**Data and Web Mining CLA-2**

**Done by:**

**Y. Sai Abhishek – AP18110010124**

**P. Murali Gautham – AP18110010111**

**V. Sathwika – AP18110010105**

**V. Sri Nilay – AP18110010679**

**V. Sai Surya - AP18110010090**

**Abstract:**

Readers who are mostly interested in book reading usually prefer libraries and some fellow reader recommendations which in some cases may not be the best recommendation according to their own interests. Nowadays we most likely tend to online for our uses and information gain and Readers also have their books available in E-Libraries hence Book recommendation system is presided and more personalized, easily accessible Application where you get your recommendation.

**Introduction:**

There are number of choices around there is need to filter prioritize and efficiently Categorize our priorities in order to find our best Interests. Recommendation system is used to search through a vast information and make recommendations to the users with personalized services.

Book Recommendation systems are the applications where the user get their personalized recommended book according to one's interests. Sometimes readers wanted to try a new genre Which they were totally unaware of hence they needed to be guided and be recommended of a good book according to their personal taste. The libraries near them may or may not have the books of the reader's choice and so the readers are only confined to the libraries limited available books but in a book recommendation system we have a vast range of books Information to suggest the best book for the users need.

In this Book Recommendation system, we are using the K-Nearest Neighbor (KNN) algorithm for classification of books, The KNN algorithm stores all the available data and classifies a new data point based on the similar measures. This means when new data appears. Then it can be easily classified into a well-suited category by using K- NN algorithm.

We also use the Singular Value Decomposition (SVD) Algorithm to reduce the cell size of our data. The SVD Algorithm by reducing the cell size makes our application Faster and accurate by avoiding noise, Which Increases the Model quality of our application.

**Survey of Book Recommendation system:**

# 1.Simple Popularity based Recommendation System

A simple popularity-based recommendation system can be built based on count of user ratings for different books. i.e., the books are recommended according to the popularity of the author or the popularity of the book

### 2. Content-based recommendation system. Content-based recommendation systems recommend items to a user by using the similarity of items. This recommender system recommends products or items based on their description or features. It identifies the similarity between the products based on their descriptions. It also considers the user's previous history in order to recommend a similar product.

3. Okon et.al. (2018) proposed a model that generates recommendations to users, through an enhanced CF algorithm, a quick sort algorithm and Object-Oriented Analysis and Design Methodology (OOADM). Scalability was ensured through the implementation of Firebase SQL. This system performed well on the evaluation metrics.

4. Kurmashov et.al. (2015) used Pearson correlation coefficient-based CF to provide internet-based recommendations to book readers and evaluated the system through an online survey.

5. Chatti et.al. (2013) suggested tag-based and rating based CF recommendation in technology enhanced learning (TEL) to resolve the data sparsity problem and extract relevant information from the rating database. Memory and model oriented 16 varied tag-based Collaborative filtering algorithms were evaluated for buyer satisfaction and accuracy of recommendations in Personal Learning Environments.

**Dataset description**

We have taken a total of 3 datasets i.e., book dataset, user dataset and book rating dataset.

The Book Dataset contains the following columns with each more than 2lakh rows.

