**Book Recommendation System**

Bindu Kovvada

Saksham Tripathi

Manoj Patil M

Gulzar

Deepak Kumar Gautam,

**Data Science Trainees,**

**AlmaBetter, Bangalore**

**Abstract:** Recommender systems improve access to relevant products and information by making personalized suggestions based on previous examples of a user's likes and dislikes. Most existing recommender systems use social/collaborative filtering methods that base recommendations on other users' preferences. and content-based filtering methods that uses information about an item itself to make suggestions**.**

**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). Recommendation 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.

2. Datasets:

The Book-Crossing dataset comprises 3 files.

● Users 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 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 flavours (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.

3. Introduction:

There is a growing interest in recommender systems that suggest music, films, books, and other products and services to users based on examples of their likes and dislikes. A number of successful start-up companies like Tinder etc are also using this with big companies like Netflix, Facebook, Youtube etc, Online book stores like Amazon and flipkart have popular recommendation services, and many libraries have a long history of providing reader's advisory services. Such services are important since readers' preferences are often complex and not readily reduced to keywords or standard subject categories, but rather best illustrated by example. Existing recommender systems almost exclusively utilize a form of computerized matchmaking called collaborative or social filtering. The system maintains a database of the preferences of individual users, finds other users whose known preferences correlate significantly with a given user, and recommends to a person other items enjoyed by their matched users. This approach assumes that a given user's tastes are generally the same as another user of the system and that a sufficient number of user ratings are available. Items that have not been rated by a sufficient number of users cannot be effectively recommended. Unfortunately, statistics on library use indicate that most books are utilized by very few users. Therefore, collaborative approaches naturally tend to recommend popular titles, perpetuating homogeneity in reading choices. Also, since significant information about other users is required to make recommendations, this approach raises concerns about privacy and access to proprietary customer data. Learning individualized profiles from descriptions of examples (content-based recommending), on the other hand, allows a system to uniquely characterize each user without having to match their interests to someone else's. Items are recommended based on information about the item itself rather than on the preferences of other users. This also allows for the possibility of providing explanations that list content features that caused an item to be recommended; potentially giving readers confidence in the system's recommendations and insight into their own preferences .

**4. Exploratory Data Analysis:**

**Observing and Exploring Dataset**

Exploratory data analysis is a method with help of this we can understand dataset, we can create some insights from data. We can understand the statistics part of the data like mean, mode etc.

With EDA we can find null values, missing values, duplicates in the data set and outliers. We can find correlation between features in the dataset. In EDA we can perform various data visualization methods on data and observe what is happening in data.

After observing the data, we would say that

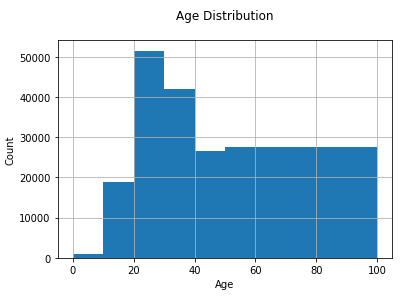
Book\_df shape is (271360, 8)

Ratings\_df shape is (1149780, 3)

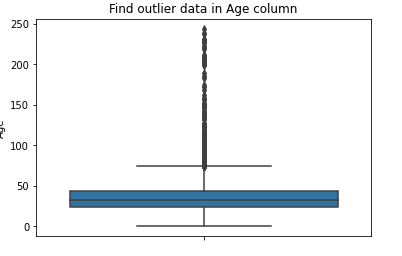
Users\_df shape is (278858, 3)

