```
!pip install plotly_express
!pip install --upgrade plotly
!pip install scikit-surprise

import pandas as pd
import plotly_express as px
import numpy as np
import re
import time
import string
import seaborn as sns
import plotly.graph_objects as go
import nltk
import requests
from IPython import display
from matplotlib import cm, gridspec
from matplotlib import pyplot as plt
from plotly.subplots import make_subplots
from wordcloud import WordCloud, STOPWORDS
from nltk.draw.dispersion import dispersion_plot
from PIL import Image
import nltk
from nltk.corpus import stopwords
nltk.download('punkt')


# Restrict minor warnings
import warnings
warnings.filterwarnings('ignore')

from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from sklearn.compose import ColumnTransformer
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.model_selection import train_test_split

##Deep Learning specific libraries
import tensorflow as tf
from tensorflow import keras
import tensorflow.keras.backend as K
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense , Concatenate
from tensorflow.keras.optimizers import Adam,SGD,Adagrad,Adadelta,RMSprop
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import model_to_dot
from tensorflow.keras.callbacks import ReduceLROnPlateau
```

```
from tensorflow.keras.layers import Dropout, Flatten,Activation,Input,Embedding
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization
from tensorflow.keras.layers import dot
from tensorflow.keras.models import Model
from tensorflow.python.data import Dataset

print(tf.__version__)
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: plotly_express in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pandas>=0.20.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: plotly>=4.1.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: scipy>=0.18 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from patsy
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packages
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packages
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/dist-packages (
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
2.8.2
```

```
books = pd.read_csv(r"/content/sample_data/goodreads_books.csv", error_bad_lines=False, en
book_tags = pd.read_csv(r"/content/sample_data/book_tags.csv", error_bad_lines=False, enco
tags = pd.read_csv(r"/content/sample_data/tags.csv", error_bad_lines=False, encoding='ISO-

print("Books Data:    ", books.shape)
print("Book tags Data:    ",book_tags.shape)
print("tags: ", tags.shape)
```

```
Books Data:     (52201, 28)
Book tags Data:     (9972, 3)
tags:  (83, 2)
```

```
books.head()
```

	goodreads_book_id	title	link	series
0	630104	Inner Circle	https://www.goodreads.com//book/show/630104.In...	(Private #5
1	9487	A Time to Embrace	https://www.goodreads.com//book/show/9487.A_Ti...	(Timeless Love #2
2	6050894	Take Two	https://www.goodreads.com//book/show/6050894-t...	(Above the Line #2
3	39030	Reliquary	https://www.goodreads.com//book/show/39030.Rel...	(Pendergas #2
4	998	The Millionaire Next Door: The Surprising	https://www.goodreads.com//book/show/998.The_M...	NaN

```
1 book_tags.head()
```

