ABSTRACT

Now-a-days, everyone depending on reviews by others in many things such as selecting a movie to watch, buying products, reading a book. Recommender systems are used for that purpose only. A recommender system is a kind of filtering system that predicts a user's rating of an item. Recommender systems recommend items to users by filtering through a large database of information using a ranked list of predicted ratings of items. Online Book recommender system is a recommender system for ones who love books. When selecting a book to read, individuals read and rely on the book ratings and reviews that previous users have written. In this paper, Hybrid Recommender system is used in which Collaborative Filtering and Content-Based Filtering techniques are used. The author used Collaborative techniques such as Clustering in which data-points are grouped into clusters. Algorithms such as K-means clustering and Gaussian mixture are used for clustering. The better algorithm was selected with the help of silhouette score and used for clustering. Matrix Factorization technique such as Truncated-SVD which takes sparse matrix as input is used for reducing the features of a dataset. Content Based Filtering System used TF-IDF vectorizer which took statements as input and return a matrix of vectors. RMSE (Root Mean Square Error) is used for finding the deviation of an absolute value from an obtained value and that value is used for finding the fundamental accuracy.

Keywords: Book Recommender System, Matrix Factorization, Clustering, K-Means, Gaussian Mixture, Root Mean Square Error.

CONTENTS

LIST OF SYMBOLS

Z	Normalization constant
μ	Mean
$\sigma 2$	Variance
X	Mean vector
Σ	2x2 covariance matrix

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

TSVD	Truncated SVD
TFIDF	TF IDF Vectorizer

CHAPTER 1 – INTRODUCTION

1.1 Introduction

Now-a-days, online rating and reviews are playing an important role in books sales. Readers were buying books depend on the reviews and ratings by the others. Recommender system focuses on the reviews and ratings by the others and filters books. In this paper, Hybrid recommender system is used to boost our recommendations. The technique used by recommender systems is Collaborative filtering. This technique filters information by collecting data from other users. Collaborative filtering systems apply the similarity index-based technique. The ratings of those items by the users who have rated both items determine the similarity of the items. The similarity of users is determined by the similarity of the ratings given by the users to an item. Content-based filtering uses the description of the items and gives recommendations which are similar to the description of the items. With these two filtering systems, books are recommended not only based on the user's behaviour but also with the content of the books. So, our recommendation system recommends books to the new users also. In this recommender system, books are recommended based on collaborative filtering technique and similar books are shown using content based filtering.

The required dataset for the training and testing of our model is downloaded from Good-Reads website. Matrix Factorization technique such as Truncated-SVD which takes sparse matrix of dataset is used for reduction of features. The reduced dataset is used for clustering to build a recommendation system. Clustering is a collaborative filtering technique that is used to build our recommendation system in which data points are grouped into clusters. . In this paper, we used two methods i.e., K-means and Gaussian mixture for clustering the users. The better model is selected based on the silhouette score and used for clustering. Silhouette score or silhouette coefficient is used to calculate how good the clustering is done. Negative value shows that clustering is imperfect whereas positive value shows that clustering was done perfectly. Difference between the mean rating before clustering and after clustering is calculated. Root Mean square Error is used to measure the error between the absolute

values and obtained values. That RMSE value is used to find the fundamental accuracy.

1.1.1 Introduction to Machine Learning

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. It gives the computer that makes it more similar to humans i.e. ability to learn. Machine learning is used in many streams than anyone would accept. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide**.** The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. Machine Learning is a sub-area of artificial intelligence, whereby the term refers to the ability of IT systems to independently find solutions to problems by recognizing patterns in databases. In other words: Machine Learning enables IT systems to recognize patterns on the basis of existing algorithms and data sets and to develop adequate solution concepts. Therefore, in Machine Learning, artificial knowledge is generated on the basis of experience.

In order to enable the software to independently generate solutions, the prior action of people is necessary. For example, the required algorithms and data must be fed into the systems in advance and the respective analysis rules for the recognition of patterns in the data stock must be defined. Once these two steps have been completed, the system can perform the following tasks by Machine Learning:

- Finding, extracting and summarizing relevant data
- Making predictions based on the analysis data
- Calculating probabilities for specific results

Basically, algorithms play an important role in Machine Learning: On the one hand, they are responsible for recognizing patterns and on the other hand, they can generate solutions. Algorithms can be divided into different categories:

1.1.1.1 Supervised learning:

In the course of monitored learning, example models are defined in advance. In order to ensure an adequate allocation of the information to the respective model groups of the algorithms, these then have to be specified. In other words, the system learns on the basis of given input and output pairs. In the course of monitored learning, a programmer, who acts as a kind of teacher, provides the appropriate values for a particular input. The aim is to train the system in the context of successive calculations with different inputs and outputs to establish connections.

Supervised learning is where you have input variables (X) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output. Y = f(X) The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (Y) for that data. It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Techniques of Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, Decision Tree and Support Vector Machine.

Supervised Learning problems can be further grouped into Regression and Classification problems. The difference between these two is that the dependent attribute is numerical for regression and categorical for classification:

- **Regression:**

  Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

  When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

- **Classification:**

Classification is a process of categorizing a given set of data into classes, It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.

In short classification either predicts categorical class labels or classification data based on the training set and the values(class labels) in classifying attributes and uses it in classifying new data.

There are number of classification models. Classification models include Logistic Regression, Decision Tree, Random Forest, Gradient Boosted Tree, One-vs.-One and Naïve Bayes.

1.1.1.2 Unsupervised learning:

In unsupervised learning, artificial intelligence learns without predefined target values and without rewards. It is mainly used for learning segmentation (clustering). The machine tries to structure and sort the data entered according to certain characteristics. For example, a machine could (very simply) learn that coins of different colors can be sorted according to the characteristic "color" in order to structure them. Unsupervised Machine Learning algorithms are used when the information used to train is neither classified nor labeled. The system does not figure out the right output but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data. Unsupervised Learning is the training of Machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.

Unsupervised Learning is classified into two categories of algorithms:

- **Clustering:** A clustering problem is where you want to discover the inherent grouping in the data such as grouping customers by purchasing behavior.

- **Association:** An Association rule learning problem is where you want to discover rules that describe large portions of your data such as people that buy X also tend to buy Y.

1.1.1.3 Applications of Machine Learning:

Virtual Personal Assistants:

Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice. Machine learning is an important part of these personal assistants as they collect and refine the information on the basis of your previous involvement with them. Later, this set of data is utilized to render results that are tailored to your preferences.

Virtual Assistants are integrated to a variety of platforms. For example:

- Smart Speakers: Amazon Echo and Google Home
- Smartphone: Samsung Bixby on Samsung S8
- Mobile Apps: Google Allo

Videos Surveillance:

- Imagine a single person monitoring multiple video cameras! Certainly, a difficult job to do and boring as well. This is why the idea of training computers to do this job makes sense.

- The video surveillance system, nowadays are powered by AI that makes it possible to detect crime before they happen. They track unusual behaviour of people like standing motionless for a long time, stumbling, or napping on benches etc. The system can thus give an alert to human attendants, which can ultimately help to avoid mishaps. And when such activities are reported and counted to be true, they help to improve the surveillance services. This happens with machine learning doing its job at the backend.

Social Media Services:

From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits.