## **A- Users\_Dataset**

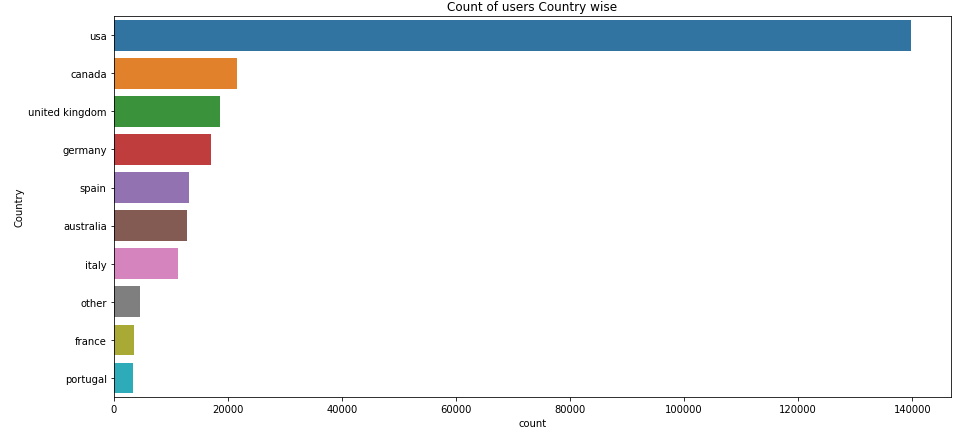
* Age has around 39% missing values.
* As we can see that most of our users are in their 20-30s.



* we can say that we have Outlier in our dataset of users



* We can clearly observe from the above countplot that most of our customers are from the US.



* **Treating Outliers:**

Age has positive Skewness (right tail)

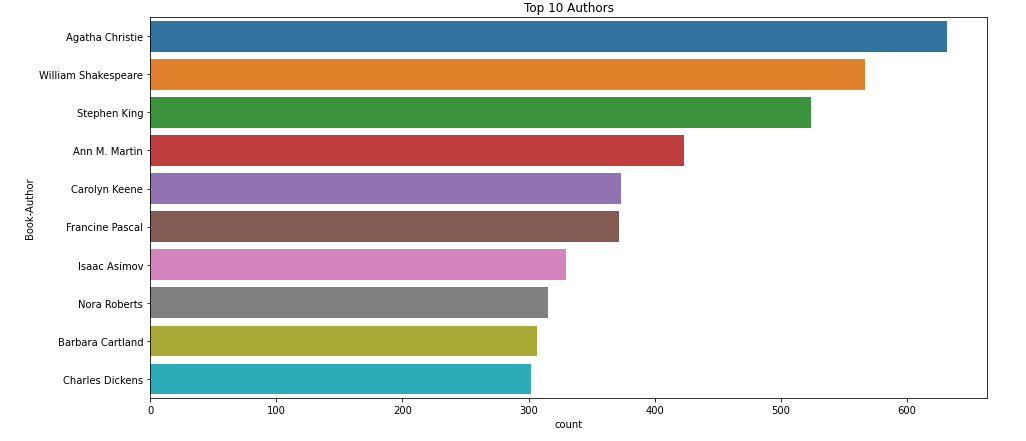
so we can use median to fill Nan values,

but for this we don't like to fill Nan value just for one range of age. To handle this we'll use the country column to fill Nan.

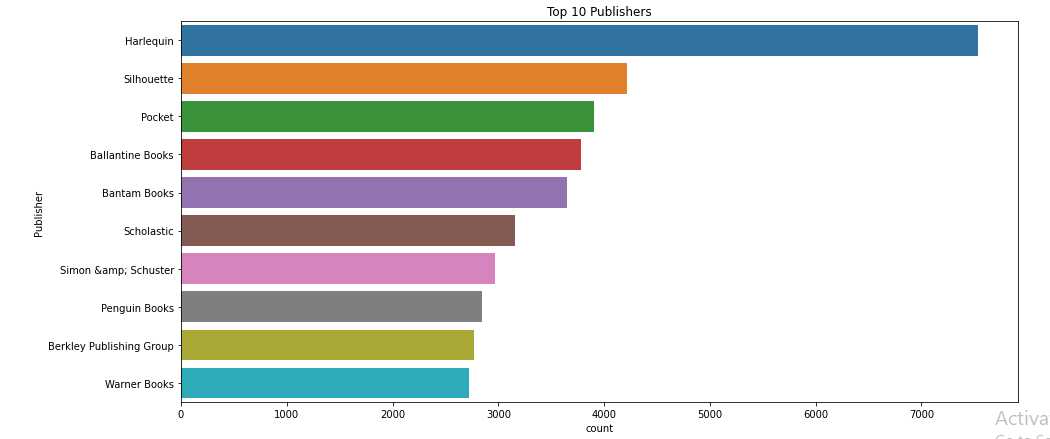
## **B- Books\_Dataset**

* We can clearly observe from the above countplot that most of our customers are

from the US.



