publisher = book_info.publisher.str.replace('&amp', '&', regex=False)
```

```
book_info=book_info.drop('image_url_s',axis=1)
book_info=book_info.drop('image_url_l',axis=1)
book_info=book_info.drop('image_url_m',axis=1)
```

```
book_info.head(5)
```

	book_id	isbn	book_title	book_author	year_of_publication	publisher
0	1	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial
1	2	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton &
2	3	425176428	What If?: The World's Foremost Military Histor...	Robert Cowley	2000	Berkley Publishing Group
3	4	452264464	Beloved (Plume Contemporary Fiction)	Toni Morrison	1994	Plume
4	5	609804618	Our Dumb Century: The Onion Presents 100 Years...	The Onion	1999	Three Rivers Press

```
len(book_info.book_id.unique())
```

```
# In[132]:
```

```
book_info[book_info['book_id']==11960]
```

	book_id	isbn	book_title	book_author	year_of_publication	publisher
11959	11960	60192356	Other Worlds	Barbara Michaels	1999	HarperCollins Publishers

Double-click (or enter) to edit

```
book_rat.info()

book_rat.columns = book_rat.columns.str.strip().str.lower().str.replace('-', '_') # clean column names
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62656 entries, 0 to 62655
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   user    62656 non-null  int64
 1   item    62656 non-null  int64
 2   rating  62656 non-null  int64
dtypes: int64(3)
memory usage: 1.4 MB
```

```
book_rat.isna().sum() #NO NULL VALUES PRESENT
```

```
user      0
item      0
rating    0
dtype: int64
```

```
book_rat.head(10)
```

	user	item	rating
0	1	6264	7
1	1	4350	7
2	1	6252	5
3	1	202	9
4	1	6266	6
5	1	4810	5
6	1	6251	9
7	1	160	9
8	1	161	8
9	1	631	10

```
len(book_rat.item.unique())
```

```
14684
```

```
book_rat[book_rat['user']==2945]
```

	user	item	rating
62626	2945	4303	8
62627	2945	13214	7
62628	2945	956	7
62629	2945	8524	6
62630	2945	12915	6
62631	2945	6881	7
62632	2945	3701	4
62633	2945	12005	6
62634	2945	1957	7
62635	2945	366	4
62636	2945	4475	8
62637	2945	23	9
62638	2945	25	9
62639	2945	9426	9
62640	2945	6099	8
62641	2945	7557	7
62642	2945	16981	7
62643	2945	14501	7
62644	2945	14473	7
62645	2945	3895	5

```
user_info.info()

user_info.columns = user_info.columns.str.strip().str.lower().str.replace('-', '_') # clean column names
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2946 entries, 0 to 2945
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   User-ID   2946 non-null   int64 
 1   Location  2946 non-null   object
 2   Age       2946 non-null   int64 
dtypes: int64(2), object(1)
memory usage: 69.2+ KB
```

62654	2945	9417	7

```
user_info.isna().sum() #NO NULL VALUES PRESENT
```

```
user_id     0
location    0
```

```
age         0
dtype: int64
```

```
user_info.head(10)
```

	user_id	location	age
0	1	minneapolis, minnesota, usa	24
1	2	san diego, california, usa	20
2	3	novinger, missouri, usa	16
3	4	sonoma, california, usa	34
4	5	berkeley, california, usa	23
5	6	king of prussia, ,	36
6	7	berkeley, ,	22
7	8	rennes, bretagne, france	22
8	9	st. louis, missouri, usa	36
9	10	minneapolis, minnesota, usa	26

```
user_info.location.values
```

```
array(['minneapolis, minnesota, usa', 'san diego, california, usa',
       'novinger, missouri, usa', ..., 'storm lake, iowa, usa',
       'lake george, new york, usa', 'pismo beach, california, usa'],
      dtype=object)
```

```
user_location_expanded = user_info.location.str.split(',', 2, expand=True)
user_location_expanded.columns = ['city', 'state', 'country']
user_info = user_info.join(user_location_expanded)
```