	goodreads_book_id	tag_id	count
0	1	30574	167697
1	1	11305	37174
2	1	11557	34173
3	1	8717	12986
4	1	33114	12716

```
1 tags.head()
```

```
1 books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52201 entries, 0 to 52200
Data columns (total 28 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   goodreads_book_id       52201 non-null  object 
 1   title                   52199 non-null  object 
 2   link                    52199 non-null  object 
 3   series                  23319 non-null  object 
 4   cover_link              51592 non-null  object 
 5   author                  52199 non-null  object 
 6   author_link             52199 non-null  object 
 7   rating_count            52199 non-null  float64
 8   review_count            52199 non-null  float64
 9   average_rating          52199 non-null  float64
 10  five_star_ratings       52199 non-null  float64
 11  four_star_ratings       52199 non-null  float64
 12  three_star_ratings      52199 non-null  float64
 13  two_star_ratings        52199 non-null  float64
 14  one_star_ratings        52199 non-null  float64
 15  number_of_pages         49869 non-null  float64
 16  date_published          51339 non-null  object 
 17  publisher               48563 non-null  object 
 18  original_title          39250 non-null  object 
 19  isbn                    40316 non-null  object 
 20  isbn13                  39507 non-null  object 
 21  asin                    5236 non-null   object 
 22  settings                11515 non-null  object 
 23  characters              13695 non-null  object 
 24  awards                  10626 non-null  object 
 25  amazon_redirect_link    52199 non-null  object 
 26  worldcat_redirect_link  48214 non-null  object 
 27  description             49624 non-null  object 
dtypes: float64(9), object(19)
memory usage: 11.2+ MB
```

```
1 book_tags.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9972 entries, 0 to 9971
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   goodreads_book_id  9972 non-null   int64
 1   tag_id             9972 non-null   int64
 2   count              9972 non-null   int64
dtypes: int64(3)
memory usage: 233.8 KB
```

```
1 tags.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 83 entries, 0 to 82
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   tag_id    83 non-null     int64 
 1   tag_name  83 non-null     object
dtypes: int64(1), object(1)
memory usage: 1.4+ KB
```

```
1 print(f'Duplicate entries: {tags.duplicated().sum()}')
```

```
Duplicate entries: 0
```

```
1 print(f'Duplicate entries: {book_tags.duplicated().sum()}')
```

```
Duplicate entries: 0
```

```
1 print(f'Duplicate entries: {books.duplicated().sum()}')
```

```
Duplicate entries: 0
```

```
1 books.isna().sum()
```

```
goodreads_book_id            0
title                        2
link                         2
series                   28882
cover_link                 609
author                       2
author_link                  2
rating_count                 2
review_count                 2
average_rating               2
five_star_ratings            2
four_star_ratings            2
three_star_ratings           2
two_star_ratings             2
one_star_ratings             2
number_of_pages           2332
date_published             862
publisher                 3638
original_title           12951
isbn                     11885
isbn13                   12694
asin                     46965
settings                 40686
characters               38506
awards                   41575
amazon_redirect_link         2
worldcat_redirect_link    3987
```

```
    description               2577
    dtype: int64
```

```
1 books = books.fillna("", inplace=False)
2 books.isna().sum()
```

```
    goodreads_book_id         0
    title                     0
    link                      0
    series                    0
    cover_link                0
    author                    0
    author_link               0
    rating_count              0
    review_count              0
    average_rating            0
    five_star_ratings         0
    four_star_ratings         0
    three_star_ratings        0
    two_star_ratings          0
    one_star_ratings          0
    number_of_pages           0
    date_published            0
    publisher                 0
    original_title            0
    isbn                      0
    isbn13                    0
    asin                      0
    settings                  0
    characters                0
    awards                    0
    amazon_redirect_link      0
    worldcat_redirect_link    0
    description               0
    dtype: int64
```

```
1 book_tags.isna().sum()
2
```

```
    goodreads_book_id    0
    tag_id               0
    count                0
    dtype: int64
```

```
1 tags.isna().sum()
```

```
    tag_id      0
    tag_name    0
    dtype: int64
```

```
1 tags['tag_name'].unique()
```

```
array(['adventure', 'angels', 'biography', 'book-club', 'books-i-own',
       'business', 'children', 'childrens', 'classic', 'classics',
       'comics', 'contemporary', 'crime', 'currently-reading',
       'dan-brown', 'dragons', 'drama', 'dystopia', 'dystopian',
       'fantasy', 'favorites', 'favourites', 'feminism', 'fiction',
       'food', 'graphic-novel', 'graphic-novels', 'harry-potter',