* We can see that Agatha Christie is the author with the most books published.



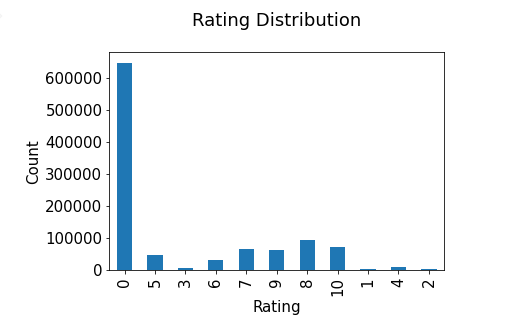
* We can see that Harlequin is the publisher with the most books published.
* The value 0 for Year-Of-Publication is invalid and as this dataset was published in 2004, We have assumed that the years after 2006 to be invalid and setting invalid years as NaN

## **C- Ratings\_Dataset**

## Ratings dataset should have books only which exist in our books dataset**.**

## Ratings dataset should have books only which exist in our books dataset.

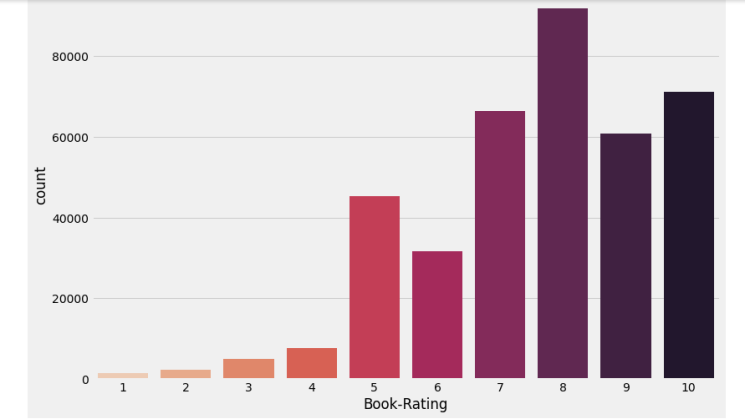
* As we can see from this above plot The ratings are very unevenly distributed, and the vast majority of ratings are 0 .As quoted in the description of the
* dataset - BX-Book-Ratings contains the book rating information. Ratings are either explicit, expressed on a scale from 1-10 higher values denoting higher appreciation, or implicit, expressed by 0.



## Hence segregating implicit and explicit ratings datasets

## ratings\_explicit dataset shape (383842, 3)

## ratings\_implicit dataset (647294, 3)



* we can observe that higher ratings are more often amongst users and rating 8 has been rated highest number of times.