```
user_info=user_info.drop('location',axis=1)
```

```
user_info.age.unique() #age must be between 5-100
```

```
array([ 24,  20,  16,  34,  23,  36,  22,  26,  30,  27,  46,  42,  25,
        29,  68,  48,  39,  33,  18,  66,  60,  32,  28,  35,  45,  31,
        62,  51,   8,  49,  21,  44, 239,  47,  37,  65,  15,  58,  38,
        41, 201,  57,  40,  53,  43,  54,  19,  56,  52,  55,  67,  13,
        59,  61,  11,  75,  12,  50,  17,  63,  14,   9, 103,  71,  77,
        83,  76, 136,   1,  72,  70,  69, 168, 148,  90,  80,  64,  73,
        81,  82, 100,  79, 116,   4, 204,   2, 101])
```

```
user_info[user_info['age']>100]=50
user_info[user_info['age']<5]=50
```

```
user_info.age.unique() #age must be between 5-100
```

```
array([ 24,  20,  16,  34,  23,  36,  22,  26,  30,  27,  46,  42,  25,
        29,  68,  48,  39,  33,  18,  66,  60,  32,  28,  35,  45,  31,
        62,  51,   8,  49,  21,  44,  50,  47,  37,  65,  15,  58,  38,
        41,  57,  40,  53,  43,  54,  19,  56,  52,  55,  67,  13,  59,
        61,  11,  75,  12,  17,  63,  14,   9,  71,  77,  83,  76,  72,
        70,  69,  90,  80,  64,  73,  81,  82, 100,  79])
```

```
user_info[user_info['user_id']==2945]
```

	user_id	age	city	state	country
2944	2945	34	lake george	new york	usa

BOOKS AND RATINGS

```
books_with_ratings = book_rat.join(book_info.set_index('book_id'), on='item')
```

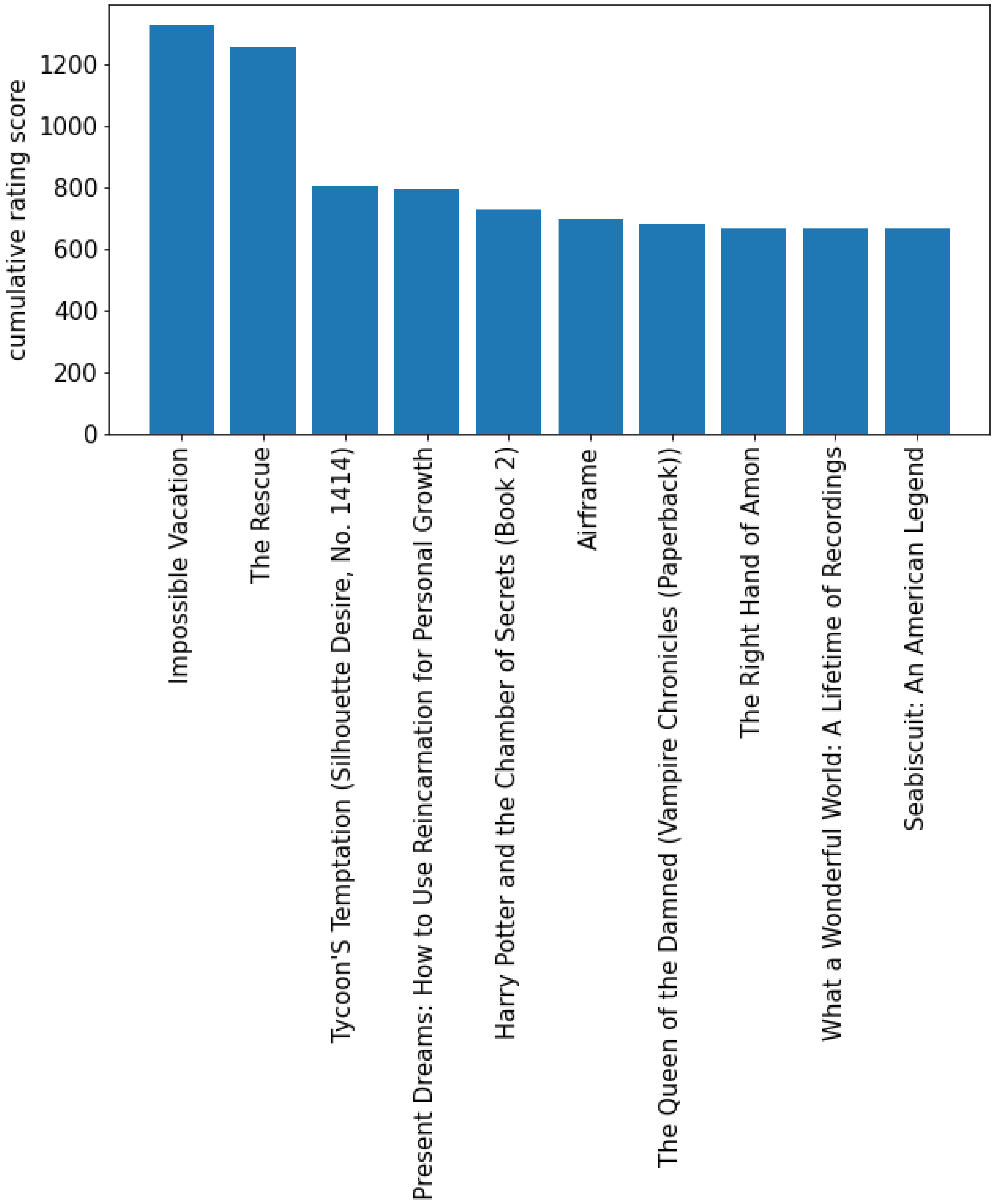
```
print(f'There are {books_with_ratings.book_title.isnull().sum()} books with no title/author information.')
print(f'This represents {len(books_with_ratings)/books_with_ratings.book_title.isnull().sum():.2f}% of the ratings dataset.')
```