- People You May Know

- Face Recognition

Search Engine Result Refining:

Google and other search engines use machine learning to improve the search results for you. Every time you execute a search, the algorithms at the backend keep a watch at how you respond to the results. If you open the top results and stay on the web page for long, the search engine assumes that the results it displayed were in accordance to the query. Similarly, if you reach the second or third page of the search results but do not open any of the results, the search engine estimates that the results served did not match requirement. This way, the algorithms working at the backend improve the search results.

1.1.2 Clustering

Clustering is an unsupervised learning method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them. Clustering is very important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for good clustering. It depends on the user, what is the criteria they may use which satisfy their need. This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters.

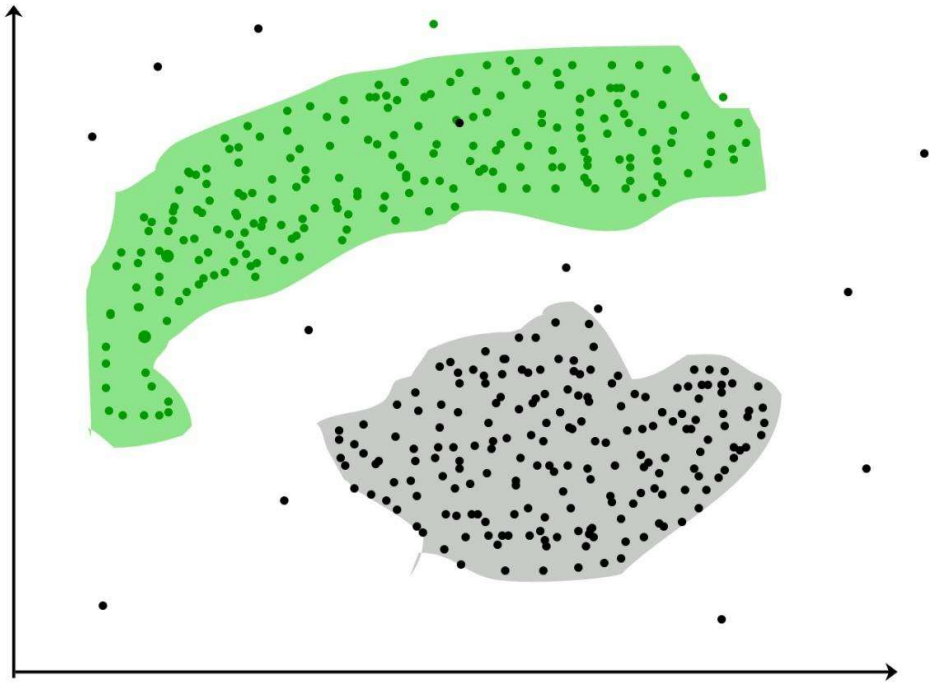


Fig 1.1 Clustering of data points

Clustering Methods:

- **Density-Based Methods:** These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space. These methods have good accuracy and ability to merge two clusters.
- Example: DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify Clustering Structure) etc.
- **Hierarchical Based Methods:** The clusters formed in this method forms a tree-type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two categories:
    - **Agglomerative** (*bottom up approach*)
    - **Divisive** (*top down approach*)
- Examples: CURE (Clustering Using Representatives), BIRCH (Balanced Iterative Reducing Clustering and using Hierarchies) etc.
- **Partitioning Methods:** These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter

example K-means, CLARANS (Clustering Large Applications based upon Randomized Search) etc.

- **Grid-based Methods:** In this method the data space is formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects example STING (Statistical Information Grid), wave cluster, CLIQUE (Clustering In Quest) etc.

In this paper, partitioning method of clustering is used. We used Clustering algorithm which is simplest unsupervised learning algorithm in this paper and it partition n observations into k clusters where each observation belongs to the cluster.

1.1.3 Matrix Factorization

Matrix factorization is a way to generate latent features when multiplying two different kinds of entities. Collaborative filtering is the application of matrix factorization to identify the relationship between items and users entities. With the input of users' ratings on the shop items, we would like to predict how the users would rate the items so the users can get the recommendation based on the prediction. Matrix Factorization is a technique to discover the latent factors from the ratings matrix and to map the items and the users against those factors. Consider a ratings matrix R with ratings by n users for m items. The ratings matrix R will have n × m rows and columns.

Matrix Factorization is a significant approach in many applications. Curse of dimensionality is a phenomenon which occurs in high dimensional space that hardly occur in lower dimensional space. Due to higher number of dimension model gets sparse. Higher dimensional space causes problem in clustering (becomes very difficult to separate one cluster data from another), search space increases, complexity of model increases. We can reduce the dimension by following two ways:

- **Feature selection:** Selecting important features which are relevant to model (it avoids the curse of dimensionality)
- **Feature extraction**: Transformation of high dimensional space into lower dimensional space by using various methods such as PCA, TSVD, T- SNE etc.

In this paper, we used Feature extraction for reducing the features and used method i.e. Truncated SVD for dimensionality reduction.

1.1.4 Silhouette Score

Silhouette score or silhouette coefficient is used to evaluate the quality of clusters created using clustering algorithms such as K-Means in terms of how well samples are clustered with other samples that are similar to each other. It refers to a method of interpretation and validation of consistency within clusters of data. Silhouette Score is a metric used to calculate the goodness of a clustering technique. The silhouette can be calculated with any distance metric, such as the Euclidean distance or the Manhattan distance.

$$\text{Silhouette Score} = (b-a)/\max(a, b)$$

Where,

a = average intra-cluster distance i.e. the average distance between each point within a cluster.

b = average inter-cluster distance i.e. the average distance between all clusters.

This value ranges from -1 to 1. Positive value indicates that mean clusters are well apart from each other and clearly distinguished so we require $a << b$. 0 indicates that mean clusters are indifferent, or we can say that the distance between clusters is not significant. Negative value indicates that mean clusters are assigned in the wrong way. We can also increase the likelihood of the silhouette being maximized at the correct number of clusters by re-scaling the data using feature weights that are cluster specific.

1.2 Motivation of the work

In the present world, all products are buying based on the reviews and ratings by the others. There are so many products with high rating and reviews but we only put our interest in some products which we like. Recommender system works on this principle only i.e. it recommends products based on the interest of the users. Our idea is to create recommender system that recommends books based on the user's interest i.e. we recommends books which are similar to the books that user already liked. It can also recommend books which are liked by similar users. Similar users are those who liked the books which are liked by the current user.

We also add another feature i.e. we recommends books which are independent of the users interest. With this feature, we can recommend books to the new users also. Book recommendation sites that were available online now a days shows the books which are recommended by the system. Here, we are also recommending books based on the description of the book. We will get books which are similar to the book we selected in this system. For that purpose only, we built Hybrid recommender system. Hybrid recommender system is a combination of Collaborative Filtering system and Content Based Filtering system.

1.3 Problem Statement

Recommending books using Machine learning algorithm is the main goal of this project. Books are recommended by the clustering model and we are going to train and build using various features such as user's rating, book description, book titles etc. The system groups users into clusters so that each data point within cluster is similar and dissimilar to the data point in the other cluster. The system we would like to develop will also be able to find an average rating for each cluster and it is going to find top rated books of users from each cluster. All these books shortlisted by our system will be used for training our model in future. The prediction model needs to be trained so as to produce better results.

1.4 Organization of Thesis

The chapters of this document describe the following:

Chapter-1 is about the introduction of our project where we have given clear insights about our project domain and other related concepts.