1)ISBN number

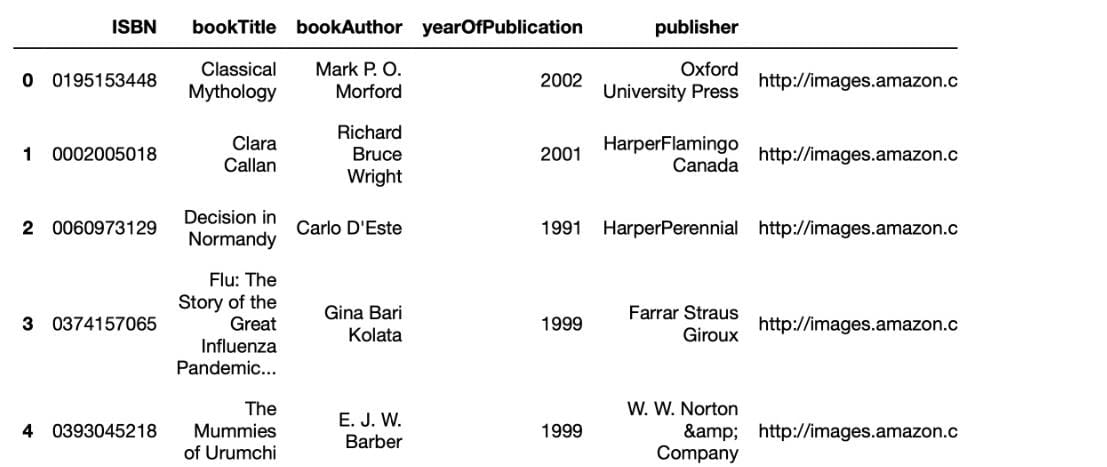
2)Book Title

3)Book Author

4)Year of Publication

5)Publisher

6)Image URLs of book

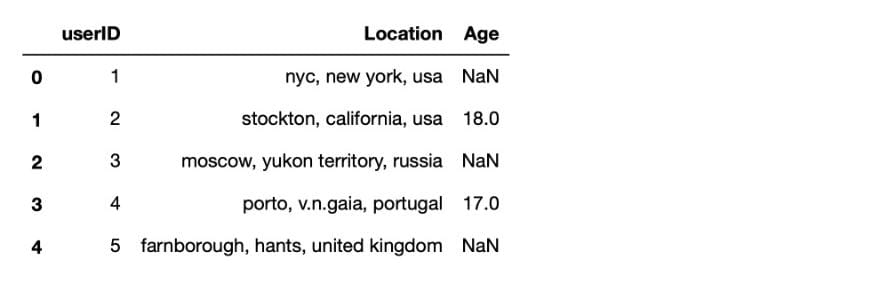


The User Dataset contains the following columns with more than 2lakh rows

1)User id

2)Location

3)Age

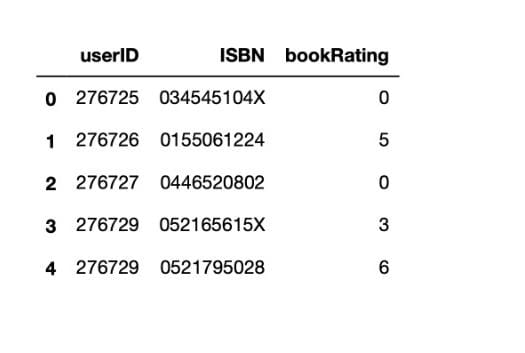


The Rating Dataset contains the following columns with more than 2 lakh rows

1)User ID

2)ISBN number

3)book rating (On a scale of 0 to 10 , 0- books the user didn’t rate )



**Data Pre-Processing**

In this step we have preprocessed the data so that the dataset is free of noise i.e we made it free of Null values and replaced implausible values with more reasonable values. In order to efficiently and accurately recommend books we narrowed down the dataset to specific regions.

The above-mentioned steps have been implemented in the code below.

**CODE**

**#import packages**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.neighbors import NearestNeighbors

from scipy.spatial.distance import correlation

from sklearn.metrics.pairwise import pairwise\_distances

import ipywidgets as widgets

from IPython.display import display, clear\_output

from contextlib import contextmanager

import warnings

warnings.filterwarnings('ignore')

import numpy as np

import os, sys

import re

from scipy.sparse import csr\_matrix

**#import data**

books = pd.read\_csv('BX-CSV-Dump/BX-Books.csv', sep=';', error\_bad\_lines=False, encoding="latin-1")

books.columns = ['ISBN', 'bookTitle', 'bookAuthor', 'yearOfPublication', 'publisher', 'imageUrlS', 'imageUrlM', 'imageUrlL']

users = pd.read\_csv('BX-CSV-Dump/BX-Users.csv', sep=';', error\_bad\_lines=False, encoding="latin-1")

users.columns = ['userID', 'Location', 'Age']

ratings = pd.read\_csv('BX-CSV-Dump/BX-Book-Ratings.csv', sep=';', error\_bad\_lines=False, encoding="latin-1")

ratings.columns = ['userID', 'ISBN', 'bookRating']

**#Data Preprocessing**

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

books.columns= books.columns.str.strip().str.replace('-', '\_')

users.columns= users.columns.str.strip().str.replace('-', '\_')

ratings.columns= ratings.columns.str.strip().str.replace('-', '\_')