```
There are 13 books with no title/author information.
This represents 4819.69% of the ratings dataset.
```

```
books_with_ratings.dropna(subset=['book_title'], inplace=True) # remove rows with missing title/author data
```

```
cm_rtg = books_with_ratings.groupby('book_title').rating.sum()
cm_rtg = cm_rtg.sort_values(ascending=False)[:10]
idx = cm_rtg.index.tolist()
vals = cm_rtg.values.tolist()

plt.figure(figsize=(10, 5))
plt.rcParams.update({'font.size': 15})
plt.bar(range(len(idx)), vals)
plt.xticks(range(len(idx)), idx, rotation='vertical')
plt.ylabel('cumulative rating score')
plt.show()
```



```
books_with_ratings.groupby('book_title').item.nunique().sort_values(ascending=False)[:581]
```

```
book_title
Pet Sematary                                   6
Wuthering Heights                              5
The Subtle Knife (His Dark Materials, Book 2)  5
Best Friends                                   4
Stardust                                       4
                                              ..
Mr. Maybe                                      2
Fair Ball: A Fan's Case for Baseball           2
Dude, Where's My Country?                      2
Zlata's Diary: A Child's Life in Sarajevo      2
```

```
High Society                                    1
Name: item, Length: 581, dtype: int64
```

```
multiple_isbns = books_with_ratings.groupby('book_title').isbn.nunique()
multiple_isbns.value_counts()
```

```
1    13433
2      509
3       59
4        9
5        2
6        1
Name: isbn, dtype: int64
```

```
book_names = books_with_ratings.book_title.unique()
book_names=book_names.tolist()
book_names[:10]
```

```
['Something Wicked This Way Comes',
 'The Mists of Avalon',
 'Sacred Sins',
 'What a Wonderful World: A Lifetime of Recordings',
 'A Coral Kiss',
 'To Marry McAllister  (Bachelor Cousins) (Harlequin Presents, No. 2273)',
 'Love Always Remembers: A Book of Poems',
 'The Subtle Knife (His Dark Materials, Book 2)',
 'Martian Chronicles',
 'Just Here Trying to Save a Few Lives : Tales of Life and Death from the ER']
```