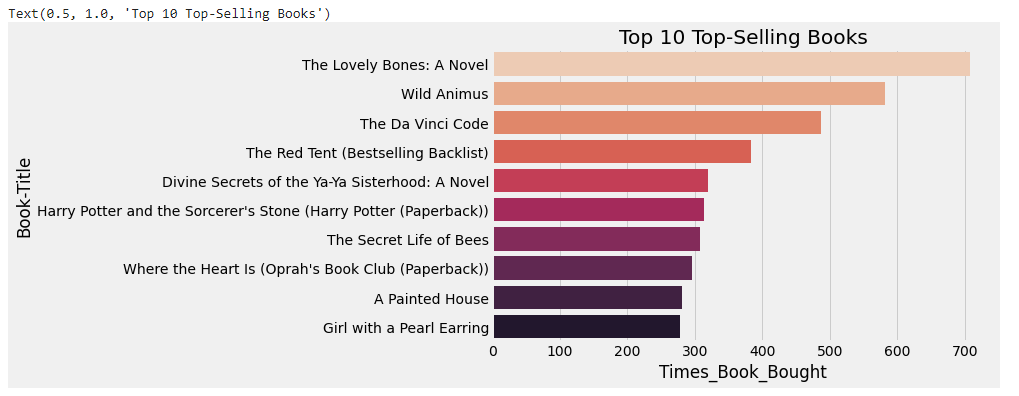
# 5. Recommendation Systems

# A. Recommendation for New Users(Cold Start)

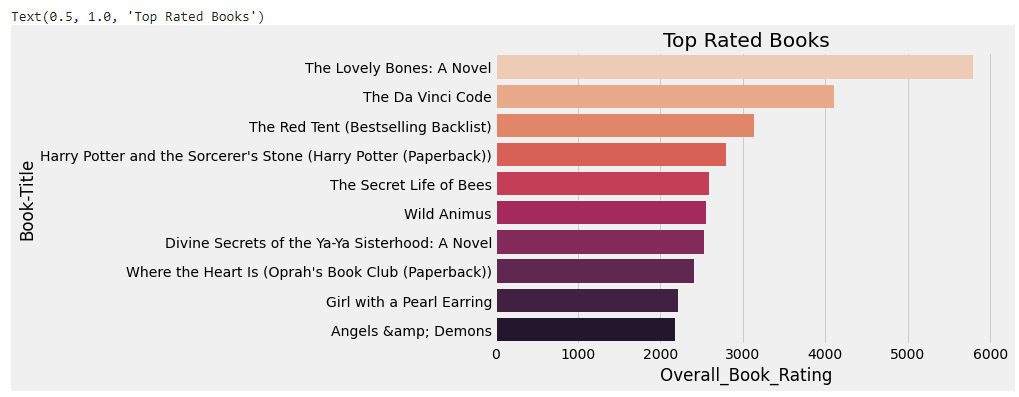
* As we all know that collaborative filtering have cold start problem so it can't recommend books for fresh new user. So we can recommend them our top read/rated books as a new user.

**Popularity Based Recommendation for new users :**

* The Popularity based recommender provides a general chart of recommended books to all the users. They are not sensitive to the interests and tastes of a particular user.



* These are our top selling books on the basis of how much time it bought by customers.



These are the top books which have the highest overall rating.



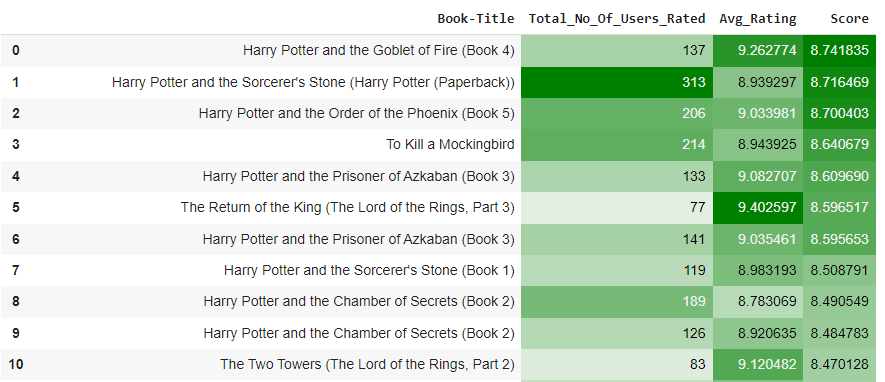
These are the top 10 books which have the highest rating and also have the highest read count at the same time.

## **b. Recommendation for New Users on the basis of Weighted Average**

As the name suggests these top recommendation systems work with the trend. It basically uses the items which are in trend right now. For example, if any book which is usually bought by every new user then there are chances that it may suggest that book to the user who just signed up.

In the above m variable we used 90th percentile as our cutoff. In other words, for a book to feature in the charts, it must have more votes than at least 90% of the books in the list.

As we saw that there are 38570 books which qualify to be in this list. Now, we need to calculate our metric for each qualified book. To do this, we will define a function, weighted\_rating() and define a new feature score, of which we’ll calculate the value by applying this function to our DataFrame of qualified books:



These are our top books on the basis of formula base-weighted ratings.

**B. Model Based Collaborative Filtering Recommender**

The goal of the recommender system is to predict user preference for a set of items based on the past experience. Two of the most popular approaches are Content-Based and Collaborative Filtering.

