# CAPSTONE PROJECT - 04

# BOOK RECOMMENDATION SYSTEM

**ROHAN A G**

**Data Science Trainee, AlmaBetter ,**

**Bangalore**

## ABSTRACT:

Recommendation systems is used for the purpose of suggesting items to purchase or to see. They direct users towards those items which can meet their needs through cutting down large database of Information. A various techniques have been introduced for recommending items i.e. content, collaborative and hybrid techniques are used. This paper solves the problem of data sparsity problem by combining the collaborative-based filtering and SVD(Single Value Decomposition) to achieve better performance and the trends are achieved by principal of popularity. The results obtained are demonstrated and the proposed recommendation algorithms perform better and solve the challenges such as data sparsity and understanding the metric evaluation.

## 1. PROBLEM STATEMENT :

During the last few decades, with the rise of YouTube, Amazon, Netflix, and many other such web services, recommender systems have taken more and more place in our lives. From e-commerce (suggest to buyers articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences), recommender systems are today unavoidable in our daily online journeys.

In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy, or anything else depending on industries).

Recommender systems are really critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors. The main objective is to create a book recommendation system for users.

Personal recommendation systems have been emerged to conduct effective search which related booksbased on user rating and interest.The proposed system used the K-NN Cosine Distance function to measure distance and Cosine Similarity function to find Similarity between the book clusters also we implemented SVD system that give us good recommandation.

## 2. INTRODUCTION:

During the last few decades, with the rise of YouTube, Amazon, Netflix, and many other such web services, recommender systems have taken more and more place in our lives. From e-commerce (suggest to buyers articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences), recommender systems are today unavoidable in our daily online journeys.

In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy, or anything else depending on industries).

Personal recommendation systems have been emerged to conduct effective search which related books based on user rating and interest. The proposed system used the K-NN
K-NN Cosine Distance function to measure distance and Cosine Similarity function to find Similarity between the books clusters also we implemented SVD system that give us good recommendation.

## 3. DATASET INFORMATION:

- **Users :** This Csv file contains the users. Note that user ID's have been anonymized and map to integers. Location and Age is available.

- **Books :** Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. More over some content-based information is given (Book-Title, year-of-Publication, Publisher) etc. obtained from Amazon Web Services.

- **Ratings** : Ratings Dataset contains Rating information, Ratings(Book-Ratings) are either explicit, expressed on a scaler from 1-10.

## 4. STEPS INVOLVED:

The next chapters have the following sections which are the steps involved for solving the Book Recommendation System,

Section 1 - Data collection
Section 2 - Data preparation
Section 3 - Exploratory data analysis
Section 4 - Feature Engineering Section
5 - Working different models Section 6 - Evaluating
model

## DATA COLLECTION AND DATA PREPARATION:

## Data Summary:

We are using Book-Crossing dataset to train and test our recommendation system. Book-Crossings is a book ratings dataset compiled by Cai-Nicolas Ziegler. It contains
1.1 million ratings of 270,000 books by 90,000 users. The ratings are on a scale from 1 to 10. The Book-Crossing dataset comprises 3 files.

● Users

This .csv file contains the users. Note that user IDs (User-ID) have been anonymized and map to integers. Demographic data is provided (Location, Age) if available. Otherwise, these fields contain NULL values.

● Books

Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given (Book- Title, Book-Author, Year-Of-Publication, Publisher), obtained from Amazon Web Services. Note that in the case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavors (Image-URL- S, Image-URL-M, Image-URL-L), i.e., small, medium, large. These URLs point to the Amazon website.

● Ratings

Contains the book rating information. Ratings (Book-Rating) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0

## Data Collection

Before building any machine learning model, it is vital to understand what the data is, and what are we trying to achieve. Data exploration reveals the hidden trends and insights and data preprocessing makes the data ready for use by ML algorithms.

So, let's begin. . .

To proceed with the problem dealing first we will load our dataset that is given to us in a .csv file into a data frame.

Mount the drive and load the csv file into a data frame

**Importing required python libraries**.

```
import pandas as pd
import numpy as np
import scipy
import math
import random
import sklearn

from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
```

# EXPLORATORY DATA ANALYSIS

The primary goal of EDA is to support the analysis of data prior to making any conclusions. Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods.

## Dimension of dataset

```
#checking shapes of the datasets
print(books.shape)
print(users.shape)
print(ratings.shape)
```

```
(271360, 8)
(278858, 3)
(1149780, 3)
```

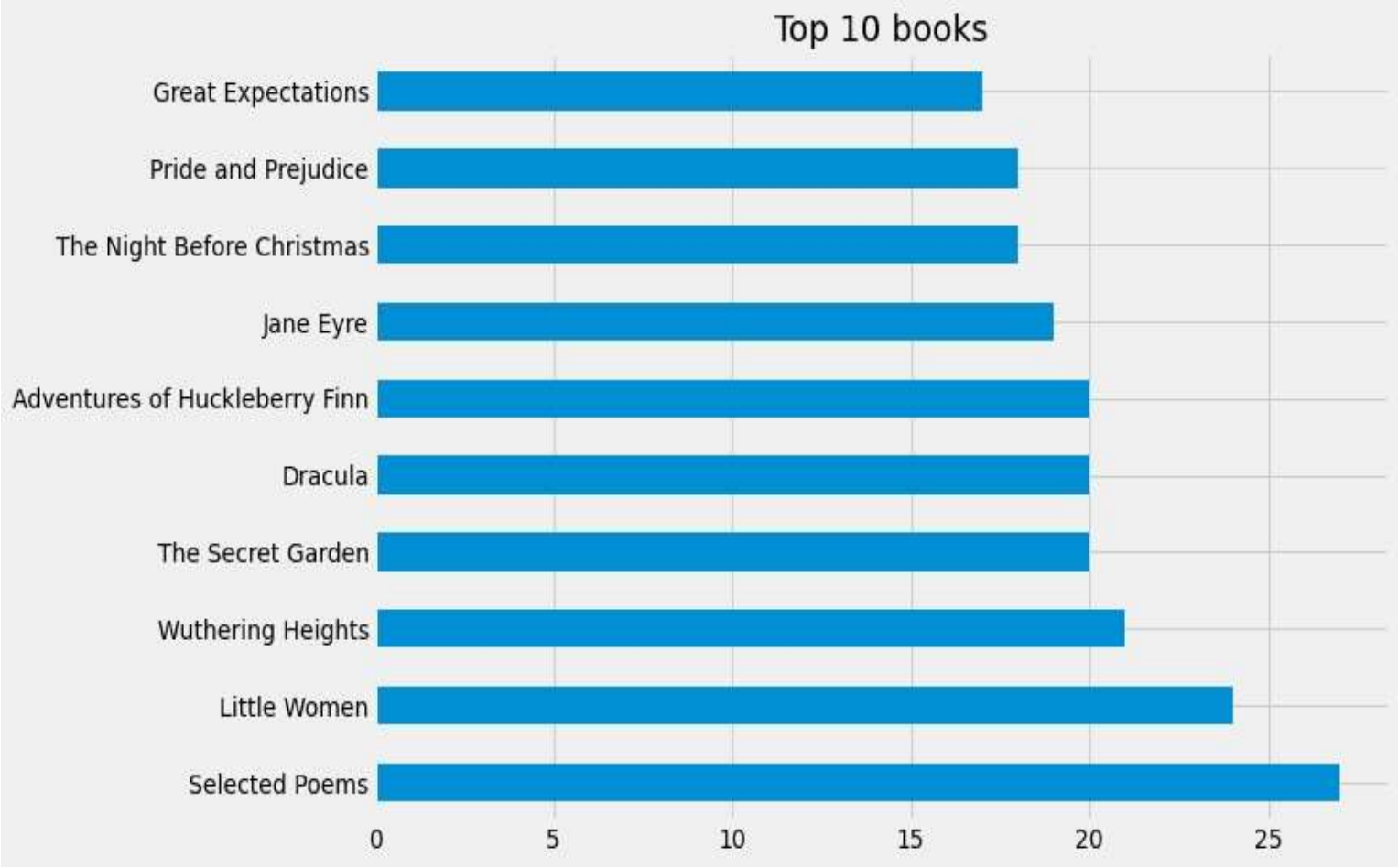
## Books Dataset:

In the books dataset we have the following feature variables.

- ISBN (unique for each book)
- Book-Title
- Book-Author
- Year-Of-Publication
- Publisher
- Image-URL-S
- Image-URL-M ● Image-URL-L

Let’s find the top 10 Book-Author and top 10 Books. Further we find that both the plots are skewed and the maximum number of books are from top 10 Book-Authors and top 10 Books.

From bar plot we observed that Agatha Christie wrote highest number of books in our given dataset selected poems book is ranked one in the list of top 10 books.



## Users_Dataset:

In the users_dataset we have the following feature variables.

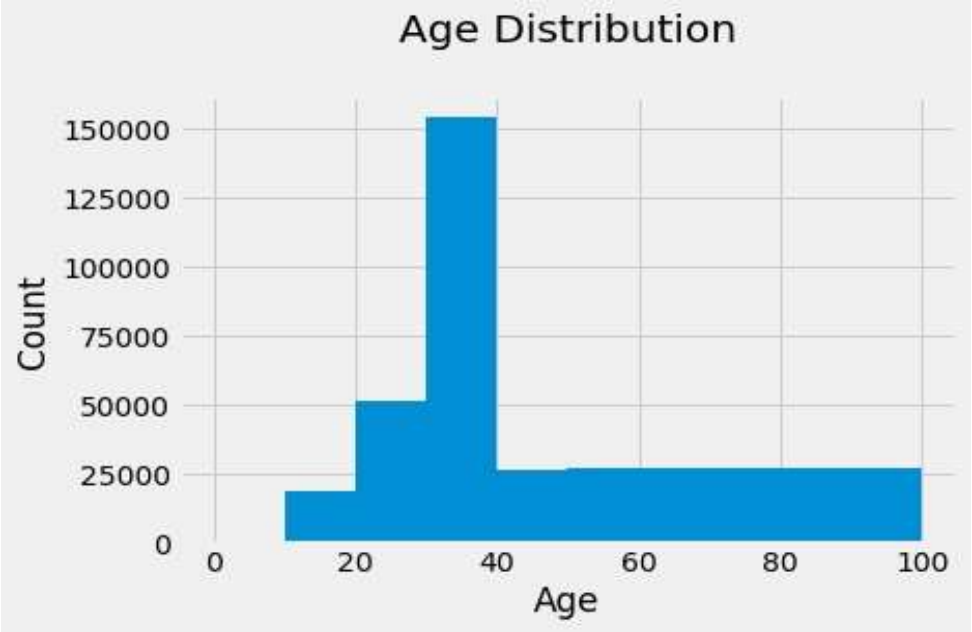
- User-ID (unique for each user)
- Location (contains city, state and country separated by commas)
- Age

Out of these features, User-ID is unique for each user, Location contains city, state and country separated by commas and we have Age given across each user.

| | user_id | location | age |
|---|---|---|---|
| 0 | 1 | nyc, new york, usa | NaN |
| 1 | 2 | stockton, california, usa | 18.0 |
| 2 | 3 | moscow, yukon territory, russia | NaN |
| 3 | 4 | porto, v.n.gaia, portugal | 17.0 |
| 4 | 5 | farnborough, hants, united kingdom | NaN |

**Age:**

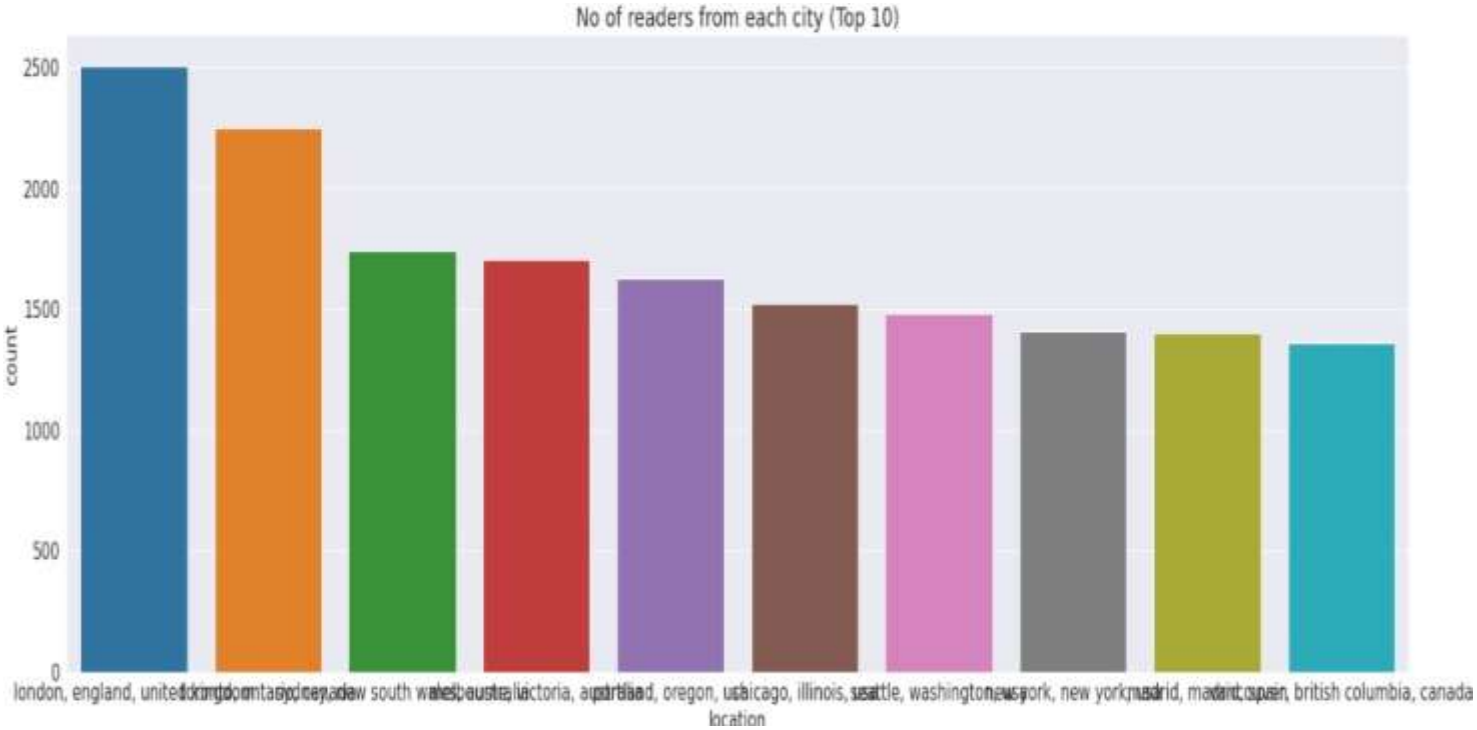
Let's understand the Age distribution of the given user dataset. We can conclude between users with age between 20-40 are highest in number.