```
for i in book_names:
    mask = books_with_ratings['book_title']==i
    c=books_with_ratings[books_with_ratings['book_title']==i].iloc[0,3]
    books_with_ratings.loc[mask,'isbn']=c
```

```
books_with_ratings.to_csv('mod_books')
```

```
books_with_ratings
```

	user	item	rating	isbn	book_title	book_author	year_of_publication	publisher
0	1	6264	7	553280325	Something Wicked This Way Comes	Ray Bradbury	1983	Bantam
1	1	4350	7	345441184	The Mists of Avalon	MARION ZIMMER BRADLEY	2000	Del Rey
2	1	6252	5	553265741	Sacred Sins	Nora Roberts	1990	Bantam Books
3	1	202	9	195086295	What a Wonderful World: A Lifetime of Recordings	Bob Thiele	1995	Oxford University Press
4	1	6266	6	446363499	A Coral Kiss	Jayne Ann Krentz	1992	Warner Books
...	...	...	...	...	...	...	...	...
62651	2945	15719	8	571169341	Arcadia	Jim Crace	1997	Ecco

```
books_with_ratings[books_with_ratings.book_title=='Jane Eyre'].head()
```

	user	item	rating	isbn	book_title	book_author	year_of_publication	publisher
1057	52	4288	8	451523326	Jane Eyre	Charlotte Bronte	1988	Signet Classics
1686	76	4288	8	451523326	Jane Eyre	Charlotte Bronte	1988	Signet Classics
16633	786	9847	7	451523326	Jane Eyre	Charlotte Bronte	0	Barnes Noble Classics
38442	1766	12162	10	451523326	Jane Eyre	Charlotte Bronte	1981	Bantam Books
47622	2249	12162	10	451523326	Jane Eyre	Charlotte Bronte	1981	Bantam Books

Double-click (or enter) to edit

```
books_users_ratings = books_with_ratings.join(user_info.set_index('user_id'), on='user')
```

```
user_item_rating = books_users_ratings[['user', 'isbn', 'rating']]
user_item_rating.head()
```

	user	isbn	rating
0	1	553280325	7
1	1	345441184	7
2	1	553265741	5
3	1	195086295	9
4	1	446363499	6

```
from sklearn import model_selection
train_data, test_data = model_selection.train_test_split(user_item_rating, test_size=0.20)
```

```
u_unique_train = train_data.user.unique()  # create a 'set' (i.e. all unique) list of vals
u_unique_train = train_data.user.unique()  # create a 'set' (i.e. all unique) list of vals
train_data_user2idx = {o:i for i, o in enumerate(u_unique_train)}

b_unique_train = train_data.isbn.unique()  # create a 'set' (i.e. all unique) list of vals
train_data_book2idx = {o:i for i, o in enumerate(b_unique_train)}


u_unique_test = test_data.user.unique()  # create a 'set' (i.e. all unique) list of vals
test_data_user2idx = {o:i for i, o in enumerate(u_unique_test)}
b_unique_test = test_data.isbn.unique()  # create a 'set' (i.e. all unique) list of vals
test_data_book2idx = {o:i for i, o in enumerate(b_unique_test)}



train_data['u_unique'] = train_data['user'].map(train_data_user2idx)
train_data['b_unique'] = train_data['isbn'].map(train_data_book2idx)

test_data['u_unique'] = test_data['user'].map(test_data_user2idx)
test_data['b_unique'] = test_data['isbn'].map(test_data_book2idx)

train_data = train_data[['u_unique', 'b_unique', 'rating']]
test_data = test_data[['u_unique', 'b_unique', 'rating']]
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:15: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  from ipykernel import kernelapp as app
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:16: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  app.launch_new_instance()
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:18: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:19: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
num_of_users = train_data['u_unique'].nunique()
num_of_books = train_data['b_unique'].nunique()



train_matrix = np.zeros((num_of_users, num_of_books))

for i in train_data.itertuples():
    train_matrix[i[1]-1, i[2]-1] = i[3]
```

```
num_of_users = test_data['u_unique'].nunique()
num_of_books = test_data['b_unique'].nunique()