Chapter-2 specifies about literature survey where all different existing methods and models are examined.

Chapter-3 specifies about proposed system with a system architecture along with detailed explanations of each module.

Chapter-4 specifies about the experimental analysis of our system along with performance measures and comparisons between different models. It also specifies about implementation along with sample code.

Chapter-5 gives the conclusion to our work with an insight for the future scope.

CHAPTER 2 – LITERATURE SURVEY

Most researchers used Pearson's Correlation Coefficient function to calculate similarity among book ratings to recommend books.

2.1 Collaborative Filtering with Jaccard Similarity to build a recommendation system

Avi Rana and K. Deeba, et.al. (2019) [1] proposed a paper "Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)". In this paper, the author used CF with Jaccard similarity to get more accurate recommendations because general CF difficulties are scalability, sparsity, and cold start. So to overcome these difficulties, they used CF with Jaccard Similarity. JS is based on pair of books index which is a ratio of common users who have rated both books divided by the sum of users who have rated books individually. Books with a high JS index are highly recommended.

2.2 Building a Recommendation System using Keras Deep learning Framework

G. Naveen Kishore, et.al. (2019) [2] proposed a paper "Online Book Recommendation System". The dataset used in this paper was taken from the website "good books-10k dataset" which contains ten thousand unique books. Features are book_id, user_id, and rating. In this paper, the author adopted a Keras deep learning framework model to create neural network embedding.

2.3 Using Quick sort Algorithm approach to design a system

Uko E Okon, et.al. (2018) [3] proposed a paper "An Improved Online Book Recommender System using Collaborative Filtering Algorithm". The authors designed and developed a recommendation model by using a quick sort algorithm,

collaborative filtering, and object-oriented analysis and design methodology (OOADM). This system produces an accuracy of 90-95%.

2.4 Using UV Decomposition and KNN for building system

Jinny Cho, et.al. (2016) [4] proposed a paper "Book Recommendation System". In this paper, the author uses two approach methods which are Content-based (CB) and Collaborative Filtering (CF). They used two algorithms as UV-Decomposition and K Nearest Neighbors (KNN). They obtained a result with an accuracy of 85%.

2.5 Recommending books through CB and CF approaches

Sushma Rjpurkar, et.al. (2015) [5] proposed a paper "Book Recommendation System". In this paper, the author used Associative Rule Mining to find association and correlation relationships among a dataset of items. They used CB and CF approaches to build a system.

2.6 Detecting patterns, correlations and uses Collaborative Filtering and Associative Rule Mining

Abhay E. Patil, et.al. (2019) [6] proposed a paper "Online Book Recommendation System using Association Rule Mining and Collaborative Filtering". The author detected recurrently occurring patterns, correlations and uses various databases such as relational databases, transactional databases to form associations. They used two approaches i.e., User-based and Item-based Collaborative Filtering, and used the Pearson correlation coefficient to find similarity between the items.

2.7 Uses Demographic, Collaborative Filtering, Content-based to build a Hybrid Recommender System

Suhas Patil, et.al. (2016) [7] proposed a paper "A Proposed Hybrid Book Recommender System". In this paper, the author used techniques such as

Demographic, Collaborative Filtering, Content-based to build a system and rarely they combined the features of these techniques to make a better recommendation system.

2.8 PHP-based CF, Fuggy logic , Context Engine for recommendation systems

Ankit Khera, et.al. (2008) [8] proposed a paper "Online Recommendation System". In this paper, the author used the User similarity matrix, Vogoo which is PHP-based CF, Fuggy logic, Context Engine for building recommendation systems. Pearson Correlation is a similarity function in this paper.

2.9 Hybrid Recommender System through Collaborative Filtering

Anagha Vaidya and Dr. Subhash Shinde, et.al. (2019) [9] proposed a paper "Hybrid Book Recommendation System". In this paper, the author used techniques such as Collaborative Filtering etc. and used the Pearson correlation coefficient. It was published in International Research Journal of Engineering and Technology (IRJET).

2.10 Using Machine Learning Algorithm to build a system

Dhirman Sarma,Tanni Mittra and Mohammad Shahadat Hossain, et.al. (2019) [10] proposed a paper "Personalized Book Recommendation System using Machine Learning Algorithm". It was published in The Science and Information Organization vol.12.

CHAPTER 3 – METHODOLOGY

3.1 Proposed system

3.1.1 System Architecture

System Architecture describes "the overall structure of the system and the ways in which the structure provides conceptual integrity". The system architecture to build a recommendation system involves the following five major steps.

3.2.1 Data Acquisition
3.2.2 Data Pre-processing
3.2.3 Feature Extraction
3.2.4 Training Methods
3.2.5 Testing Data

In Step 3.2.1, Dataset was collected from Good Reads Website in which three datasets are present i.e. Books Dataset, Ratings Dataset, Users Dataset. In Step 3.2.2, Datasets were pre-processed to make suitable for developing the Recommendation system. In Step 3.2.3, Feature extraction is performed in which Truncated-SVD is used to reduce the features of the dataset and Data splitting is done in which training dataset and testing dataset are divided into 80:20 ratio. In Step 3.2.4, Content Based Filtering System is developed in which book description is taken as an input and Collaborative Filtering System is developed by building a model using K-Means Algorithm over Gaussian Mixture after comparing with Silhouette scores. In step 3.2.5, Testing of model with test data is performed.