# *printing all year of publication to see whether there is error in yearofpublication*

books.yearOfPublication.unique()

books.loc[books.yearOfPublication == 'DK Publishing Inc', :]#found error

# *Data Cleaning, replace irrelevant values with most probable values*

books.loc[books.ISBN == '0789466953','yearOfPublication '] = 2000

books.loc[books.ISBN == '0789466953','bookAuthor '] = "James Buckley"

books.loc[books.ISBN == '0789466953','publisher '] = "DK Publishing Inc"

books.loc[books.ISBN == '0789466953','bookTitle '] = "DK Readers: Creating the X-Men, How Comic Books Come to Life (Level 4: Proficient Readers)James Buckley"

books.loc[books.ISBN == '078946697X','yearOfPublication '] = 2000

books.loc[books.ISBN == '078946697X','bookAuthor '] = "JMichael Teitelbaum"

books.loc[books.ISBN == '078946697X','publisher '] = "DK Publishing Inc"

books.loc[books.ISBN == '078946697X','bookTitle '] = "DK Readers: Creating the X-Men, How It All Began (Level 4: Proficient Readers)\";Michael Teitelbaum"

*#Found another error*

books.loc[books.yearOfPublication == 'Gallimard', :]

*#cleaning*

books.loc[books.ISBN == '2070426769','yearOfPublication '] = 2003

books.loc[books.ISBN == '2070426769','bookAuthor '] = "Jean-Marie Gustave"

books.loc[books.ISBN == '2070426769','publisher '] = "Gallimard"

books.loc[books.ISBN == '2070426769','bookTitle '] = "Peuple du ciel, suivi de Les Bergers"

*#to convert yearofpublication into numeric*

books.yearOfPublication = pd.to\_numeric(books.yearOfPublication, errors = 'coerce')

*#printing yearofpublication*

print (sorted(books['yearOfPublication'].unique()))

# *replacing the years which have more than 2006 because dataset belongs to 2004 year*

books.loc[(books.yearOfPublication > 2006) | (books.yearOfPublication == 0), 'yearOfPublication'] = np.NAN

books.yearOfPublication.fillna(round(books.yearOfPublication.mean()), inplace = True)

books.yearOfPublication = books.yearOfPublication.astype(np.int32)

#checking the author data has null values or not

books.loc[(books['bookAuthor'].isnull()),: ]

#replacing with other for author which having NULL

books.loc[(books['ISBN'] == '9627982032'),'bookAuthor'] = 'other'

# Same with publisher

books.loc[books.publisher.isnull(),:]

books.loc[(books.ISBN == '193169656X'), 'publisher'] = 'other'

books.loc[(books.ISBN == '1931696993'), 'publisher'] = 'other'

# Printing Users Data

print(users.shape)

print(list(users.columns))

users.head()

#Age filtering who have more than 90 and less than 5

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

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

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

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

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

*#Ratings Dataset*

ratings.shape

print(ratings.shape)

print(list(ratings.columns))

ratings.head()

*# Ratings Distribution*

new\_ratings = ratings[ratings.ISBN.isin(books.ISBN)]

new\_ratings = new\_ratings[new\_ratings.userID.isin(users.userID)]

new\_ratings.head()

#Checking that the chance of book not missing after merging two datasets

sparsity = 1.0 - len(new\_ratings)/float(n\_users\*n\_books)

print( 'The sparsity level of Book Crossing Dataset is' + str(sparsity\*100)+'%')

ratings.bookRating.unique()

**# Implementing the Book Recommendation Using KNN Algorithm**

*#combine book data with rating data.*

combine\_book\_rating = pd.merge(ratings, books, on = 'ISBN')

columns = ['bookAuthor','yearOfPublication', 'publisher']

combine\_book\_rating = combine\_book\_rating.drop(columns, axis = 1)

combine\_book\_rating.head()

#group by book titles and create a new column for total rating count.

combine\_book\_rating = combine\_book\_rating.dropna(axis = 0, subset = ['bookTitle'])

book\_ratingcount = (combine\_book\_rating.

groupby(by = ['bookTitle',])['bookRating'].count().reset\_index().

rename(columns = {'bookRating':'TotalRatingCount'})

[['bookTitle','TotalRatingCount']])

book\_ratingcount.head()

*#Combine the rating data with the total rating count data, this gives us exactly # what we need to filter out the less known books.*

rating\_with\_totalratingcount = combine\_book\_rating.merge(book\_ratingcount, left\_on = 'bookTitle', right\_on = 'bookTitle', how = 'inner' )

pd.set\_option('display.float\_format', lambda x: '%.3f' % x)

print(book\_ratingcount[['TotalRatingCount']].describe())