       'historical', 'historical-fiction', 'historical-romance',
       'history', 'horror', 'humor', 'hunger-games', 'literature',
       'magic', 'manga', 'memoir', 'mystery', 'mythology', 'new-adult',
       'non-fiction', 'nonfiction', 'owned', 'owned-books', 'paranormal',
       'paranormal-romance', 'philosophy', 'picture-books', 'plays',
       'poetry', 'psychology', 're-read', 'read-in-2016',
       'realistic-fiction', 'religion', 'romance', 'school', 'sci-fi',
       'science', 'science-fiction', 'scifi', 'self-help', 'series',
       'shakespeare', 'short-stories', 'star-wars', 'steampunk',
       'stephen-king', 'thriller', 'time-travel', 'to-read', 'travel',
       'true-crime', 'twilight', 'urban-fantasy', 'vampire', 'vampires',
       'writing', 'ya', 'young-adult', 'zombies'], dtype=object)
```

```
1 tags['tag_name']=tags['tag_name'].replace(['childrens'],'children')
2 tags['tag_name']=tags['tag_name'].replace(['dystopian'],'dystopia')
3 tags['tag_name']=tags['tag_name'].replace(['favourites'],'favorites')
4 tags['tag_name']=tags['tag_name'].replace(['historical-fiction'],'historical')
5 tags['tag_name']=tags['tag_name'].replace(['graphic-novels'],'graphic-novel')
6 tags['tag_name']=tags['tag_name'].replace(['nonfiction'],'non-fiction')
7 tags['tag_name']=tags['tag_name'].replace(['owned-books'],'owned')
8 tags['tag_name']=tags['tag_name'].replace(['classics'],'classic')
9 tags['tag_name']=tags['tag_name'].replace(['science-fiction','scifi'],'sci-fi')
10 tags['tag_name']=tags['tag_name'].replace(['vampires'],'vampire')
```

```
1 tags['tag_name'].unique()
```

```
array(['adventure', 'angels', 'biography', 'book-club', 'books-i-own',
       'business', 'children', 'classic', 'comics', 'contemporary',
       'crime', 'currently-reading', 'dan-brown', 'dragons', 'drama',
       'dystopia', 'fantasy', 'favorites', 'feminism', 'fiction', 'food',
       'graphic-novel', 'harry-potter', 'historical',
       'historical-romance', 'history', 'horror', 'humor', 'hunger-games',
       'literature', 'magic', 'manga', 'memoir', 'mystery', 'mythology',
       'new-adult', 'non-fiction', 'owned', 'paranormal',
       'paranormal-romance', 'philosophy', 'picture-books', 'plays',
       'poetry', 'psychology', 're-read', 'read-in-2016',
       'realistic-fiction', 'religion', 'romance', 'school', 'sci-fi',
       'science', 'self-help', 'series', 'shakespeare', 'short-stories',
       'star-wars', 'steampunk', 'stephen-king', 'thriller',
       'time-travel', 'to-read', 'travel', 'true-crime', 'twilight',
       'urban-fantasy', 'vampire', 'writing', 'ya', 'young-adult',
       'zombies'], dtype=object)
```

```
1 #Remove Unnecessary columns from books data
2 books.info()
```

```
3 books = books.drop(['link','publisher','cover_link','number_of_pages','date_published','or
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52201 entries, 0 to 52200
Data columns (total 28 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   goodreads_book_id       52201 non-null  object
 1   title                   52201 non-null  object
 2   link                    52201 non-null  object
 3   series                  52201 non-null  object
 4   cover_link              52201 non-null  object
 5   author                  52201 non-null  object
 6   author_link             52201 non-null  object
 7   rating_count            52201 non-null  object
 8   review_count            52201 non-null  object
 9   average_rating          52201 non-null  object
 10  five_star_ratings       52201 non-null  object
 11  four_star_ratings       52201 non-null  object
 12  three_star_ratings      52201 non-null  object
 13  two_star_ratings        52201 non-null  object
 14  one_star_ratings        52201 non-null  object
 15  number_of_pages         52201 non-null  object
 16  date_published          52201 non-null  object
 17  publisher               52201 non-null  object
 18  original_title          52201 non-null  object
 19  isbn                    52201 non-null  object
 20  isbn13                  52201 non-null  object
 21  asin                    52201 non-null  object
 22  settings                52201 non-null  object
 23  characters              52201 non-null  object
 24  awards                  52201 non-null  object
 25  amazon_redirect_link    52201 non-null  object
 26  worldcat_redirect_link  52201 non-null  object
 27  description             52201 non-null  object
dtypes: object(28)
memory usage: 11.2+ MB
```

```
1 books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52201 entries, 0 to 52200
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   goodreads_book_id   52201 non-null  object
 1   title               52201 non-null  object
 2   author              52201 non-null  object
 3   author_link         52201 non-null  object
 4   rating_count        52201 non-null  object
 5   review_count        52201 non-null  object
 6   average_rating      52201 non-null  object
 7   five_star_ratings   52201 non-null  object
 8   four_star_ratings   52201 non-null  object
 9   three_star_ratings  52201 non-null  object
 10  two_star_ratings    52201 non-null  object
```

```
 11  one_star_ratings    52201 non-null  object
 12  isbn                52201 non-null  object
dtypes: object(13)
memory usage: 5.2+ MB
```

```
1 books.head()
```