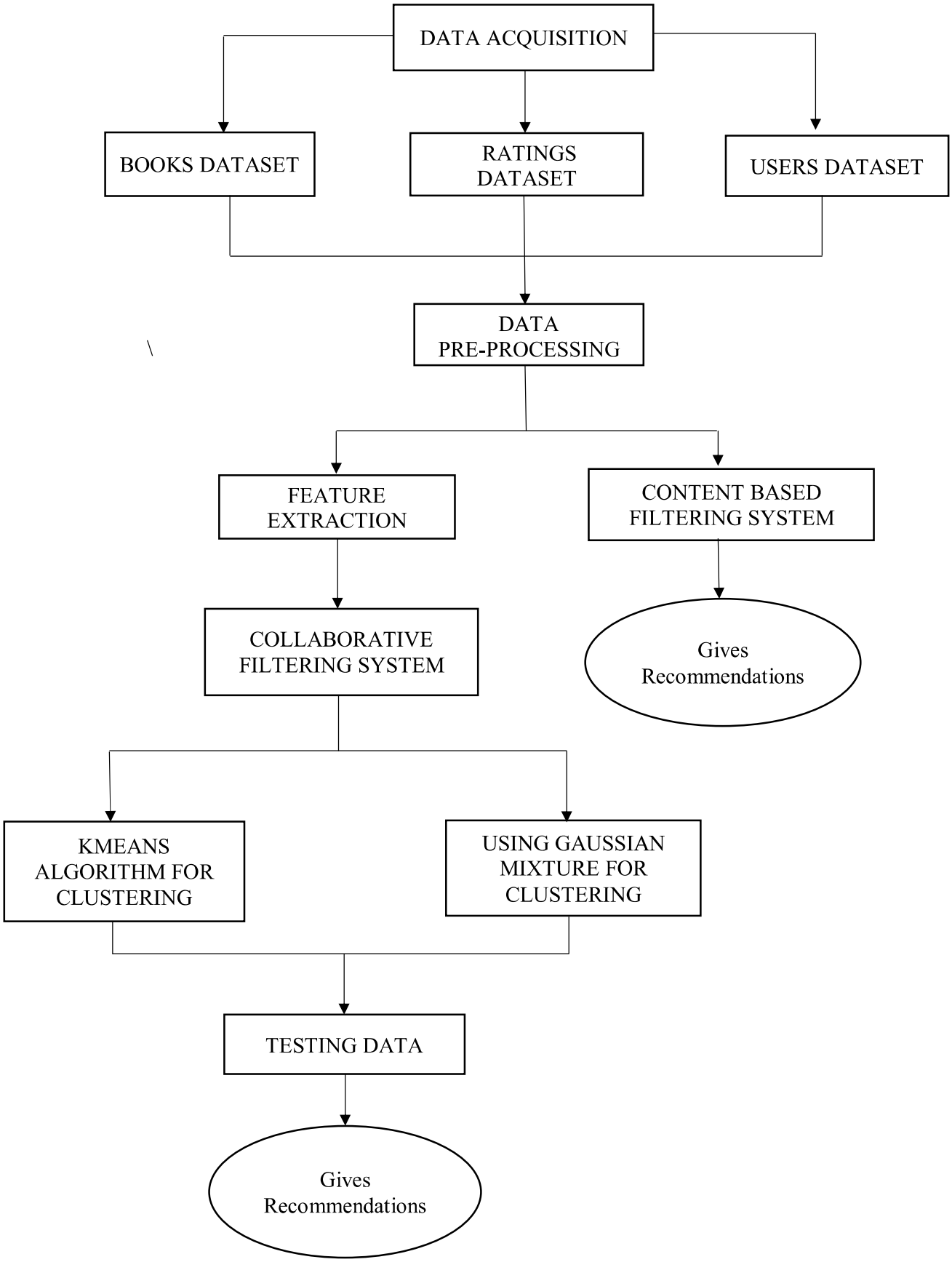


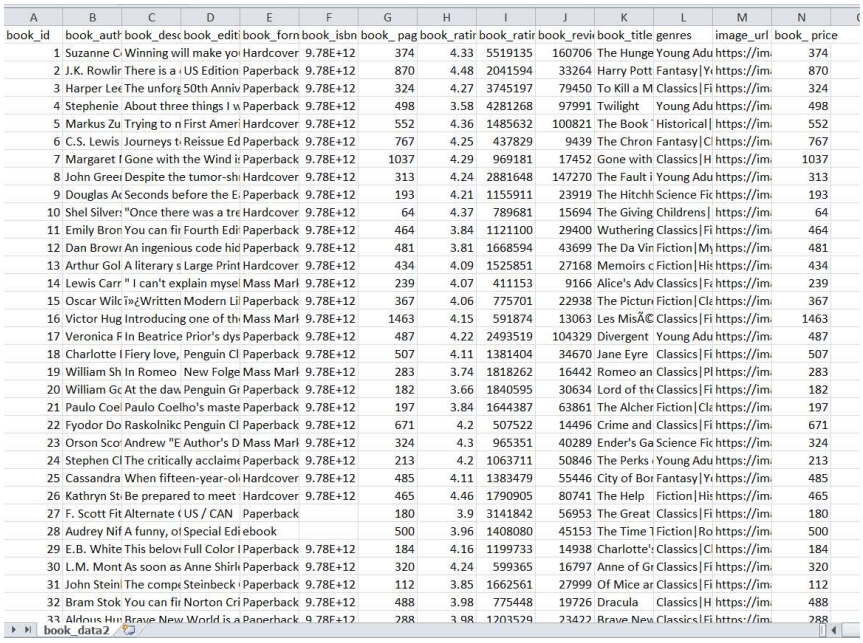
Fig-3.1 System Architecture

3.2 Modules Division

Let us discuss about the various modules in our proposed system and what each module contributes in achieving our goal.

3.2.1 Data Acquisition:

The goal of this step is to find and acquire all the related datasets or data sources. In this step, the main aim is to identify various available data sources, as data are often collected from various online sources like databases and files. The size and the quality of the data in the collected dataset will determine the efficiency of the model. The Books dataset is collected from the Goodreads website.



book_id	book_auth	book_desc	book_editi	book_form	book_isbn	book_pag	book_ratir	book_ratir	book_revi	book_title	genres	image_url	book_price
1	Suzanne C	Winning will make yo		Hardcover	9.78E+12	374	4.33	5519135	160706	The Hunge	Young Adu	https://im	374
2	J.K. Rowlir	There is a	US Edition	Paperback	9.78E+12	870	4.48	2041594	33264	Harry Pott	Fantasy|Y	https://im	870
3	Harper Lee	The unforg	50th Anniv	Paperback	9.78E+12	324	4.27	3745197	79450	To Kill a M	Classics|Fi	https://im	324
4	Stephenie	About three things I w		Paperback	9.78E+12	498	3.58	4281268	97991	Twilight	Young Adu	https://im	498
5	Markus Zu	Trying to n	First Amer	Hardcover	9.78E+12	552	4.36	1485632	100821	The Book	Historical|	https://im	552
6	C.S. Lewis	Journeys t	Reissue Ed	Paperback	9.78E+12	767	4.25	437829	9439	The Chron	Fantasy|Cl	https://im	767
7	Margaret I	Gone with the Wind is		Paperback	9.78E+12	1037	4.29	969181	17452	Gone with	Classics|H	https://im	1037
8	John Gree	Despite the tumor-shi		Hardcover	9.78E+12	313	4.24	2881648	147270	The Fault i	Young Adu	https://im	313
9	Douglas A	Seconds before the E		Paperback	9.78E+12	193	4.21	1155911	23919	The Hitchh	Science Fic	https://im	193
10	Shel Silver	"Once there was a tre		Hardcover	9.78E+12	64	4.37	789681	15694	The Giving	Childrens|	https://im	64
11	Emily Bron	You can fir	Fourth Edi	Paperback	9.78E+12	464	3.84	1121100	29400	Wuthering	Classics|Fi	https://im	464
12	Dan Brow	An ingenious code hid		Paperback	9.78E+12	481	3.81	1668594	43699	The Da Vin	Fiction|My	https://im	481
13	Arthur Gol	A literary s	Large Print	Hardcover	9.78E+12	434	4.09	1525851	27168	Memoirs c	Fiction|His	https://im	434
14	Lewis Carr	" I can't explain myse		Mass Marl	9.78E+12	239	4.07	411153	9166	Alice's Adv	Classics|Fa	https://im	239
15	Oscar Wilc	ï»¿Written	Modern Li	Paperback	9.78E+12	367	4.06	775701	22938	The Pictur	Fiction|Cla	https://im	367
16	Victor Hug	Introducing one of th		Mass Marl	9.78E+12	1463	4.15	591874	13063	Les MisÃ€	Classics|Fi	https://im	1463
17	Veronica F	In Beatrice Prior's dys		Paperback	9.78E+12	487	4.22	2493519	104329	Divergent	Young Adu	https://im	487
18	Charlotte I	Fiery love,	Penguin Cl	Paperback	9.78E+12	507	4.11	1381404	34670	Jane Eyre	Classics|Fi	https://im	507
19	William Sh	In Romeo	New Folge	Mass Marl	9.78E+12	283	3.74	1818262	16442	Romeo an	Classics|Pl	https://im	283
20	William Gc	At the daw	Penguin Gr	Paperback	9.78E+12	182	3.66	1840595	30634	Lord of the	Classics|Fi	https://im	182
21	Paulo Coe	Paulo Coelho's maste		Paperback	9.78E+12	197	3.84	1644387	63861	The Alcher	Fiction|Cla	https://im	197
22	Fyodor Do	Raskolnikc	Penguin Cl	Paperback	9.78E+12	671	4.2	507522	14496	Crime and	Classics|Fi	https://im	671
23	Orson Sco	Andrew "E	Author's D	Mass Marl	9.78E+12	324	4.3	965351	40289	Ender's Ga	Science Fic	https://im	324
24	Stephen Cl	The critically acclaime		Paperback	9.78E+12	213	4.2	1063711	50846	The Perks	Young Adu	https://im	213
25	Cassandra	When fifteen-year-ol		Hardcover	9.78E+12	485	4.11	1383479	55446	City of Bor	Fantasy|Y	https://im	485
26	Kathryn St	Be prepared to meet		Hardcover	9.78E+12	465	4.46	1790905	80741	The Help	Fiction|His	https://im	465
27	F. Scott Fit	Alternate (	US / CAN	Paperback		180	3.9	3141842	56953	The Great	Classics|Fi	https://im	180
28	Audrey Nif	A funny, ol	Special Edi	ebook		500	3.96	1408080	45153	The Time T	Fiction|Ro	https://im	500
29	E.B. White	This belov	Full Color I	Paperback	9.78E+12	184	4.16	1199733	14938	Charlotte's	Classics|Cl	https://im	184
30	L.M. Mont	As soon as	Anne Shirl	Paperback	9.78E+12	320	4.24	599365	16797	Anne of Gr	Classics|Fi	https://im	320
31	John Stein	The compe	Steinbeck	Paperback	9.78E+12	112	3.85	1662561	27999	Of Mice ar	Classics|Fi	https://im	112
32	Bram Stok	You can fir	Norton Cri	Paperback	9.78E+12	488	3.98	775448	19726	Dracula	Classics|H	https://im	488
33	Aldous Hu	Brave New World is a		Paperback	9.78E+12	288	3.98	1203529	23422	Brave New	Classics|Fi	https://im	288