*#The median book has been rated only once. Let’s look at the top of the distribution:*

print(book\_ratingcount['TotalRatingCount'].quantile(np.arange(.9,1,.01)))

*#About 1% of the books received 50 or more ratings. Because we have so many #books in our data, we will limit it to the top 1%, and this will give us 2713 unique #books.*

popularity\_threshold = 50

rating\_popular\_book = rating\_with\_totalratingcount.query('TotalRatingCount >= @popularity\_threshold')

rating\_popular\_book.head()

*#Filtering to users in US and Canada only*

*# In order to improve computing speed, and not run into the “MemoryError”issue, #we are limiting our user data to those in the US and Canada. And then combine the #user data with rating data and total rating count data.*

combined = rating\_popular\_book.merge(users, left\_on = 'userID', right\_on = 'userID', how = 'left')

us\_canada\_user\_rating = combined[combined['Location'].str.contains("usa|canada")]

us\_canada\_user\_rating = us\_canada\_user\_rating.drop('Age', axis = 1)

us\_canada\_user\_rating.tail(30)

if not us\_canada\_user\_rating[us\_canada\_user\_rating.duplicated(['userID', 'bookTitle'])].empty:

initial\_rows = us\_canada\_user\_rating.shape[0]

print('Initial dataframe shape {0}'.format(us\_canada\_user\_rating.shape))

us\_canada\_user\_rating = us\_canada\_user\_rating.drop\_duplicates(['userID', 'bookTitle'])

current\_rows = us\_canada\_user\_rating.shape[0]

print('New dataframe shape {0}'.format(us\_canada\_user\_rating.shape))

print('Removed {0} rows'.format(initial\_rows - current\_rows))

us\_canada\_user\_rating\_pivot = us\_canada\_user\_rating.pivot(index = 'bookTitle',columns = 'userID', values = 'bookRating').fillna(0)

us\_canada\_user\_rating\_matrix = csr\_matrix(us\_canada\_user\_rating\_pivot.values)

*#Finding the Nearest Neighbors for the books*

from sklearn.neighbors import NearestNeighbors

model\_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')

model\_knn.fit(us\_canada\_user\_rating\_matrix)

*# Test our model and Make few Recommendations*

us\_canada\_user\_rating\_pivot.head()

query\_index = np.random.choice(us\_canada\_user\_rating\_pivot.shape[0])

distances, indices = model\_knn.kneighbors(us\_canada\_user\_rating\_pivot.iloc[query\_index, :].values.reshape(1, -1), n\_neighbors = 6)

for i in range(0, len(distances.flatten())):

if i == 0:

print('Recommendations for {0}:\n'.format(us\_canada\_user\_rating\_pivot.index[query\_index]))

else:

print('{0}: {1}, with distance of {2}:'.format(i, us\_canada\_user\_rating\_pivot.index[indices.flatten()[i]], distances.flatten()[i]))

us\_canada\_user\_rating\_pivot2 = us\_canada\_user\_rating.pivot(index = 'userID', columns = 'bookTitle', values = 'bookRating').fillna(0)

us\_canada\_user\_rating\_pivot2.head()

us\_canada\_user\_rating\_pivot2.shape

X = us\_canada\_user\_rating\_pivot2.values.T

X.shape

*#Now we are using SVD from Surprise module to decrease the dataset size to only* important values

import sklearn

from sklearn.decomposition import TruncatedSVD

SVD = TruncatedSVD(n\_components=12, random\_state=17)

matrix = SVD.fit\_transform(X)

matrix.shape

import warnings

warnings.filterwarnings("ignore",category =RuntimeWarning)

corr = np.corrcoef(matrix)

corr.shape

*#Taking the input from user to give the similar book recommendation which have #good ratings*

us\_canada\_book\_title = us\_canada\_user\_rating\_pivot2.columns

us\_canada\_book\_list = list(us\_canada\_book\_title)

us\_canada\_book\_list = [x.lower() for x in us\_canada\_book\_list]

s=input("Please Enter the book to get the recommendation of similar books : ")

s=s.lower()

coffey\_hands = us\_canada\_book\_list.index(s)

corr\_coffey\_hands = corr[coffey\_hands]

list(us\_canada\_book\_title[(corr\_coffey\_hands<1.0) & (corr\_coffey\_hands>0.9)])