	goodreads_book_id	title	author	author_link
0	630104	Inner Circle	Kate Brian, Julian Peploe	https://www.goodreads.com/author/show/94091.Ka...
1	9487	A Time to Embrace	Karen Kingsbury	https://www.goodreads.com/author/show/3159984....
2	6050894	Take Two	Karen Kingsbury	https://www.goodreads.com/author/show/3159984....
3	39030	Reliquary	Douglas Preston, Lincoln Child	https://www.goodreads.com/author/show/12577.Do...
4	998	The Millionaire Next Door: The Surprising Secr...	Thomas J. Stanley, William D. Danko	https://www.goodreads.com/author/show/659.Thom...

```
1 #Book ratings count using group by
2 ratings = books.groupby(['title'])['five_star_ratings'].count().sort_values(ascending=Fals
3
```

```
1 ratings
```

	title	five_star_ratings
0	Broken	14
1	Legacy	14
2	Selected Poems	13
3	Twisted	11
4	Hunted	10
...	...	...

```
1 avgrate=books.groupby(['title'])['average_rating'].count().sort_values(ascending=False).re
```

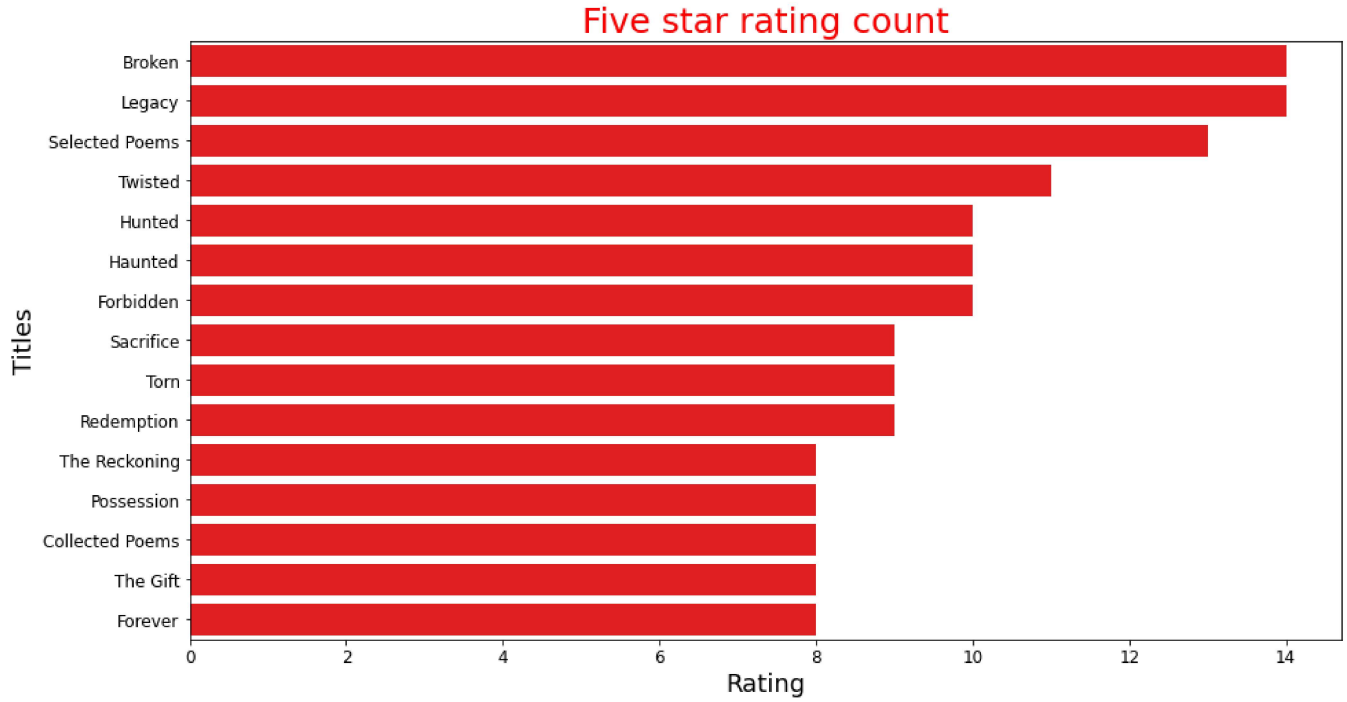
49682	In Pursuit of the Proper Sinner	1

```
1 plt.figure(figsize=(15,8))
2 bardis=sns.barplot(avgrate['average_rating'][:15],avgrate['title'][:15],color='black')
3 bardis.set_title('Average rating count', fontsize=25,fontweight='200',color='black')
4 bardis.set_xlabel('Rating',fontsize=18)
5 bardis.set_ylabel('Titles',fontsize=18)
6 plt.yticks(fontsize=12)
7 plt.xticks(fontsize=12)
```

```
(array([ 0.,  2.,  4.,  6.,  8., 10., 12., 14., 16.]),
```

```
plt.figure(figsize=(15,8))
bardis=sns.barplot(ratings['five_star_ratings'][:15],ratings['title'][:15],color='red')
bardis.set_title('Five star rating count', fontsize=25,fontweight='200',color='red')
bardis.set_xlabel('Rating',fontsize=18)
bardis.set_ylabel('Titles',fontsize=18)
plt.yticks(fontsize=12)
plt.xticks(fontsize=12)
```

```
(array([ 0.,  2.,  4.,  6.,  8., 10., 12., 14., 16.]),
 <a list of 9 Text major ticklabel objects>)
```



Double-click (or enter) to edit

```
#Book tags data
book_tags.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9972 entries, 0 to 9971
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
```

```
 0   goodreads_book_id  9972 non-null   int64
 1   tag_id             9972 non-null   int64
 2   count              9972 non-null   int64
dtypes: int64(3)
memory usage: 233.8 KB
```