book_data2

Fig-3.2 Sample of acquired books dataset from Good reads Website

In the above Fig-3.2, we can see a sample of the dataset we have collected. This acquired dataset has around 3,000 books and has 14 different features. The features are listed below:

- book_id
- book_authors
- book_desc
- book_edition
- book_format

- book_isbn
- book_pages
- book_rating
- book_rating_count
- book_review_count
- book_title
- genres
- image_url
- book_price

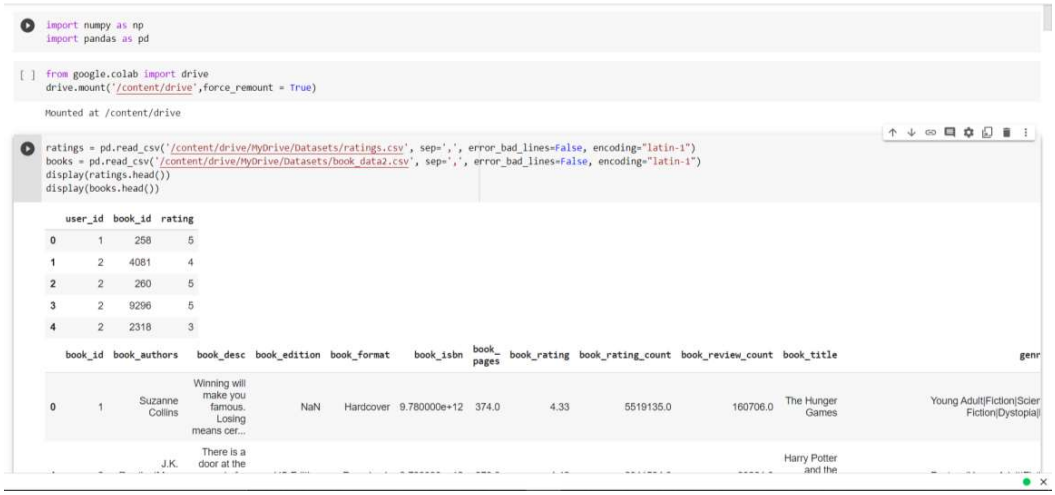
One more dataset i.e. ratings dataset was also collected from Goodreads website.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	user_id	book_id	rating										
2	1	258	5										
3	2	4081	4										
4	2	260	5										
5	2	9296	5										
6	2	2318	3										
7	2	26	4										
8	2	315	3										
9	2	33	4										
10	2	301	5										
11	2	2686	5										
12	2	3753	5										
13	2	8519	5										
14	4	70	4										
15	4	264	3										
16	4	388	4										
17	4	18	5										
18	4	27	5										
19	4	21	5										
20	4	2	5										
21	4	23	5										
22	4	24	5										
23	4	964	4										
24	4	103	5										
25	4	255	2										
26	4	35	5										
27	4	287	3										
28	4	337	4										
29	4	26	3										
30	4	84	4										
31	4	58	4										
32	4	1117	3										

Fig-3.3 Sample of acquired ratings dataset from Good reads Website

In the above Fig-3.3, we can see a sample of the dataset we have collected. This acquired dataset has around 400000 ratings and has 3 different features.

- user_id
- book_id
- rating



```
import numpy as np
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive',force_remount = True)
```

```
Mounted at /content/drive
```

```
ratings = pd.read_csv('/content/drive/MyDrive/Datasets/ratings.csv', sep=',', error_bad_lines=False, encoding="latin-1")
books = pd.read_csv('/content/drive/MyDrive/Datasets/book_data2.csv', sep=',', error_bad_lines=False, encoding="latin-1")
display(ratings.head())
display(books.head())
```

	user_id	book_id	rating
0	1	258	5
1	2	4081	4
2	2	260	5
3	2	9296	5
4	2	2318	3

	book_id	book_authors	book_desc	book_edition	book_format	book_isbn	book_pages	book_rating	book_rating_count	book_review_count	book_title	genr
0	1	Suzanne Collins	Winning will make you famous. Losing means cer...	NaN	Hardcover	9.780000e+12	374.0	4.33	5519135.0	160706.0	The Hunger Games	Young Adult|Fiction|Scier Fiction|Dystopia|
		J.K.	There is a door at the								Harry Potter and the	

Fig-3.4 Reading the dataset from CSV file into python notebook

After acquiring the data our next step is to read the data from the csv file into python notebook. Python notebook is used in our project for data pre-processing, features selection and for model comparison. In the fig-3.4, we have read data from csv file using the inbuilt python functions that are part of pandas library.