We observed that 41.9% of age 34 group read more books compared to other age groups. Also the users with the age 60 and above do not read more books.

**Location:**

Here we can observe that user with locations London, England, united kingdom, Toronto, Ontario, Canada are high in numbers.




## Ratings Dataset:

In the ratings_dataset we have the following feature variables.

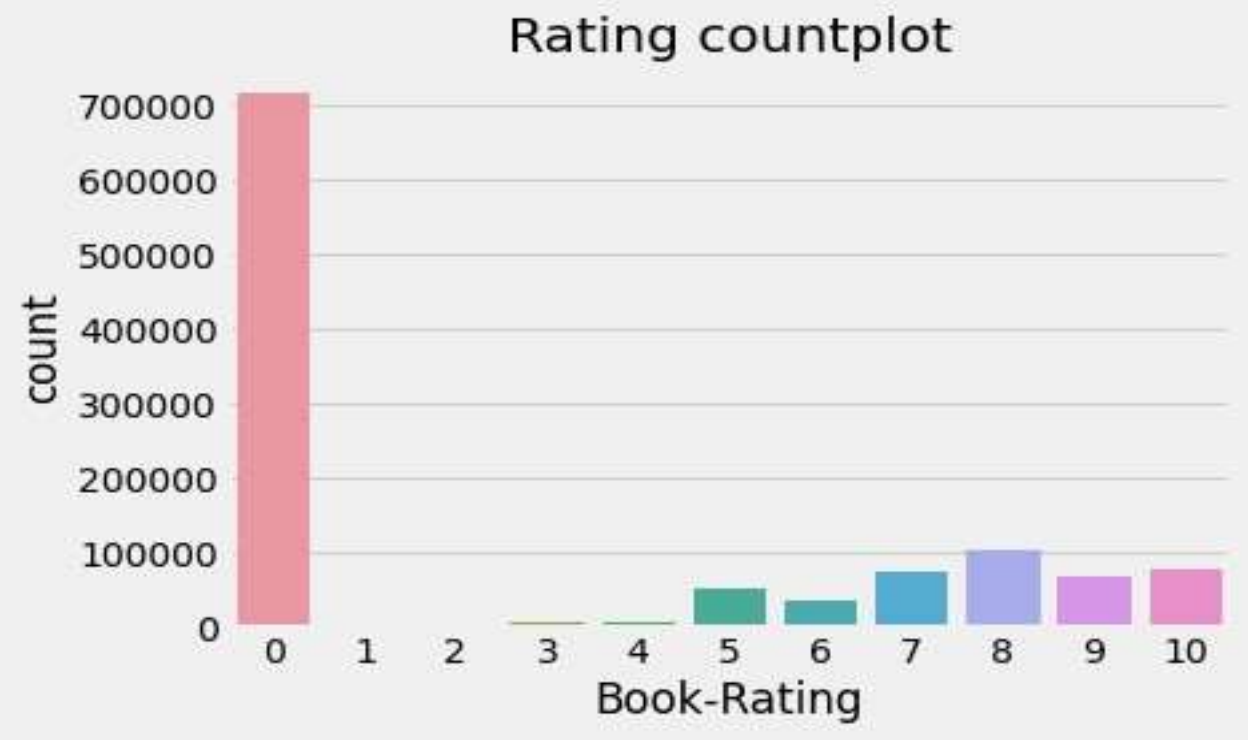
- User-ID

- ISBN
- Book-Rating

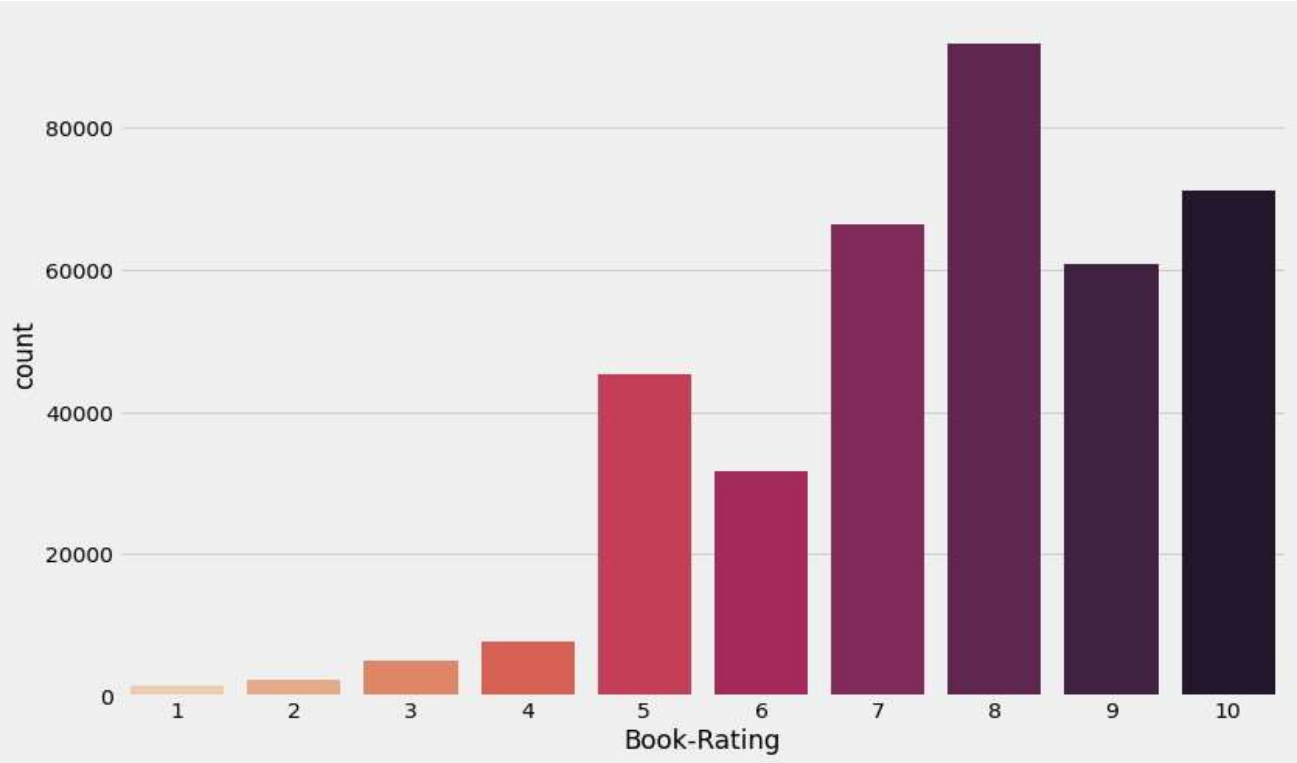
| | user_id | ISBN | rating |
|---|---|---|---|
| 0 | 276725 | 034545104X | 0 |
| 1 | 276726 | 0155061224 | 5 |
| 2 | 276727 | 0446520802 | 0 |
| 3 | 276729 | 052165615X | 3 |
| 4 | 276729 | 0521795028 | 6 |