Collaborative filtering is a technique used by websites like Amazon, YouTube, and Netflix. It filters out items that a user might like on the basis of reactions of similar users. There are two categories of collaborative filtering algorithms: memory based and model based.

Model based approach involves building machine learning algorithms to predict user's ratings. They involve dimensionality reduction methods that reduce high dimensional matrices containing an abundant number of missing values with a much smaller matrix in lower-dimensional space.

## **SVD and NMF models comparison**

Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF) are matrix factorization techniques used for dimensionality reduction.

Surprise package provides implementation of those algorithms.

Here are the results of SVD Model:

test\_rmse 1.600470

test\_mae 1.238534

fit\_time 10.343655

test\_time 0.929151

Here are the results of NMF Model:

test\_rmse 2.618939

test\_mae 2.234966

fit\_time 11.548717

test\_time 0.656570

It's clear that for the given dataset much better results can be obtained with the SVD approach - both in terms of accuracy and training / testing time.

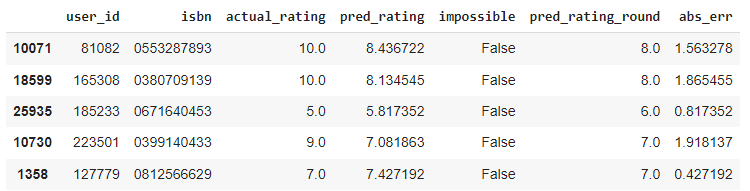
## **Optimisation of SVD algorithm**

Grid Search Cross Validation computes accuracy metrics for an algorithm on various combinations of parameters, over a cross-validation procedure. It's useful for finding the best configuration of parameters.

The improvement obtained with Grid Search is very small.

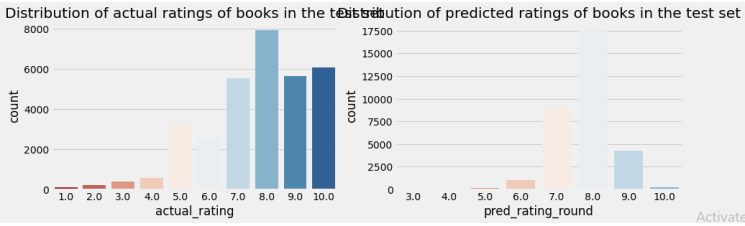
## **Analysis of Collaborative Filtering model-based results**

In this part, let's examine in detail the results obtained by the SVD model that provided the best RMSE score.



Distribution of actual and predicted ratings in the test set According to the distribution of actual ratings of books in the test set, the biggest part of users give positive scores - between 7 and 10. The mode equals 8 but count of ratings 7, 9, 10 is also noticeable. The distribution of predicted ratings in the test set is visibly different. One more time, 8 is a mode but scores 7, 9 and 10 are clearly less frequent.

It shows that the recommender system is not perfect and it cannot reflect the real distribution of book ratings.

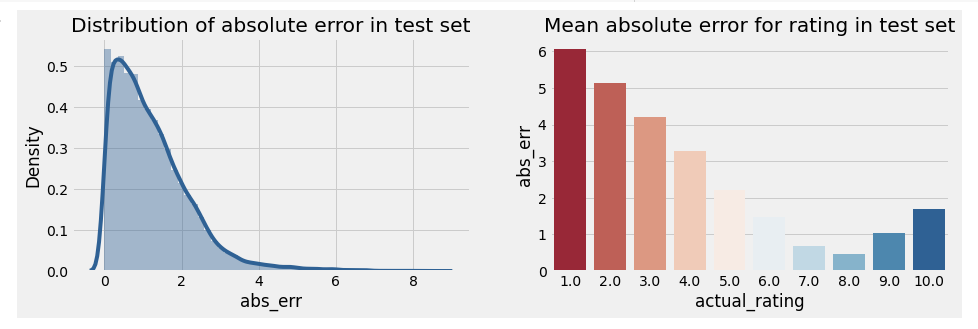


Absolute error of predicted ratings\*\*

The distribution of absolute errors is right-skewed, showing that the majority of errors is small: between 0 and 1. There is a long tail that indicates that there are several observations for which the absolute error was close to 10.