3.2.2 Data Pre-Processing:

The goal of this step is to study and understand the nature of data that was acquired in the previous step and also to know the quality of data. In this step, we will check for any null values and remove them as they may affect the efficiency. Identifying duplicates in the dataset and removing them is also done in this step.

```
# Merge the two tables then pivot so we have Users X Books dataframe.
ratings_title = pd.merge(ratings, books[['book_id', 'book_title']], on='book_id' )
user_book_ratings = pd.pivot_table(ratings_title, index='user_id', columns= 'book_title', values='rating')

print('dataset dimensions: ', user_book_ratings.shape, '\n\nSubset example:')
user_book_ratings.iloc[:25, :10]
```

```
dataset dimensions:  (5850, 2080)

Subset example:
```

book_title	'Salem's Lot	'Tis A Memoir	09-Nov	11/22/1963	1984	1Q84	2001: A Space Odyssey	2666	84, Charing Cross Road	A Bend in the Road
user_id										
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN

Fig-3.5 Merging the tables and then pivot so that we get Users X Books Dataframe

In the above fig-3.5, we combine datasets by using merge function using ‘on’ keyword and pivot tables so that we get Users X Books data frame.

```
[6] # Drop users that have given fewer than 100 ratings of these most-rated books
    user_book_ratings = user_book_ratings.dropna(thresh=100)
    print('dataset dimensions: ', user_book_ratings.shape, '\n\nSubset example:')
    user_book_ratings.iloc[:25, :10]

dataset dimensions:  (528, 2080)

Subset example:
```

book_title	'Salem's Lot	'Tis A Memoir	09-Nov	11/22/1963	1984	1Q84	2001: A Space Odyssey	2666	84, Charing Cross Road	A Bend in the Road
user_id										
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
35	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN
79	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN
85	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN
158	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
202	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
207	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN
209	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
215	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
226	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
228	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN
247	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
253	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Fig-3.6 Dropping the users who have given fewer than 100 ratings

In the above fig-3.6, we drop the users who have given fewer than 100 ratings of the Dataframe.

```
[29] user_book_ratings_without_nan = user_book_ratings.fillna(0)
     user_book_ratings_without_nan.iloc[:25,:10]
```

book_title	'Salem's Lot	'Tis A Memoir	09-Nov	11/22/1963	1984	1Q84	2001: A Space Odyssey	2666	84, Charing Cross Road	A Bend in the Road
user_id										
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0
79	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0
85	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0
158	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
202	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
207	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0
209	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
215	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
226	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
228	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0
247	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
253	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
338	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0
345	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
366	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig-3.7 Dropping the null values and filling with zeroes

In the above fig-3.7, we now drop the null values from the dataset and then fill with zero values so that we get sparse matrix which is used to fit into Truncated SVD model.

```
[5]  # removing the stop words
     books_tfidf = TfidfVectorizer(stop_words='english')
     # replace NaN with empty strings
     book_description['book_desc'] = book_description['book_desc'].fillna('')
     # computing TF-IDF matrix required for calculating cosine similarity
     book_description_matrix = books_tfidf.fit_transform(book_description['book_desc'])
```

Fig-3.8 Dropping the null values and replacing with empty string

In the above fig-3.8, we now drop the null values from the dataset and then replaced with empty strings so that we can use it to fit for TF-IDF Vectorizer model.

3.2.3 Feature Extraction:

After pre-processing the acquired data, the next step is to reduce the features i.e. Dimensionality reduction. The reduced features should be able to give high efficiency. We used Matrix Factorization technique such as Truncated SVD which takes sparse matrix as input for reduction of features.

```
[7] from sklearn.decomposition import TruncatedSVD

    # replace NaN's with zeroes for Truncated SVD
    user_book_ratings_without_nan = user_book_ratings.fillna(0)

    tsvd = TruncatedSVD(n_components=200, random_state=42)
    user_book_ratings_tsvd = tsvd.fit(user_book_ratings_without_nan).transform(user_book_ratings_without_nan)
    print('Original number of features:', user_book_ratings_without_nan.shape[1])
    print('Reduced number of features:', user_book_ratings_tsvd.shape[1])
    print('Explained variance ratio:', tsvd.explained_variance_ratio_[0:200].sum())

    Original number of features: 2080
    Reduced number of features: 200
    Explained variance ratio: 0.7848664915980508
```

Fig-3.9 Reduction of features using Truncated SVD

In scikit-learn python library, sklearn.decomposition.TruncatedSVD module is used for carrying out Truncated SVD. In the fig 3.9 we can see in the screenshot that we are reducing the dimension so that complexity of the model decreases.

Splitting the Dataset into the Training set and Test set:

In machine learning projects, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose, if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

```
[9] from sklearn.model_selection import train_test_split
    book_ratings_training, book_ratings_testing = train_test_split(book_ratings_for_clustering, test_size=0.20, random_state=42)

    print('Training data shape: ', book_ratings_training.shape)
    print('Testing data shape: ', book_ratings_testing.shape)
    book_ratings_testing.head()

    Training data shape:  (422, 200)
    Testing data shape:  (106, 200)
```

Fig-3.10 Splitting Dataset into Training Set and Testing Set

Usually, dataset will be split into train and test in the ratio of 8:2 i.e., 80 percent of data is used for training and 20 percent of data is used for testing the model. We have also done in the same way. It can be seen in the above Fig 3.10.

3.2.4 Training Methods:

Now, we have our training and testing data. The next step is to identify the possible training methods and train our models. We have used two different clustering methods for training models. After that based on the silhouette score of each model, we would decide on which model to use finally.

3.2.4.1 K-Means Clustering:

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters in such a manner that each dataset belongs to only one group that has similar features. Here K defines the number of pre-defined clusters. We have to associate each cluster with a centroid in this algorithm. The sum of distances between the data point and their corresponding clusters should be minimized. The unlabeled dataset is taken as an input and the dataset into k-number of clusters is divided, and the process is repeated until it does not find the best clusters. We have to predetermine the k value in this algorithm. Elbow method is used to find the value of k which decides the number of clusters. This method uses the Within Cluster Sum of Squares (WCSS) value that defines the total variations within a cluster.

The Formula for calculating the value of WCSS for n clusters is as follows:

$$WCSS = \sum_{Pi \text{ in Cluster1distance}} (P_i C_1)^2 + \sum_{Pi \text{ in Cluster2distance}} (P_i C_2)^2 + \ldots\ldots + \sum_{Pi \text{ in Cluster n distance}} (P_i C_n)^2$$

The basic steps involved in K-Means Clustering algorithm is as follows:

Step-1: Select the number K which gives the number of clusters.

Step-2: Select random K number of points or centroids.

Step-3: Each data point to their nearest centroid should be assigned, which forms the predefined K clusters.

Step-4: Calculate the variance and place a new centroid for each cluster.

Step-5: We have to repeat the step-3, each data-point to the new closest centroid of each cluster should be reassigned.

Step-6: If reassignment happens, then go to step-4 or else go to step-7.

Step-7: Stop.

In scikit-learn python library, sklearn.cluster.KMeans module is used for carrying out K-Means Clustering. We have to specify the number of clusters a parameter for this function. We will use our training dataset to fit the model. Fig 3.11 shows the sample code for training model using KMeans.

Fig-3.11 K-Means Clustering Model

3.2.4.2 Gaussian Mixture:

Gaussian Mixture models are powerful clustering algorithms. It assumes that there are a certain number of Gaussian distributions where each distribution represents a cluster. This model groups the data points together into a single distribution. These models used the soft clustering technique for assigning data points to Gaussian distributions.

In a one dimensional space, the probability density function of a Gaussian distribution (univariate) is as follows:

$P(x \mid \mu, \sigma 2) = N(x; \mu, \sigma 2) = 1/Z\ [\exp(-(x-\mu)\ 2/\ 2\sigma 2\ )]$

Where,

Z is the normalization constant i.e., $Z = \sqrt{}\ 2\pi\sigma 2$,

μ is the mean i.e., $\mu = E[x]$, and

$\sigma 2$ is the variance of the distribution i.e., $\sigma 2 = var[x]$.