Double-click (or enter) to edit

```
#merge data into one
print(f'Number of Duplicate entries: {book_tags.duplicated().sum()}')

```

```
Number of Duplicate entries: 0
```

```
datamerge = pd.merge(books, book_tags,on='goodreads_book_id', how='inner')
datamerge = pd.merge(datamerge, tags,  how='inner')
datamerge.info()
datamerge.shape
datamerge
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7154 entries, 0 to 7153
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   goodreads_book_id   7154 non-null   object
 1   title               7154 non-null   object
 2   author              7154 non-null   object
 3   author_link         7154 non-null   object
 4   rating_count        7154 non-null   object
 5   review_count        7154 non-null   object
 6   average_rating      7154 non-null   object
 7   five_star_ratings   7154 non-null   object
 8   four_star_ratings   7154 non-null   object
 9   three_star_ratings  7154 non-null   object
 10  two_star_ratings    7154 non-null   object
 11  one_star_ratings    7154 non-null   object
 12  isbn                7154 non-null   object
 13  tag_id              7154 non-null   int64 
 14  count               7154 non-null   int64 
 15  tag_name            7154 non-null   object
dtypes: int64(2), object(14)
memory usage: 950.1+ KB
```

	goodreads_book_id	title	author	author_
0	23553419	The Ice Twins	S.K. Tremayne	https://www.goodreads.com/author/show/8431C
1	4714126	The Walking Dead, Vol. 9: Here We Remain	Robert Kirkman, Charlie Adlard, Cliff Rathburn	https://www.goodreads.com/author/show/12425
2	15764	The Amateur Marriage	Anne Tyler	https://www.goodreads.com/author/show/457.A
3	12275680	Wicked Business	Janet Evanovich	https://www.goodreads.com/author/show/2384.

```
1 print(f'Number of Duplicate entries: {datamerge.duplicated().sum()}')
```

```
Number of Duplicate entries: 0
```

Military

```
1 datamerge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7154 entries, 0 to 7153
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   goodreads_book_id   7154 non-null   object
```

```
 1   title               7154 non-null   object
 2   author              7154 non-null   object
 3   author_link         7154 non-null   object
 4   rating_count        7154 non-null   object
 5   review_count        7154 non-null   object
 6   average_rating      7154 non-null   object
 7   five_star_ratings   7154 non-null   object
 8   four_star_ratings   7154 non-null   object
 9   three_star_ratings  7154 non-null   object
 10  two_star_ratings    7154 non-null   object
 11  one_star_ratings    7154 non-null   object
 12  isbn                7154 non-null   object
 13  tag_id              7154 non-null   int64
 14  count               7154 non-null   int64
 15  tag_name            7154 non-null   object
dtypes: int64(2), object(14)
memory usage: 950.1+ KB
```

```
dataset1 = datamerge[datamerge['average_rating'] != 0]
dataset1 = dataset1.reset_index(drop = True)
dataset1.shape
```

```
(7154, 16)
```

```
def popularity_based(dataframe, n):
    if n >= 1 and n <= len(dataframe):
        data = pd.DataFrame(dataframe.groupby('isbn')['average_rating'].count()).sort_valu
        print(data)
        result = pd.merge(data, books, on='isbn')
        return result
    return "Invalid number of books entered!!"
```

```
print("Top 5 Popular books are: ")
popularity_based(dataset1,5)
```

```
Top 5 Popular books are:
            average_rating
isbn
                       814
439023483               19
439064864               17
439023513               17
043965548X              15
```

	isbn	average_rating_x	goodreads_book_id	title	author	
0		814	6050894	Take Two	Karen Kingsbury	https
1		814	16116760	Silence	Natasha Preston	https
2		814	18523711	Darkness	Laurann Dohner	https
3		814	17234659	Ten Thousand Skies Above You	Claudia Gray	https
4		814	25906415	Liam: Midsummer's Magic Bonus Book	Emmie Lou Kates	https
...	...	...	...	...	...	
11884		814	35965782	The Lost Continent	Tui T. Sutherland	https

```
def avgRating(newdf, df):
    newdf['average_rating'] = 0
    for x in range(len(newdf)):
        l = list(df.loc[df['title'] == newdf['title'][x]]['five_star_ratings'])
        newdf['average_rating'][x] = sum(l)/len(l)
    return newdf

df = pd.DataFrame(datamerge['title'].value_counts())
df['five_star_ratings'] = df['title']
df['title'] = df.index
df.reset_index(level=0, inplace=True)
df = df.drop('index',axis=1)
df = avgRating(df, datamerge)
df.to_pickle('weightedData')
#df = pd.read_pickle('rating_count')
```

```
## C - Mean vote across the whole
C = df['average_rating'].mean()