**Ratings Distribution:**

Let's find the distribution of ratings frequency in our ratings dataset. From the following frequency plot, we find that most of the ratings are 0 which is implicit rating



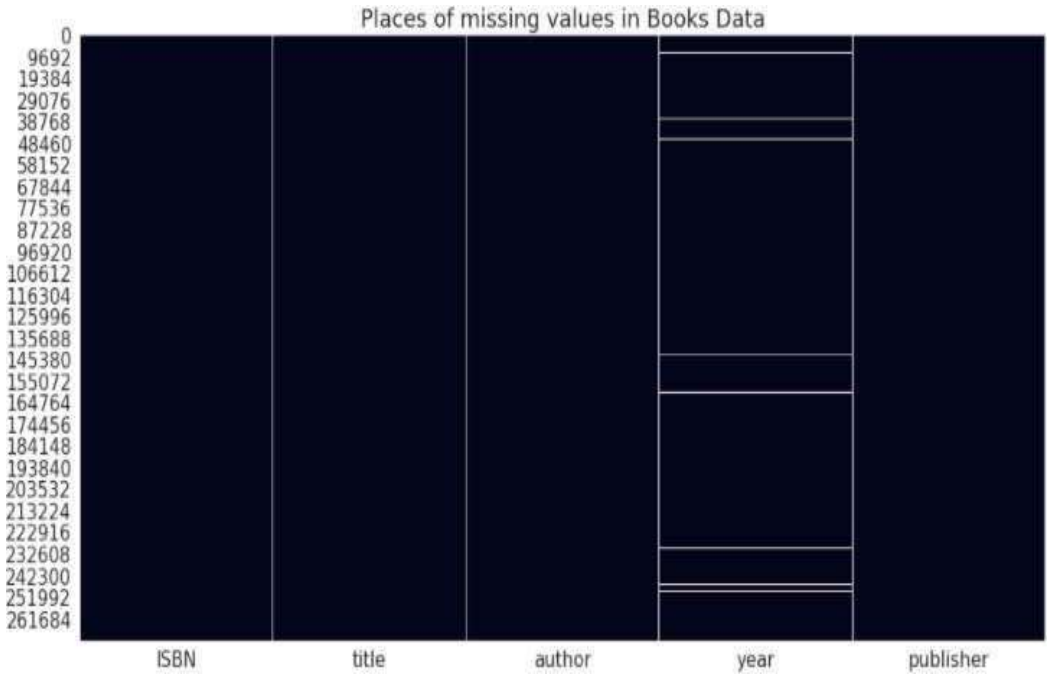

The ratings are very unevenly distributed, and the vast majority of ratings are 0 .As mentioned in the description of the dataset - BX-Book-Ratings contains the book rating information. Ratings are either explicit, expressed on a scale from 1-10 higher values or implicit, expressed by 0. Hence segregating implicit and explicit ratings dataset.




It can be observed that higher ratings are more common amongst users and rating 8 has been rated the highest number of time.

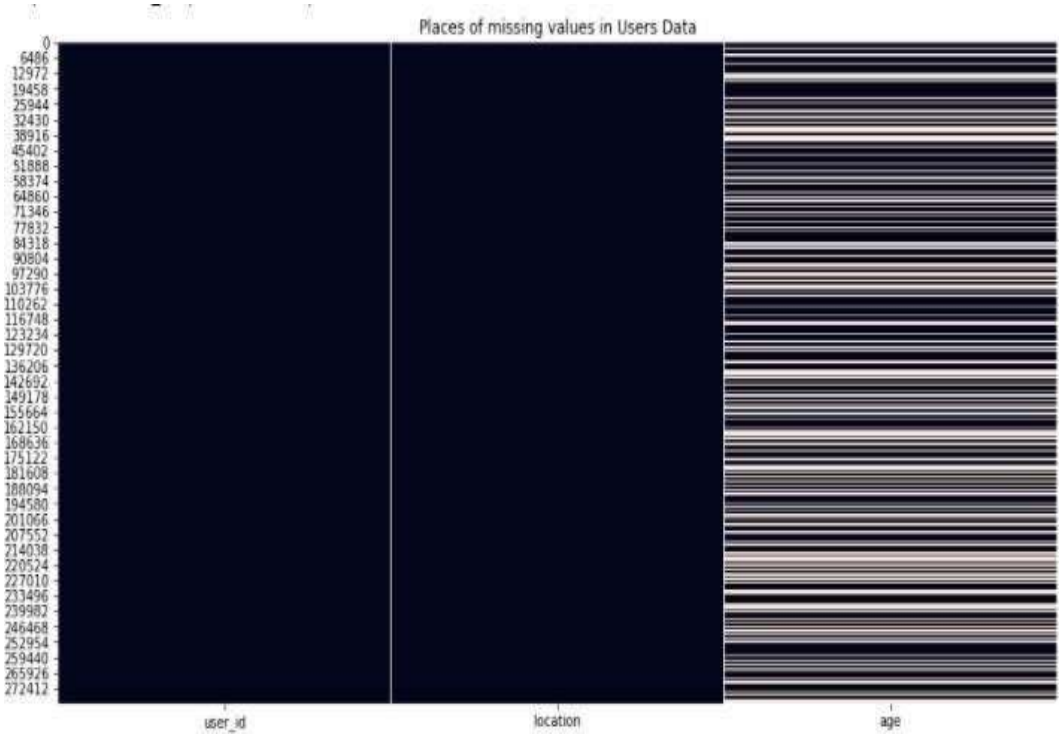
## Missing data:

Understanding the reasons why data are missing is important **for handling the remaining data correctly**.
If values are missing completely at random, the data sample is likely still representative of the population.
But if the values are missing systematically, analysis may be biased.
There were missing values in some columns in our dataset.
**Book Dataset**:



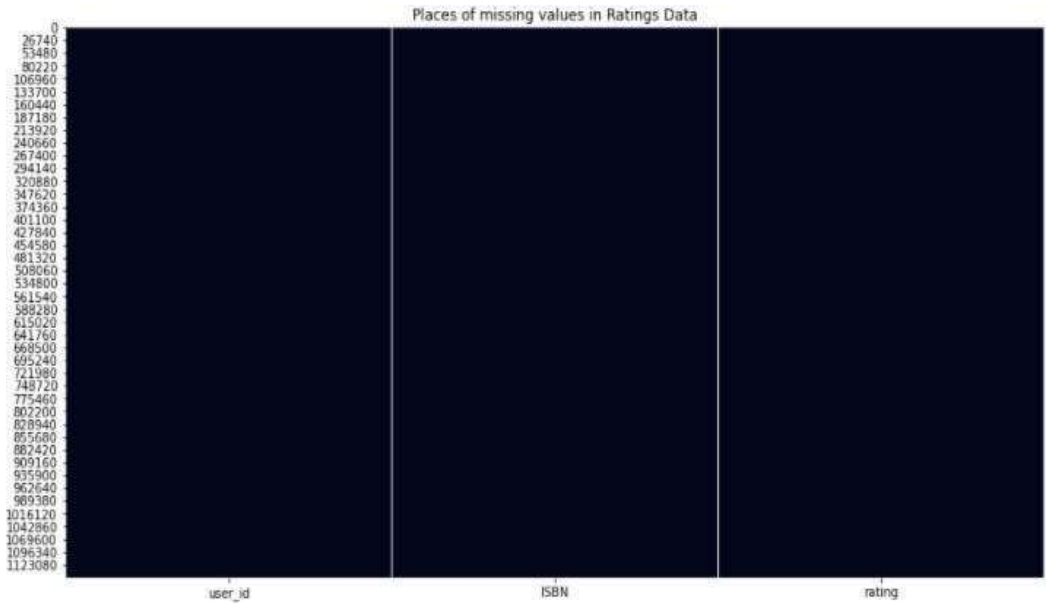

The book dataset ‘year’ column have **4690** missing values.

**User Dataset:**




The user dataset ‘Age’ column have 40% missing values.

**Ratings Dataset:**




The ratings dataset don’t have missing value.

## Data Cleaning:

In the data cleaning section we will drop off the features which do not contribute towards making a good recommendation system.

**Books_df:**

We have already seen in our EDA part that the Image-URL-S, Image-URL-M and Image-URL-L do not contribute towards our final goal. So, we will drop off these columns and also dropping these columns we will not be losing any information.

We will rename the columns of each file. Because the name of the column contains space, and uppercase letters so we will correct as to make it easy to use. Let’s simplified the column names.

```
# Renaming column name of 'books', 'users' & 'ratings' dataset
books.rename(columns = {'Book-Title':'title', 'Book-Author':'author', 'Year-Of-Publication':'year', 'Publisher':'publisher'}, inplace=True)
users.rename(columns = {'User-ID':'user_id', 'Location':'location', 'Age':'age'}, inplace=True)
ratings.rename(columns = {'User-ID':'user_id', 'Book-Rating':'rating'}, inplace=True)
```

# 5. Popularity Based Approach

It is a type of recommendation system which works on the principle of popularity and or anything which is in trend. These systems check about the books which are in trend or are most popular among the users and directly recommend them.

For example, if a product is often purchased by most people then the system will get to know that that product is most popular so for every new user who just signed it, the system will recommend that product to that user also and chances become high that the new user will also purchase that.

**Merits of Popular based recommendation system:**

The popularity-based recommendation system eliminates the need for knowing other factors like user browsing history, user preferences, genre, and other factors. Hence, the single-most factor considered is the star rating to generate a scalable recommendation system. This increases the chances of user engagement as compared to when there was no recommendation system.