In a multidimensional space, the probability density function of a Gaussian distribution (multivariate) is as follows:

$P(x \mid \mu, \Sigma) = N(x; \mu, \Sigma) = 1/Z\ [\exp(-1/2\ (x-\mu)T\Sigma-1(x-\mu))]$

Where,

X is the input vector,

μ is the 2D mean vector, and

Σ is the 2×2 covariance matrix

Thus, we would have K (number of clusters) Gaussian distributions.

Fig-3.12 Gaussian Mixture Model

In scikit-learn python library, sklearn.mixture.GaussianMixture module is used for Gaussian Mixture. Fig 3.12 shows the sample code for training model using Gaussian Mixture.

3.2.5 Testing Data

Once Clustering model has been trained on pre-processed dataset, then the model is tested using different data points. In this testing step, the model is checked for the silhouette score for checking goodness of clustering. All the training methods need to be verified for finding out the best model to be used. In figures 3.10, 3.11, after fitting our model with training data, we used this model to predict values for test dataset. These predicted values on testing data are used for models comparison. The users in the test set, on average, rated their clusters' favorite books higher than a random set of 10 books by 0.47 stars, or nearly half a star.

3.3 Content Based Filtering

A Content-Based filtering system recommends items that are similar to the content of the item. This System uses the description of the items and gives the recommendations that are similar to the description. We used Cosine similarity as a similarity function for this system. The Item-Content Matrix which describes the attributes of the features is taken as an input. Based on the angle between the vectors, Cosine similarity is calculated. We improve the quality of the content-based system by normalizing and tuning the attributes with the use of the TF-IDF vectorizer. TF (Term Frequency) is termed as a word frequency in a document. IDF (Inverse Document Frequency) is universe document frequency. The TF-IDF Vectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents. It transforms text to feature vectors that can be used as input to estimator. Vocabulary is a dictionary that converts each token (word) to feature index in the matrix, each unique token gets a feature index. It tells you that the token 'me' is represented as feature number 8 in the output matrix. A vocabulary of 8 words is learned from the documents and each word is assigned a unique integer index in the output vector. In this paper, TF-IDF that takes stop_words as a parameter transforms book description into matrix of vectors.

3.4 User Interface

User interface is very essential for any project because everyone who tries to utilize the system for a purpose will try to access it using an interface. Indeed, our system also has a user interface built to facilitate users to utilize the services we provide. Users in our system can login/signup using the interface provided to them. They can view all the existing books in our database. The books that are extracted from the datasets are stored in a database. They can search any book by its title or by its author. Users can also view the books they have rated and they can also logged out themselves. Web application interface is what we call as the front-end of our project. This can be accessed from any browser. The interface has been built using Django Framework.

CHAPTER-4
EXPERIMENTAL ANALYSIS AND RESULTS

4.1 System Requirements

A requirement is a feature that the system must have or a constraint that it must to be accepted by the client. Requirement Engineering aims at defining the requirements of the system under construction. Requirement Engineering include two main activities requirement elicitation which results in the specification of the system that the client understands and analysis which in analysis model that the developer can unambiguously interpret. A requirement is a statement about what the proposed system will do.

Requirements can be divided into two major categories:

- Functional Requirements.
- Non-Functional Requirements.

4.1.1 Functional Requirements:

A Functional Requirement is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements describe the interactions between the system and its environment independent of its application.

- Applying the algorithms on the train data
- Display the recommendations by the model.

4.1.2 Non-Functional Requirements:

Non-Functional Requirements specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the

software system.

Example of nonfunctional requirement, *"how fast does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs.

- Accuracy
- Reliability
- Flexibility

4.2 System Configuration

4.2.1 Software Requirements

1. Software:

    - Python Version 3.0 or above
    - Django Framework
    - MySQL python connector version 8.0 or above

2. Operating System: Windows 10

3. Database server: MySQL

4. Tools: Microsoft Visual Studio, Xampp, Web Browser (Google Chrome or Firefox)

5. Python Libraries: Numpy, pandas, sklearn, Pickle, Matplotlib, Seaborn

4.2.1.1 Introduction to Python

Python is an interpreted, high-level, general-purpose programming language. Python is simple and easy to read syntax emphasizes readability and therefore reduces system maintenance costs. Python supports modules and packages, which promote system layout and code reuse. It saves space but it takes slightly higher time when its code is compiled. Indentation needs to be taken care while coding.

Python does the following:

- Python can be used on a server to create web applications.
- It can connect to database systems. It can also read and modify files.
- It can be used to handle big data and perform complex mathematics.

- It can be used for production-ready software development.

Python has many inbuilt library functions that can be used easily for working with machine learning algorithms. All the necessary python libraries must be pre-installed using "pip" command.

4.2.1.2 Introduction to Django Framework

Django is a Python-based free and open-source web application framework that follows the model–template–views architectural pattern. Django was developed to ease the creation of database driven websites and user reusability of components. It is an advanced Python web framework which allows faster development of secure and maintainable websites.

Django takes great care of the web development, so you can focus on writing your app without having to update the wheel. The core framework of Django is light weight, stand-alone web server for development and testing. The main design principles of Django are DRY– Don't Repeat Yourself and CRUD – Create Read Update and Delete. It's designed to feel comfortable and easy-to-learn to those used to working with HTML, like designers and front-end developers. Django provides a powerful form library that handles rendering forms as HTML, validating user-submitted data, and converting that data to native Python types.

4.2.1.3 Python Libraries

NumPy:

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code

- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Pandas:

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is fast and it has high-performance & productivity for users. It provides high-performance and is easy-to-use data structures and data analysis tools for the Python language. Pandas is used in a wide range of fields including academic and commercial domains including economics, Statistics, analytics, etc.

SKLearn:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It is an open-source Python library that implements a range of machine learning, pre-processing, cross-validation and visualization algorithms using a unified interface. Sklearn provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Pickle:

Python pickle module is used for serializing and de-serializing a Python object structure. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script. Pickling is useful for applications where you need some degree of persistency in your data. Your program's state data can be saved to disk, so you can continue working on it later on.

Matplotlib:

It is a very powerful plotting library useful for those working with Python and NumPy. And for making statistical interference, it becomes very necessary to visualize our data and Matplotlib is the tool that can be very helpful for this purpose. It provides MATLAB like interface only difference is that it uses Python and is open source.

Seaborn:

Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.
It offers the following functionalities:

- Dataset oriented API to determine the relationship between variables.
- Automatic estimation and plotting of linear regression plots.
- It supports high-level abstractions for multi-plot grids.
- Visualizing univariate and bivariate distribution.

4.2.2 Hardware Requirements

1. RAM: 4 GB or above
2. Storage: 30 to 50 GB
3. Processor: Any Processor above 500MHz

4.3 Feasibility Study

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Economic Feasibility

- Technical Feasibility
- Operational Feasibility

4.3.1 Economic Feasibility:

As system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies java1.6 open source, there is nominal expenditure and economic feasibility for certain.

4.3.2 Technical Feasibility:

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves evaluation of the hardware, software, and other technology requirements of the proposed system. This assessment is based on an outline design of system requirements, to determine whether the company has the technical expertise to handle completion of the project. When writing a feasibility report, the following should be taken to consideration:

- A brief description of the business to assess more possible factors which could affect the study
- The part of the business being examined
- The human and economic factor
- The possible solutions to the problem At this level, the concern is whether the proposal is both technically and legally feasible (assuming moderate cost). The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

4.3.3 Operational Feasibility:

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation.