```

```
## Minimum number of votes required to be in the chart
m = df['five_star_ratings'].quantile(0.90)
```

```
def five_star_ratings(x, m=m, C=C):
    v = x['five_star_ratings']    #v - number of votes
    R = x['average_rating']   #R - Average Rating
    return (v/(v+m) * R) + (m/(m+v) * C)
```

```
df = df.loc[df['five_star_ratings'] >= m]
df['score'] = df.apply(five_star_ratings, axis=1)
df = df.sort_values('score', ascending=False)
print("Recommended Books:-\n")
df.head(2)
```

```
Recommended Books:-
```

	title	five_star_ratings	average_rating	score
1	Harry Potter and the Sorcerer's Stone	19	4397134	3.804310e+06
0	The Hunger Games	19	3349551	2.899579e+06

```
from surprise import NMF, SVD, SVDpp, CoClustering
from surprise.model_selection import cross_validate
from surprise import Reader, Dataset
from surprise import KNNBasic, KNNWithMeans, KNNWithZScore, KNNBaseline

```

```
def Basic(min_rating):
  rating_counts = pd.DataFrame(datamerge['title'].value_counts())
  low_reviewed_books= rating_counts[rating_counts['title'] <= min_rating].index
  common_books = datamerge[~datamerge['title'].isin(low_reviewed_books)]
  return (low_reviewed_books, common_books)
```

```
def pivot_table(common_books):
  user_book_df = common_books.pivot_table(index=['goodreads_book_id'],
                                          columns=['title'],
                                          values='average_rating',fill_value=0)
  return user_book_df
```

```
min_rating = 1000
low_reviewed_books, common_books =Basic(min_rating)
new_Books= datamerge[['goodreads_book_id','average_rating']]
new_Books.head()
```

	goodreads_book_id	average_rating
0	23553419	3.65
1	4714126	4.24
2	15764	3.62
3	12275680	3.79

```
1 book_considered=datamerge[['goodreads_book_id','title','average_rating']]
2 book_considered.head()
```

	goodreads_book_id	title	average_rating
0	23553419	The Ice Twins	3.65
1	4714126	The Walking Dead, Vol. 9: Here We Remain	4.24
2	15764	The Amateur Marriage	3.62
3	12275680	Wicked Business	3.79
4	12143200	Drift: The Unmooring of American Military Power	4.08

```
1 reader = Reader(rating_scale=(1, 5))
2 data = Dataset.load_from_df(book_considered, reader)
```

```
1 unique_ids = book_considered['goodreads_book_id'].unique()
2 # get the list of the ids that the userid 226 has rated
3 iids226 = book_considered.loc[book_considered['goodreads_book_id']==226, 'goodreads_book_i
4 # remove the rated movies for the recommendations
5 movies_to_predict = np.setdiff1d(unique_ids,iids226)
```

```
1 def book_names(pd):
2   books_name=[]
3   for id in pd["goodreads_book_id"]:
4     books_name.append(list(books.loc[books['goodreads_book_id']==id, 'title'])[0])
5
6   return(books_name)
```

```
1 current_user=226
```

```
1 algo = NMF()
2 start1=time.time()
3 algo.fit(data.build_full_trainset())
4 my_recs1 = []
5 for iid in movies_to_predict:
6     my_recs1.append((iid, algo.predict(uid=current_user,iid=iid).est))
7 end1=time.time()
```

```

pd1=pd.DataFrame(my_recs1, columns=['goodreads_book_id', 'predictions']).sort_values('pred
pd1["book_title"]=book_names(pd1)
pd1
```

	goodreads_book_id	predictions	book_title
0	1	4.055995	Harry Potter and the Half-Blood Prince
3025	3293821	4.055995	The Last Straw
3031	3367956	4.055995	Hotel on the Corner of Bitter and Sweet
3030	3347892	4.055995	Envy
3029	3344411	4.055995	Every Man Dies Alone

```
algo = SVDpp()
start3=time.time()
algo.fit(data.build_full_trainset())
my_recs3 = []
for iid in movies_to_predict:
    my_recs3.append((iid, algo.predict(uid=current_user,iid=iid).est))
end3=time.time()
pd3=pd.DataFrame(my_recs3, columns=['goodreads_book_id', 'predictions']).sort_values('pred
pd3['book_title']=book_names(pd3)
pd3
```