test_matrix = np.zeros((num_of_users, num_of_books))

for i in test_data.itertuples():
    test_matrix[i[1]-1, i[2]-1] = i[3]
```

```
train_mat = pd.DataFrame(train_matrix)
train_mat.head(20)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	..
0	7.0	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	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	10.0	9.0	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	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	.
2	0.0	0.0	9.0	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	0.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.
3	0.0	0.0	0.0	8.0	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.
4	0.0	0.0	0.0	0.0	8.0	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	9.0	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	.
5	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	5.0	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	0.0	0.0	0.0	0.0	.
6	9.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.
8	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	8.0	0.0	0.0	0.0	6.0	9.0	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	.
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	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	5.0	.
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.0	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	0.0	0.0	0.0	0.0	0.0	.
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.0	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	0.0	0.0	0.0	0.0	.
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0	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	0.0	0.0	0.0	.
13	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	8.0	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	0.0	0.0	.
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0	0.0	0.0	9.0	0.0	0.0	7.0	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	0.0	.
15	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	6.0	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	.
16	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	8.0	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	.
17	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	9.0	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	.
18	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	8.0	0.0	0.0	8.0	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	.
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.

20 rows × 13144 columns

```
train_matrix_small = train_matrix[:10000, :10000]
test_matrix_small = test_matrix[:10000, :10000]

from sklearn.metrics.pairwise import pairwise_distances
user_similarity = pairwise_distances(train_matrix_small, metric='cosine')
item_similarity = pairwise_distances(train_matrix_small.T, metric='cosine')
```

```
user_similarity.shape
```

```
(1295, 1295)
```

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

        ratings_diff = (ratings - mean_user_rating[:, np.newaxis])
        pred = mean_user_rating[:, np.newaxis] + similarity.dot(ratings_diff) / np.array([np.abs(similarity).sum(axis=1)]).T
    elif type == 'item':
        pred = ratings.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
    return pred
```