How good/bad the model is with predicting certain scores? As expected from the above

charts, the model deals very well with predicting score = 8 (the most frequent value). The further the rating from score = 8, the higher the absolute error. The biggest errors happen to observations with scores 1 or 2 which indicates that probably the model is predicting high ratings for those observations.



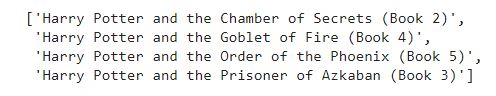
Analysis of predicted ratings of a particular user

For this part of the analysis, the user with id 193458 was selected. By analyzing book ratings by this user, it can be noted that he/she likes diverse types of readings: English romantic novels (Pride and Prejudice, Sense and Sensibility), fantasy (Narnia) as well as historical novels (Schindler's List). Among the recommended books there are other works from Narnia's series, two historical novels and one romance which correlates with the user's previous preferences.

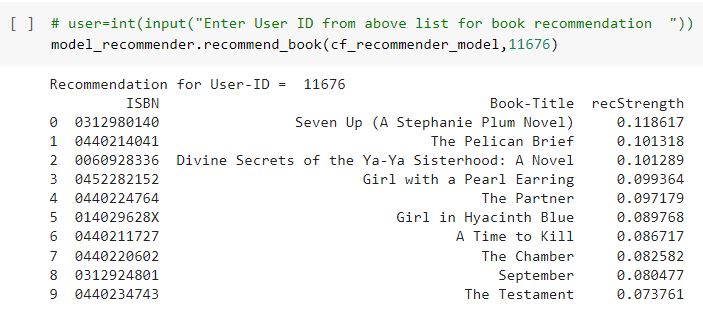
**C. Memory Based Collaborative Filtering Recommender**

**a. Item-Item based-** after using the help of Nearest Neighbours and cosine similarity we will make this system which can recommend similar books.

Let's find books similar to 'Harry Potter and the Sorcerer's Stone (Book 1)'



**b. User-Item based-** In this model we used svds, dot products as well as pivot table. which can recommend books according to the taste of similar users.



EVALUATION:

Recall@5=0.23 & Recall@10=0.30

**D. Content Based Filtering Recommender Systems**

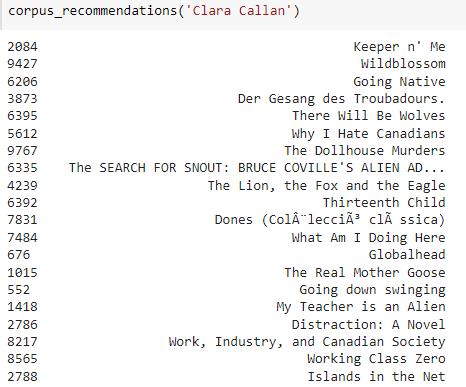
For Content Based Book Recommendation we have to use NLP techniques like Keyword extraction -> Extract keywords from title Cosine Similarity -> Find cosine similarity between all book titles.

## 

## **Content-Based Recommendation on the basis of Search Keyword(Tags, Book-name,Author-name,Publication-name)**

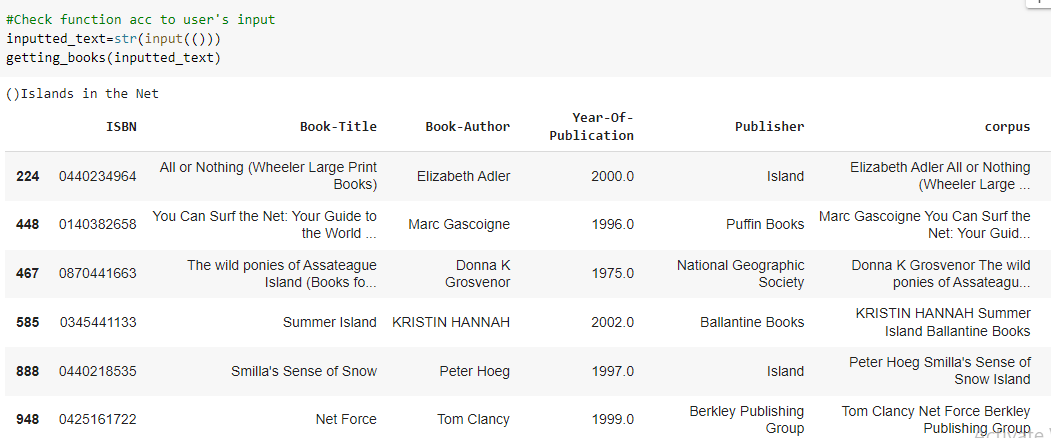
### Via Tf-idf vectorizer:

* Making a corpus column by joining Book-author, book-title and book publishers
* Applying Vectorizer
* Build a 1-dimensional array with book titles
* Function that get book recommendations based on the cosine similarity score of books tags



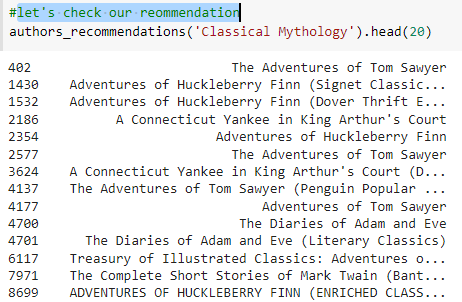
**Via Tags**

* Downloading package stopwords to /root/nltk\_data…
* Making a corpus column by joining Book-author, book-title and book publishers
* extracting the stopwords from nltk library
* Making necessary function for applying on tags column for better results
* Create an object of stemming function
* Making function for applying all function of above
* Making function for getting books according to our customer's search

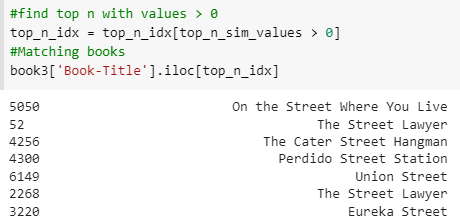


## Content-Based Recommendation on the basis of Author

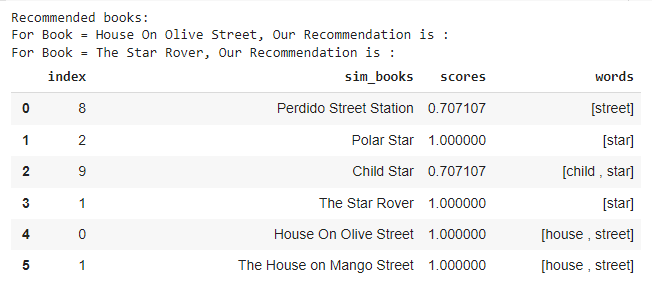
* Applying tf-idf vectorizer
* Applying linear kernal for getting similarity
* A function that returns the 20 most similar books based on the cosine similarity score.
* Build a 1-dimensional array with book titles
* let's check our recommendation



## **Content-Based Recommendation on the basis of Book-Title(with count-vectorizer)**



As we can see, all the books similar to 'Street' will be recommended by this recommendation.

**Content-Based Recommendation on the basis of Book-Purchase history list** -****

# 6. Conclusion

Alas! We have reached the end of our technical documentation.

For this project our client is an online book selling firm. They now need assistance in developing a model to recommend another books on the basis of customer purchase-history and other information which are given in the datasets.

Building a model to recommend another books is extremely beneficial to the company because it can increase their sales via recommend relevant books to their customers and optimize its business model and revenue accordingly.

* For modeling, it was observed that for **model based** collaborative filtering SVD technique worked way better than NMF with lower Mean Absolute Error (MAE) .
* Amongst the memory based approach, **item-item CF performed better** than **user-item CF** because of lower computation.
* Content-based recommendation on the basis of **Tags** are also doing good in terms of results.

## **Future Work**

* We can recommend books to our customers on basis of genres also but we have no information on that so we have to record books genres also for better recommendation.
* We can also record Date-time of our users when they buy book, By using that we can recommend our top books, authors, publication on monthly basis.

**References:**

* Stackoverflow
* GeeksforGeeks
* Kaggle
* Machinelearningmastery
* Stackexchange