	goodreads_book_id	predictions	book_title
0	1	4.055995	Harry Potter and the Half-Blood Prince
3025	3293821	4.055995	The Last Straw
3031	3367956	4.055995	Hotel on the Corner of Bitter and Sweet
3030	3347892	4.055995	Envy
3029	3344411	4.055995	Every Man Dies Alone

```
algo = CoClustering()
start4=time.time()
algo.fit(data.build_full_trainset())
my_recs5 = []
for iid in movies_to_predict:
    my_recs5.append((iid, algo.predict(uid=current_user,iid=iid).est))
time4=time.time()
pd5=pd.DataFrame(my_recs5, columns=['goodreads_book_id', 'predictions']).sort_values('pred
pd5["book_title"]=book_names(pd5)
pd5
```

	goodreads_book_id	predictions	book_title
0	1	4.055995	Harry Potter and the Half-Blood Prince
3025	3293821	4.055995	The Last Straw
3031	3367956	4.055995	Hotel on the Corner of Bitter and Sweet
3030	3347892	4.055995	Envy

Evaluation

```
#split data
# creating a surprise object

reader = Reader(rating_scale=(0, 10))
data   = Dataset.load_from_df(datamerge[['goodreads_book_id','title','five_star_ratings']]


# Split the data into training & testing sets. Python's surprise documentation has the ste
# https://surprise.readthedocs.io/en/stable/FAQ.html

raw_ratings = data.raw_ratings
import random
random.shuffle(raw_ratings)                 # shuffle dataset

threshold   = int(len(raw_ratings)*0.8)

train_raw_ratings = raw_ratings[:threshold] # 80% of data is trainset
test_raw_ratings  = raw_ratings[threshold:] # 20% of data is testset

data.raw_ratings = train_raw_ratings        # data is now the trainset
trainset         = data.build_full_trainset()
testset          = data.construct_testset(test_raw_ratings)
```

```
books[books['title']=='Stardust']
```