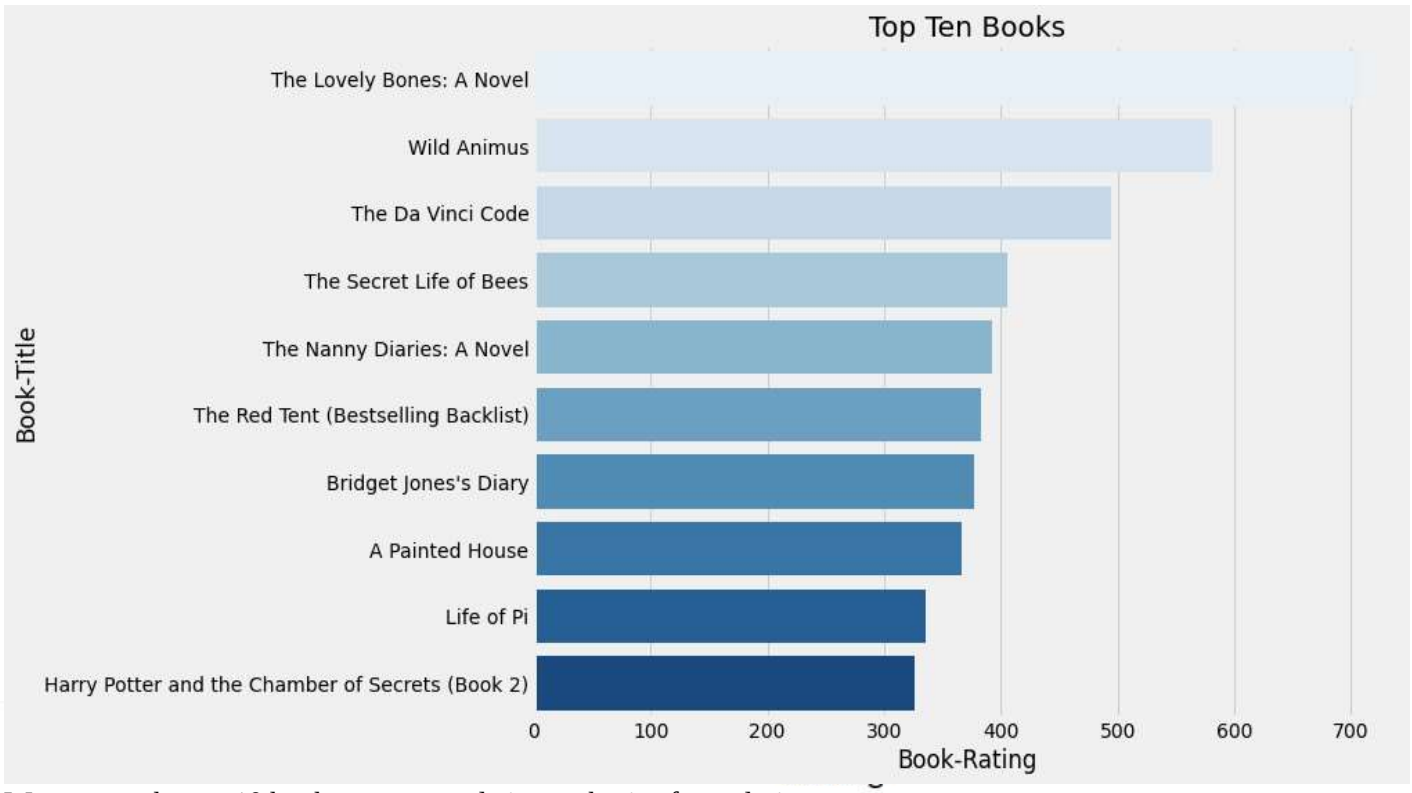
Cold start is a potential problem in computer-based information systems which involves a degree of automated data modeling. Specifically, it concerns the issue that the system cannot draw any inferences for users or items about which it has not yet gathered sufficient information. There are three cases of cold start:

- **New community:** refers to the start-up of the recommender, when, although a catalogue of items might exist, almost no users are present and the lack of user interaction makes it very hard to provide reliable recommendations.
- **New item:** a new item is added to the system, it might have some content information but no interactions are present.
- **New user:** a new user registers and has not provided any interaction yet, therefore it is not possible to provide personalized recommendations.

A popularity based model does not suffer from cold start problems which means on day 1 of the business also it can recommend products on various different filters. There is no need for the user's historical data.

Now let's try to build our first recommendation system based on popularity. These systems check about the product which are in trend or are most popular among the users and directly recommend those.

```
# Plotting horizontal bar blot
plt.figure(figsize=(10, 8))
g = sns.barplot(x='Book-Rating',y='Book-Title',data=top_ten_books, orient='h', palette="Blues")
plt.title("Top Ten Books")
```




We can see the top 10 books recommendation on basis of popularity.

| | title |
|---|---|
| The Lovely Bones: A Novel | 707 |
| Wild Animus | 581 |
| The Da Vinci Code | 494 |
| The Secret Life of Bees | 406 |
| The Nanny Diaries: A Novel | 393 |
| The Red Tent (Bestselling Backlist) | 383 |
| Bridget Jones's Diary | 377 |
| A Painted House | 366 |
| Life of Pi | 336 |
| Harry Potter and the Chamber of Secrets (Book 2) | 326 |

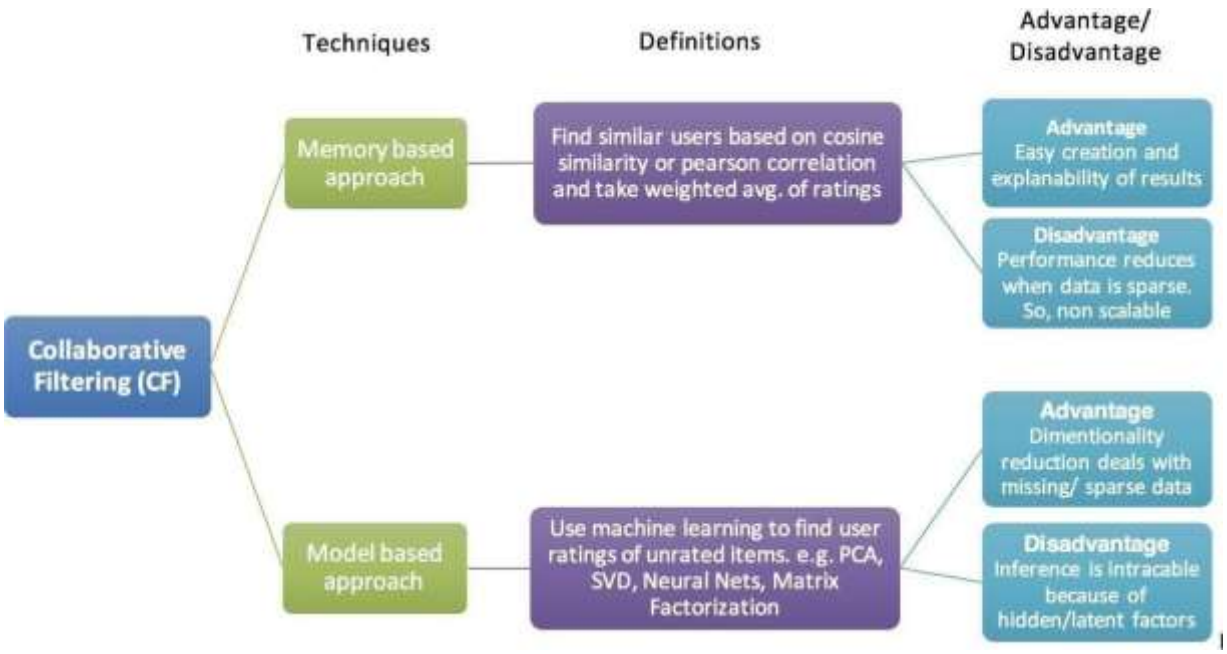
# 6. Collaborative Filtering:

Collaborative filtering (CF) systems work by collecting user feedback in the form of ratings for items in a given domain and exploiting similarities in rating behavior among several users in determining how to recommend an item.CF accumulates customer product ratings, identifies customers with common ratings, and offers recommendations based on inter-customer comparisons. It's based on the idea that people who agree in their evaluations of certain items in the past are likely to agree again in the future. For example, most people ask their trusted friends for restaurant or movie suggestions.

Collaborative filtering models are based on an assumption that people like things similar to other things they like, and things that are liked by other people with similar taste.