**Graphs**

**#1**

plt.rc("font", size = 15)

ratings.bookRating.value\_counts(sort = False).plot(kind = 'bar')

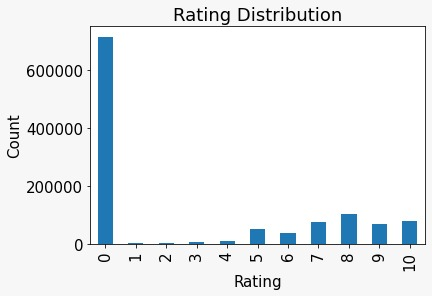
plt.title('Rating Distribution')

plt.xlabel('Rating')

plt.ylabel('Count')

plt.plot()

plt.savefig("Ratings Distribution.jpg", bbox\_inches = "tight", dpi = 100)



#**2**

users.Age.hist(bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100])

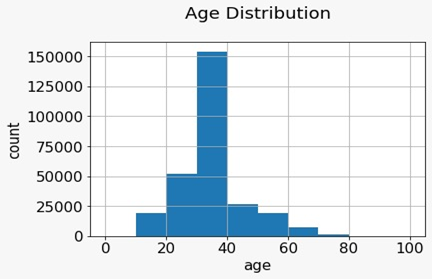
plt.title('Age Distribution\n')

plt.xlabel('age')

plt.ylabel('count')

plt.savefig('age\_dist.png', bbox\_inches='tight')

plt.show()



**#3**

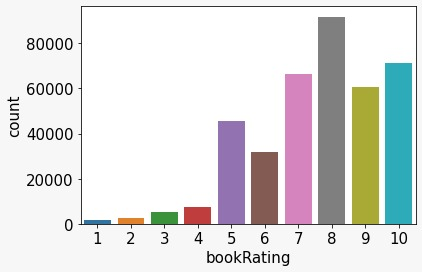
new\_ratings = ratings[ratings.ISBN.isin(books.ISBN)]

new\_ratings = new\_ratings[new\_ratings.userID.isin(users.userID)]

ratings\_explicit = new\_ratings[new\_ratings.bookRating != 0]

sns.countplot(data = ratings\_explicit, x = 'bookRating')

plt.show()



**#4**

year = books.yearOfPublication.value\_counts().sort\_index()

year = year.where(year>5)

plt.figure(figsize=(10, 8))

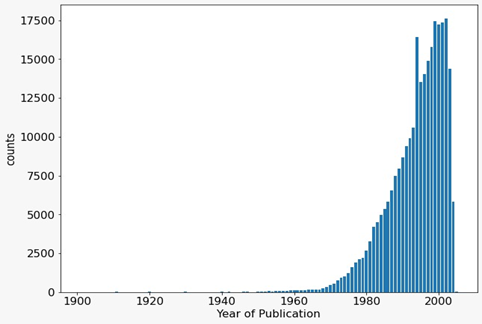
plt.rcParams.update({'font.size': 15})

plt.bar(year.index, year.values)

plt.xlabel('Year of Publication')

plt.ylabel('counts')

plt.show()



**Conclusion:** This application of Book recommendation system is easily used by the users to get best book of their Interest. In this application we used KNN and SVD Algorithm for the best results to advance the performance and accuracy of system. We also have discovered areas that are open to many further improvements, and where there is a lot of Scope to update our application further.

### **References:**

1. Simple popularity-based recommendation system:  
   <https://towardsdatascience.com/my-journey-to-building-book-recommendation-system-5ec959c41847>
2. Content-based recommendation system:  
   <https://www.kdnuggets.com/2020/07/building-content-based-book-recommendation-engine.html>
3. Okon, E.U., Eke, B.O. and Asagba, P.O. (2018):  
    An improved online book recommender system using collaborative filtering algorithm. International Journal of Computer Applications (0975- 8887) Volume 179-No.46, June 2018
4. Kurmashov, N., Konstantin, L., Nussipbekov, A. (2015):  
   Online book recommendation System. Proceedings of Twelve International Conference on Electronics Computer and Computation (ICECC)
5. Chatti, M.A., Dakova, S., Thus, H. and Schroeder, U. (2013):  
   Tag-Based Collaborative Filtering Recommendation in Personal Learning Environments. IEEE Transactions on Learning Technologies, Vol. 6, No. 4, October-December 2013