```
item_prediction = predict(train_matrix_small, item_similarity, type='item')
user_prediction = predict(train_matrix_small, user_similarity, type='user')
```

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

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

# Call on test set to get error from each approach ('user' or 'item')
print("User-based RMSE:", rmse(user_prediction, test_matrix_small))
print("Item-based RMSE:",rmse(item_prediction, test_matrix_small))
```

```
User-based RMSE: 8.08183993287675
Item-based RMSE: 8.090812055176023
```

```
!pip install surprise
from surprise import Reader, Dataset
```

```
Collecting surprise
  Downloading https://files.pythonhosted.org/packages/61/de/e5cba8682201fcf9c3719a6fdda95693468ed061945493dea2dd37c5618b/surprise-0.1-py2.py3-none-any.whl
Collecting scikit-surprise
  Downloading https://files.pythonhosted.org/packages/97/37/5d334adaf5ddd65da99fc65f6507e0e4599d092ba048f4302fe8775619e8/scikit-surprise-1.1.1.tar.gz (11.8MB)
     |████████████████████████████████| 11.8MB 6.7MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise->surprise) (0.17.0)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise->surprise) (1.18.5)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise->surprise) (1.4.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise->surprise) (1.15.0)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp36-cp36m-linux_x86_64.whl size=1670949 sha256=ac16fe47d6c6f45bd13c26b90f1bb717a0d7d8b206d5f339833468b13a21270d
  Stored in directory: /root/.cache/pip/wheels/78/9c/3d/41b419c9d2aff5b6e2b4c0fc8d25c538202834058f9ed110d0
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.1 surprise-0.1
```

```
from surprise import Reader, Dataset



reader = Reader(rating_scale=(1, 10))

data = Dataset.load_from_df(user_item_rating, reader)


from surprise import SVD, NMF, model_selection, accuracy



model = SVD()

# Train on books dataset
get_ipython().run_line_magic('time', "model_selection.cross_validate(model, data, measures=['RMSE'], cv=5, verbose=True)")
```

```
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.4991  1.4876  1.4868  1.4951  1.4820  1.4901  0.0061
Fit time          3.22    3.17    3.22    3.11    3.20    3.18    0.04
Test time         0.10    0.10    0.16    0.07    0.10    0.10    0.03
CPU times: user 17 s, sys: 20.9 ms, total: 17 s
Wall time: 17 s
{'fit_time': (3.22021746635437,
  3.17022442817688,
  3.218785047531128,
  3.107053279876709,
  3.2004053592681885),
 'test_rmse': array([1.49911145, 1.48759224, 1.48682344, 1.4950668 , 1.48199087]),
 'test_time': (0.09833097457885742,
  0.09599781036376953,
  0.15578722953796387,
  0.07186698913574219,
  0.10065412521362305)}
```

```
trainset, testset = model_selection.train_test_split(data, test_size=0.2)

# Instantiate the SVD model.
model = SVD()

# Train the algorithm on the training set, and predictt ratings for the test set
model.fit(trainset)
predictions = model.test(testset)

# Then compute RMSE
accuracy.rmse(predictions)
```

```
RMSE: 1.5000
1.4999732859976673
```

```
uid = 69  # the user_id int
iid = '61057819' # the unique_isbn string
```

```
pred = model.predict(uid, iid, verbose=True)
```

```
user: 69         item: 61057819   r_ui = None   est = 9.36   {'was_impossible': False}
```

```
print(" Estimated rating for the book :",pred.est)
actual_rtg = user_item_rating[(user_item_rating.user==pred.uid) & (user_item_rating.isbn==pred.iid)].rating.values[0]
print("The actual rating given :" ,actual_rtg)
```

```
 Estimated rating for the book : 9.360094790923787
The actual rating given : 10
```

```
from collections import defaultdict

def get_top_n(predictions, n=100):

    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))

    print(len(top_n[uid]))
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]

    return top_n
```

```
reading_list = defaultdict(list)
```

```
print(predictions)
```

```
[Prediction(uid=2612, iid='375502947', r_ui=8.0, est=8.1726994161929, details={'was_impossible': False}), Prediction(uid=807, iid='515118907', r_ui=7.0, est=8.024957467071145, details={'wa
```

```
books_bought = pd.read_csv('/content/drive/My Drive/book_history.dat',sep='\t')
```

```
books_bought.head(100)
```

	user	item	accessed
0	1	152	1
1	1	153	1
2	1	2176	1
3	1	154	1
4	1	734	1
...	...	...	...
95	1	1315	1

```
books_bought.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 272678 entries, 0 to 272677
Data columns (total 3 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   user      272678 non-null  int64
 1   item      272678 non-null  int64
 2   accessed  272678 non-null  int64
dtypes: int64(3)
memory usage: 6.2 MB
```

```
#books_bought[books_bought['user']==1]['item'].unique().tolist()
```

```
books_users_ratings[books_users_ratings['book_title']=='Something Wicked This Way Comes']['item'].unique().tolist()
```

```
[6264, 1878]
```

```
books_b = pd.DataFrame({'user':[],'isbn':[]})
```

```
books_b
```

	user	isbn

```
books_users_ratings[books_users_ratings['isbn']=='61057819']['item'].values[0]
```

```
2148
```

```
pred = model.test(testset)
top_n = get_top_n(pred)
```

```
def get_reading_list(userid):

    reading_list = defaultdict(list)
    print(books_b)
    for i in [10,20,30,40,50,60,2000]:
      top_n = get_top_n(predictions, n=i)
      print(len(top_n[userid]))
      for n in top_n[userid]:
          book, rating = n
          title = books_users_ratings.loc[books_users_ratings.isbn==book].book_title.unique()[0]
          item=books_users_ratings[books_users_ratings['book_title']==title]['isbn'].unique().tolist()
          url = books_users_ratings.loc[books_users_ratings.isbn==book].image_url_m.unique()[0]

          user_seen = books_b[books_b['user']==userid]['isbn'].unique().tolist()
          print("\n\n :",i,item,user_seen)
          if item[0] not in user_seen:
            reading_list[title] = [rating,url]
      if(len(reading_list)>15):
        print("\nOK!")
        break;
    return reading_list
```

```
4
```

```
example_reading_list = get_reading_list(userid=69)
for book, rating in example_reading_list.items():
    print(f'{book}: {rating}')
```

```
Empty DataFrame
Columns: [user, isbn]
Index: []
4
10
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-69-cd281af55ade> in <module>()
----> 1 example_reading_list = get_reading_list(userid=69)
      2 for book, rating in example_reading_list.items():
      3     print(f'{book}: {rating}')

                                    1 frames

/usr/local/lib/python3.6/dist-packages/pandas/core/generic.py in __getattr__(self, name)
   5137             if self._info_axis._can_hold_identifiers_and_holds_name(name):
   5138                 return self[name]
-> 5139             return object.__getattribute__(self, name)
   5140
   5141     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'image_url_m'
```