Two major approaches of collaborative filtering

1. **Memory-based approach**: This approach is based on taking a matrix of preferences for items by users using this matrix to predict missing preferences and recommend items with high predictions
2. **Model-based approach**: In this approach, CF models are developed using machine learning algorithms to predict a user's rating of unrated items. Some of these models/techniques include: k- nearest neighbors, clustering, matrix factorization.

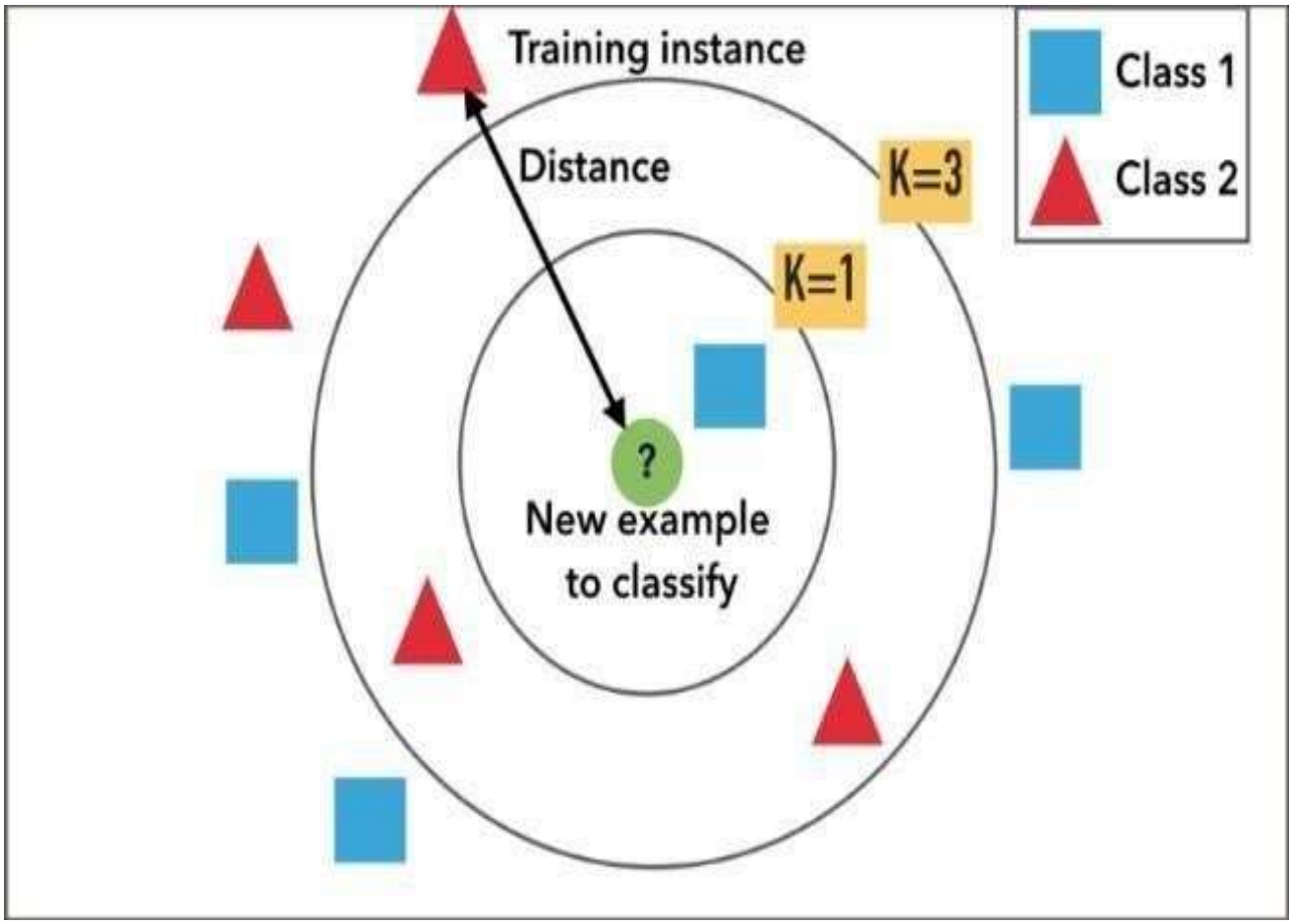



Types of collaborative filtering approaches

## Collaborative Filtering Using KNN (k-Nearest Neighbors)

**KNN (k-Nearest Neighbors)** as an algorithm seems to be inspired from real life. The full k-Nearest Neighbors algorithm works much in the way some of us ask for recommendations from our friends. First, we start with people whose taste we feel we share, and then we ask a bunch of them to recommend something to us. If many of them recommend the same thing, we deduce that we'll like it as well. Our behavior is guided by the friends we grew up with kNN is a machine learning algorithm to find clusters of similar users based on common book ratings, and make predictions using the average rating of top-k nearest neighbors.

KNN does not make any assumptions on the underlying data distribution but it relies on **item feature similarity**. When KNN makes inference about a movie, KNN will calculate the "distance" between the target book and every other book in its database, then it ranks its distances and returns the top K nearest neighbor movies as the most similar book recommendations.




Here we assume that users who given ratings more than 200 are users who read at least 20 books (suppose on user given rating 10/10 so minimum he read books (200 ratings/10 ratings per book=20).Hence for statistical significance we should consider only the data of user who given more than 200 ratings.

```
#Users with more than 200 ratings
numbers1 = ratings_explicit['user_id'].value_counts()
ratings = ratings_explicit[ratings_explicit['user_id'].isin(numbers1[numbers1 >= 200].index)]
#Books with more than 100 Ratings
number2 = ratings_explicit['rating'].value_counts()
ratings = ratings_explicit[ratings_explicit['rating'].isin(number2[number2 >= 100].index)]
```

First we will the dataset 'Ratings' and 'Books' have common column 'ISBN' so create new data frame merging the two data frames then we will group by book titles and create a new column for total rating count.

```
#Merging the dataframes
totalRatingCount_df = books_with_rating.merge(rating_count_df, left_on = 'title', right_on = 'title', how = 'left')
totalRatingCount_df.head()
```

| | user_id | ISBN | rating | title | author | year | publisher | total_ratings |
|---|---|---|---|---|---|---|---|---|
| 0 | 276726 | 0155061224 | 5 | Rites of Passage | Judith Rae | 2001 | Heinle | 5 |
| 1 | 276729 | 052165615X | 3 | Help!: Level 1 | Philip Prowse | 1999 | Cambridge University Press | 1 |
| 2 | 276729 | 0521795028 | 6 | The Amsterdam Connection : Level 4 (Cambridge English Readers) | Sue Leather | 2001 | Cambridge University Press | 1 |
| 3 | 276744 | 038550120X | 7 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 |
| 4 | 11676 | 038550120X | 10 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 |

This gives us exactly what we need to find out which books are popular and filter out lesser-known books.