	goodreads_book_id	title	author	author_link
41860	16793	Stardust	Neil Gaiman	https://www.goodreads.com/author/show/1221698....

```
# Entire dataset will be used for building recommendations

reader = Reader(rating_scale=(0, 10))
data = Dataset.load_from_df(datamerge[['goodreads_book_id','title','five_star_ratings']],
trainset = data.build_full_trainset()

# A list of useful trainset methods are explained here:
# https://surprise.readthedocs.io/en/stable/trainset.html
```

```
def generate_recommendationsNMF(userID=16793, like_recommend=5, get_recommend =10):

    ''' This function generates "get_recommend" number of book recommendations using
        KNNWithMeans & item based filtering. The function needs as input three
        different parameters:
        (1) userID i.e., userID for which recommendations need to be generated
        (2) like_recommend i.e., number of top recommendations for the userID to be
        considered for making recommendations
        (3) get_recommend i.e., number of recommendations to generate for the userID
        Default values are: userID=13552, like_recommend=5, get_recommend=10
    '''

    # Compute item based similarity matrix
    sim_options       = {'name':'cosine','min_support':3,'user_based':False}
    similarity_matrix = KNNWithMeans(sim_options=sim_options).fit(trainset).\
                          compute_similarities()

    userID      = trainset.to_inner_uid(userID)    # converts the raw userID to innerID
    userRatings = trainset.ur[userID]              # method .ur takes user innerID &
                                                   # returns back user ratings


    # userRatings is a list of tuples [(,),(,),(,)..]. Each tuple contains item & rating
    # given by the user for that item. Next, the tuples will be sorted within the list
    # in decreasing order of rating. Then top 'like_recommend' items & ratings are extract

    temp_df = pd.DataFrame(userRatings).sort_values(by=1, ascending=False).\
              head(like_recommend)
    userRatings = temp_df.to_records(index=False)

    # for each (item,rating) in top like_recommend user items, multiply the user rating fo
    # the item with the similarity score (later is obtained from item similarity_matrix) f
    # all items. This helps calculate the weighted rating for all items. The weighted rati
    # are added & divided by sum of weights to estimate rating the user would give an item

    recommendations   = {}

    for user_top_item, user_top_item_rating  in userRatings:

        all_item_indices          =   list(pd.DataFrame(similarity_matrix)[user_top_item].
        all_item_weighted_rating  =   list(pd.DataFrame(similarity_matrix)[user_top_item].
                                          user_top_item_rating)

        all_item_weights          =   list(pd.DataFrame(similarity_matrix)[user_top_item].


        # All items & final estimated ratings are added to a dictionary called recommendat

        for index in range(len(all_item_indices)):
            if index in recommendations:
```

```
            # sum of weighted ratings
            recommendations[index] += all_item_weighted_rating[index]
        else:
            recommendations[index]  = all_item_weighted_rating[index]


for index in range(len(all_item_indices)):
        if all_item_weights[index]  !=0:
            # final ratings (sum of weighted ratings/sum of weights)
            recommendations[index]   =recommendations[index]/\
                                       (all_item_weights[index]*like_recommend)


# convert dictionary recommendations to a be a list of tuples [(,),(,),(,)]
# with each tuple being an item & estimated rating user would give that item
# sort the tuples within the list to be in decreasing order of estimated ratings

temp_df = pd.Series(recommendations).reset_index().sort_values(by=0, ascending=False)
recommendations = list(temp_df.to_records(index=False))

# return get_recommend number of recommedations (only return items the user
# has not previously rated)

final_recommendations = []
count = 0

for item, score in recommendations:
    flag = True
    for userItem, userRating in trainset.ur[userID]:
        if item == userItem:
            flag = False       # If item in recommendations has not been rated by user
            break              # add to final_recommendations
    if flag == True:
        final_recommendations.append(trainset.to_raw_iid(item))
        count +=1              # trainset has the items stored as inner id,
                               # convert to raw id & append

    if count > get_recommend:  # Only get 'get_recommend' number of recommendations
        break
return(final_recommendations)
```

```
recommendationsKNN = generate_recommendationsNMF(userID=16793, like_recommend=5, get_recom
recommendationsKNN
```

```
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
['The Ice Twins',
 'The Complete Tales and Poems of Winnie-the-Pooh',
 'Happy Ever After',
```

```
    "Archangel's Consort",
    'The Stranger Beside Me: Ted Bundy: The Shocking Inside Story',
    'The 7 Habits Of Highly Effective Teens',
    'One Second After',
    'The Story of Art',
    'Second Chance Summer',
    'Antigone',
    'Tick Tock']
```

```
# SVD

def generate_recommendationsCOCLUSTERING(userID=16793, get_recommend =10):

    ''' This function generates "get_recommend" number of book recommendations
        using Singular value decomposition. The function needs as input two
        different parameters:
        (1) userID i.e., userID for which recommendations need to be generated
        (2) get_recommend i.e., number of recommendations to generate for the userID
        Default values are: userID=13552, get_recommend=10
    '''

    model = SVD(n_factors=50, n_epochs=10, lr_all=0.005, reg_all= 0.2)
    model.fit(trainset)

    # predict rating for all pairs of users & items that are not in the trainset

    testset = trainset.build_anti_testset()
    predictions = model.test(testset)
    predictions_df = pd.DataFrame(predictions)

    # get the top get_recommend predictions for userID

    predictions_userID = predictions_df[predictions_df['uid'] == userID].\
                         sort_values(by="est", ascending = False).head(get_recommend)

    recommendations = []
    recommendations.append(list(predictions_userID['iid']))
    recommendations = recommendations[0]

    return(recommendations)
```

```
recommendationsSVD = generate_recommendationsCOCLUSTERING(userID=16793, get_recommend =10)
recommendationsSVD
```

```
# Using another UserID to build recommendations

def generate_recommendationsSVD++(userID=99298, get_recommend =5):

    ''' This function generates "get_recommend" number of book recommendations
        using Singular value decomposition. The function needs as input two
```

```
        different parameters:
        (1) userID i.e., userID for which recommendations need to be generated
        (2) get_recommend i.e., number of recommendations to generate for the userID
        Default values are: userID=13552, get_recommend=10
    '''

    model = SVD(n_factors=50, n_epochs=10, lr_all=0.005, reg_all= 0.2)
    model.fit(trainset)

    # predict rating for all pairs of users & items that are not in the trainset

    testset = trainset.build_anti_testset()
    predictions = model.test(testset)
    predictions_df = pd.DataFrame(predictions)

    # get the top get_recommend predictions for userID

    predictions_userID = predictions_df[predictions_df['uid'] == userID].\
                         sort_values(by="est", ascending = False).head(get_recommend)

    recommendations = []
    recommendations.append(list(predictions_userID['iid']))
    recommendations = recommendations[0]

    return(recommendations)
```

```
recommendationsSVD = generate_recommendationsSVD++(userID=99298, get_recommend =5)
recommendationsSVD
```

```
#Defining a function that will recommend top 5 books
def recommend(user_id):

  book_id = list(datamerge.goodreads_book_id.unique()) #grabbing all the unique books

  book_arr = np.array(book_id) #geting all book IDs and storing them in the form of an arr
  user_arr = np.array([user_id for i in range(len(book_id))])
  prediction = model.predict([book_arr, user_arr])

  prediction = prediction.reshape(-1) #reshape to single dimension
  prediction_ids = np.argsort(-prediction)[0:5]

  recommended_books = pd.DataFrame(datamerge.iloc[prediction_ids], columns = ['goodreads_b
  print('Top 5 recommended books for you: \n')
  return recommended_books
```

```
recommend(16793)
```