SEARCH STACK OVERFLOW

```
books_b
```

```
books_b['user']==69.0
```

```
item=books_users_ratings[books_users_ratings['book_title']=='Falling for April (Zebra Historical Romance)']['isbn'].unique().tolist()
```

```
user_seen = books_b[books_b['user']==69.0]['isbn'].unique().tolist()
```

```
item[0] not in user_seen
```

```
item[0]
```

```
#books_b = pd.DataFrame({'user':[],'isbn':[]})
books_b=add_entry(69,item[0],books_b)
```

```
def add_entry(uid,isbn,b):
  b = b.append({'user': int(uid),'isbn':isbn}, ignore_index=True)
  return b
```

```
model.pu[0].shape
```

```
np.matmul(model.qi[100].T,model.pu[1])
```

```
import pickle
```

```
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

```
loaded_model = pickle.load(open(filename, 'rb'))
```

```
loaded_model
```

```
predictions = loaded_model.test(testset)
```

Choosing the parameters

```
trainset, testset = model_selection.train_test_split(data, test_size=0.2)
```

```
from surprise.model_selection import GridSearchCV

param_grid = {'n_factors': [60,80, 100, 120], 'lr_all': [0.001, 0.005, 0.01,0.00001], 'reg_all': [0.01, 0.02, 0.04,0.002]}
```

```
#TwO MEtrics used - Root mean sq and mean abs error
gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=3)
```

```
%time gs.fit(data)
```

```
    CPU times: user 8min 31s, sys: 257 ms, total: 8min 31s
    Wall time: 8min 32s
```

```
Model = gs.best_estimator['rmse']

print(gs.best_score['rmse'])
print(gs.best_params['rmse'])
```

```
    1.488975042203028
    {'n_factors': 60, 'lr_all': 0.005, 'reg_all': 0.04}
```

```
testset
```

```
model = SVD(n_factors=60, lr_all=0.005, reg_all=0.04)
model.fit(trainset)
test_pred = model.test(testset)
print("SVD : Test Set")
accuracy.rmse(test_pred, verbose=True)
```

```
    SVD : Test Set
    RMSE: 1.5110
    1.510984421729155
```

```
filename = 'finalized_model1.sav'
pickle.dump(model, open(filename, 'wb'))
```

```
loaded_model = pickle.load(open(filename, 'rb'))
```

```
cnt=0
for i in testset:
  if i[0]==2329:
    cnt+=1
```

```
cnt
```

```
pd.DataFrame(testset).to_csv("test")
```

```
 d = pd.read_csv("test")
 d_list = d.values.tolist()
 for i in range(len(d_list)):
```

```
 for i in range(len(d_list)):
   d_list[i]=tuple(d_list[i])
```

```
len(testset)
```

```
len(d_list)
```

```
testset[0]
```

```
d_list[0]
```

```
d = d.iloc[:,1:]
d_list = d.values.tolist()
for i in range(len(d_list)):
   d_list[i]=tuple(d_list[i])
```

```
books_users_ratings.to_csv("books_users_ratings")
```

```
books_users_ratings
```

```
ddd = pd.read_csv("books_users_ratings")
```

```
ddd
```