We will consider the only books having minimum total 50 ratings.

```
#let's keep threshold value 50
popularity_threshold = 50
rating_popular_book = totalRatingCount_df.query('total_ratings >= @popularity_threshold')
rating_popular_book.head()
```

| | user_id | ISBN | rating | title | author | year | publisher | total_ratings |
|---|---|---|---|---|---|---|---|---|
| 3 | 276744 | 038550120X | 7 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 |
| 4 | 11676 | 038550120X | 10 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 |
| 5 | 16877 | 038550120X | 9 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 |
| 6 | 17975 | 038550120X | 6 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 |
| 7 | 20806 | 038550120X | 6 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 |

This will be our final data frame.

```
combined.head()
```

| | user_id | ISBN | rating | title | author | year | publisher | total_ratings | location | age |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 276744 | 038550120X | 7 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 | torrance, california, usa | 34 |
| 1 | 11676 | 038550120X | 10 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 | n/a, n/a, n/a | 34 |
| 2 | 16877 | 038550120X | 9 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 | houston, arkansas, usa | 37 |
| 3 | 17975 | 038550120X | 6 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 | fargo, north dakota, usa | 34 |
| 4 | 20806 | 038550120X | 6 | A Painted House | JOHN GRISHAM | 2001 | Doubleday | 366 | union, kentucky, usa | 34 |

Wait, but how do we feed the data frame of ratings into a KNN model? First, we need to transform the data frame of ratings into a proper format that can be consumed by a KNN model. We want the data to be in an m x n array, where m is the number of books and n is the number of users. To reshape data frame of ratings, we'll pivot the data frame to the wide format with books as rows and users as columns. Then we'll fill the missing observations with 0s since we're going to be performing linear algebra operations (calculating distances between vectors). Let's call this new data frame a data frame of books features.

Now we can fit our model with pivot matrix. We used metric **cosine** and algorithm **brute**.

```
from scipy.sparse import csr_matrix
#csr is compressed sparse matrix since there are lots of NAn-->0
book_sparse = csr_matrix(pivot_matrix)
```

```
# metric used for evaluation is cosine and algorithm is brute
model = NearestNeighbors(metric = 'cosine',algorithm='brute')
```

```
#Let's fit the model
model.fit(book_sparse)
```

```
NearestNeighbors(algorithm='brute', leaf_size=30, metric='cosine',
                 metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                 radius=1.0)
```

## Testing model and making Recommendations:

In this step, the KNN algorithm measures distance to determine the “closeness” of instances. It then classifies an instance by finding its nearest neighbors, and picks the most popular class among the neighbors.

## Recommendation for random book:

```
Recommendations for Cat's Cradle:

1: Slaughterhouse Five or the Children's Crusade: A Duty Dance With Death, with distance of 0.9103445085990503:
2: Catch 22, with distance of 0.9135549316310942:
3: Cryptonomicon, with distance of 0.9187612511299216:
4: Stargirl, with distance of 0.9218718063164879:
5: The Two Towers (The Lord of the Rings, Part 2), with distance of 0.9286713029531773:
```

## Recommendation for known book from dataset:

```
Recommendations for 1984:

1: Animal Farm, with distance of 0.8441751059913439:
2: Brave New World, with distance of 0.8630224130863633:
3: The Vampire Lestat (Vampire Chronicles, Book II), with distance of 0.9098446731344783:
4: Slaughterhouse Five or the Children's Crusade: A Duty Dance With Death, with distance of 0.9102283484822355:
5: American Psycho (Vintage Contemporaries), with distance of 0.9126168848307201:
```

Here we can see that we have good recommendation the distance describes the closeness of the book with the recommended books.

## Singular Value Decomposition (SVD):

Singular value decomposition also known as the **SVD** algorithm is used as a collaborative filtering method in recommendation systems. SVD is a matrix factorization method that is used to reduce the features in the data by reducing the dimensions from N to K where (K<N).

For the part of the recommendation, the only part which is taken care of is matrix factorization that is done with the user-item rating matrix. Matrix-factorization is all about taking 2 matrices whose product is the original matrix. Vectors are used to represent item 'qi' and user 'pu' such that their dot product is the expected rating as given below

$$\text{expected rating} = \hat{r}_{ui} = q_i^T p_u$$

qi' and 'pu' can be calculated in such a way that the square error difference between the dot product of user and item and the original ratings in the user-item matrix is least.

**Benefits of using SVD?**

There are 3 primary benefits :

- It's very efficient
- The basis is hierarchical, ordered by relevance
- It tends to perform quite well for most data sets

**Surprise** is a Python scikit for building and analysing recommender systems that deal with explicit Rating data. The name **SurPRISE** (roughly) stands for Simple Python Recommendation System Engine.

Implementing SVD on our dataset along with cross validation involving 5 folds i.e. cv=5, we can see the following results:

**Recommendation by giving user-id as input:**

On picking a random user_id = 116866 our model recommend this books

```
October Light: 7.815437160287302
Lucy: The Beginnings of Humankind: 7.815437160287302
An Introduction to Stochastic Modeling: 7.815437160287302
The Biosphere.: 7.815437160287302
Algebra and Trigonometry, Unit Circle (6th Edition): 7.815437160287302
River Why: 7.815437160287302
Excel: 7.815437160287302
The Progress of Love (King Penguin): 7.557823608083711
Metamagical Themas: Questing for the Essence of Mind and Pattern: 7.423558267082783
```

Above recommended books seems pretty much related.

# CHALLENGES

- Understanding the metric for evaluation was a challenge as well.
- Decision making on missing value imputations quite challenging.
- Handling of sparsity was a major challenge.

# CONCLUSION

- Recommendation system is unturned to exist in the e-commerce businesses with the help of collaborative or content-based filtering to predict different items and yes, users are most satisfied with the products recommended to them.
- While performing Exploratory Data Analysis we observed that almost **42%** of readers with **age-34** read more books compared to other age group of readers.
- Books with publication years are somewhat between **1950 - 2005**.
- Also the readers mostly give 8 ratings (on scale 1-10) to books followed by 10 and 7.
- There are more readers from locations London, England, United Kingdom, Toronto, ontar io, Canada compare to other locations.
- KNN model gives good recommendation for books.
- The best collaborative book recommender model is **SVD(Singular value decomposition)** with best accuracy on test data which give stronger recommendations. These results show that our proposed system can remove boring books from the recommendation list more efficiently.
- Popularity based recommendation systems helpful to new users. we don't have data about new user so here popularity based recommendations are more useful
- Recommender system has the ability to predict whether a particular user would prefer an item or not based on the user's profile. Recommender systems are beneficial to both service providers and users [3]. They reduce transaction costs of finding and selecting items in an online shopping
- In Our case also Hybrid approach gives best recommendations...