Some of the important issues raised are to test the operational feasibility of a project includes the following:

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits? This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

4.4 Sample Code

a)collaborative filtering.py

```
import numpy as np

import pandas as pd


from google.colab import drive

drive.mount('/content/drive',force_remount = True)


ratings = pd.read_csv('/content/drive/MyDrive/Datasets/ratings.csv', sep=',', error_bad_lines=False, encoding="latin-1")
```

```
books = pd.read_csv('/content/drive/MyDrive/Datasets/book_data2.csv', sep=',', error_bad_lines=False, encoding="latin-1")

display(ratings.head())

display(books.head())


#ratings = ratings.iloc[:400000,:]

ratings.shape


# Merge the two tables then pivot so we have Users X Books dataframe.

ratings_title = pd.merge(ratings, books[['book_id', 'book_title']], on='book_id' )

user_book_ratings = pd.pivot_table(ratings_title, index='user_id', columns= 'book_title', values='rating')


print('dataset dimensions: ', user_book_ratings.shape, '\n\nSubset example:')

user_book_ratings.iloc[:25, :10]


# Drop users that have given fewer than 100 ratings of these most-rated books

user_book_ratings = user_book_ratings.dropna(thresh=100)

print('dataset dimensions: ', user_book_ratings.shape, '\n\nSubset example:')

user_book_ratings.iloc[:25, :10]


from sklearn.decomposition import TruncatedSVD


# replace NaN's with zeroes for Truncated SVD

user_book_ratings_without_nan = user_book_ratings.fillna(0)
```

```
tsvd = TruncatedSVD(n_components=200, random_state=42)
user_book_ratings_tsvd = tsvd.fit(user_book_ratings_without_nan).transform(user_book_ratings_without_nan)
print('Original number of features:', user_book_ratings_without_nan.shape[1])
print('Reduced number of features:', user_book_ratings_tsvd.shape[1])
print('Explained variance ratio:', tsvd.explained_variance_ratio_[0:200].sum())

# view result in a Pandas dataframe, applying the original indices
indices = user_book_ratings.index

book_ratings_for_clustering = pd.DataFrame(data=user_book_ratings_tsvd).set_index(indices)
print('dataset dimensions: ', book_ratings_for_clustering.shape, '\n\nSubset example:')

book_ratings_for_clustering.iloc[:25, :10]

from sklearn.model_selection import train_test_split
book_ratings_training, book_ratings_testing = train_test_split(book_ratings_for_clustering, test_size=0.20, random_state=42)

print('Training data shape: ', book_ratings_training.shape)
print('Testing data shape: ', book_ratings_testing.shape)
book_ratings_testing.head()
```

```
# find the per-book ratings of the test set
indices = book_ratings_testing.index
test_set_ratings = user_book_ratings.loc[indices]
test_set_ratings.head()

mean_ratings_for_random_10 = []

# for each user, pick 10 books at random that the reader has rated and get the reader's average score for those books
for index, row in test_set_ratings.iterrows():
    ratings_without_nas = row.dropna()
    random_10 = ratings_without_nas.sample(n=10)
    random_10_mean = random_10.mean()
    mean_ratings_for_random_10.append(random_10_mean)

# get the mean of the users' mean ratings for 10 random books each
mean_benchmark_rating = sum(mean_ratings_for_random_10) / len(mean_ratings_for_random_10)

print('Mean rating for 10 random books per test user: ', mean_benchmark_rating)

# trying with the training data after preprocessing
from sklearn.cluster import KMeans

clusterer_KMeans = KMeans(n_clusters=7).fit(book_ratings_training)
```

```
preds_KMeans = clusterer_KMeans.predict(book_ratings_training)

from sklearn.metrics import silhouette_score
kmeans_score = silhouette_score(book_ratings_training, preds_KMeans)
print(kmeans_score)

# trying with the training data after preprocessing
from sklearn.mixture import GaussianMixture

clusterer_GMM = GaussianMixture(n_components=7).fit(book_ratings_training)
preds_GMM = clusterer_GMM.predict(book_ratings_training)

GMM_score = silhouette_score(book_ratings_training, preds_GMM)
print(GMM_score)

indices = book_ratings_training.index
preds = pd.DataFrame(data=preds_KMeans, columns=['cluster']).set_index(indices)
preds.head()

# get a list of the highest-rated books for each cluster
def get_cluster_favorites(cluster_number):
    # create a list of cluster members
    cluster_membership = preds.index[preds['cluster'] == cluster_number].tolist()
    # build a dataframe of that cluster's book ratings
```

```
    cluster_ratings = user_book_ratings.loc[cluster_membership]
    # drop books that have fewer than 10 ratings by cluster members
    cluster_ratings = cluster_ratings.dropna(axis='columns', thresh=10)
    # find the cluster's mean rating overal and for each book
    means = cluster_ratings.mean(axis=0)
    # sort books by mean rating
    favorites = means.sort_values(ascending=False)
    return favorites

# for each cluster, determine the overall mean rating cluster members have given books
def get_cluster_mean(cluster_number):
    # create a list of cluster members
    cluster_membership = preds.index[preds['cluster'] == cluster_number].tolist()
    # create a version of the original ratings dataset that only includes cluster members
    cluster_ratings = ratings[ratings['user_id'].isin(cluster_membership)]
    # get the mean rating
    return cluster_ratings['rating'].mean()

cluster0_books_storted = get_cluster_favorites(0)
cluster0_mean = get_cluster_mean(0)

print('The cluster 0 mean is:', cluster0_mean)
cluster0_books_storted[0:10]
```

```
cluster1_books_storted = get_cluster_favorites(1)

cluster1_mean = get_cluster_mean(1)


print('The cluster 1 mean is:', cluster1_mean)

cluster1_books_storted[0:10]


cluster2_books_storted = get_cluster_favorites(2)

cluster2_mean = get_cluster_mean(2)


print('The cluster 2 mean is:', cluster2_mean)

cluster2_books_storted[0:10]


cluster3_books_storted = get_cluster_favorites(3)

cluster3_mean = get_cluster_mean(3)


print('The cluster 3 mean is:', cluster3_mean)

cluster3_books_storted[0:10]


cluster4_books_storted = get_cluster_favorites(4)

cluster4_mean = get_cluster_mean(4)


print('The cluster 4 mean is:', cluster4_mean)

cluster4_books_storted[0:10]
```

```
cluster5_books_storted = get_cluster_favorites(5)

cluster5_mean = get_cluster_mean(5)


print('The cluster 5 mean is:', cluster5_mean)

cluster5_books_storted[0:10]


cluster6_books_storted = get_cluster_favorites(6)

cluster6_mean = get_cluster_mean(6)


print('The cluster 6 mean is:', cluster6_mean)

cluster6_books_storted[0:10]


# associate each test user with a cluster

test_set_preds = clusterer_KMeans.predict(book_ratings_testing)

test_set_indices = book_ratings_testing.index

test_set_clusters = pd.DataFrame(data=test_set_preds,
columns=['cluster']).set_index(test_set_indices)


test_set_clusters


mean_ratings_for_cluster_favorites = []


# put each cluster's sorted book list in an array to reference
```

```
cluster_favorites = [cluster0_books_storted, cluster1_books_storted, cluster2_books_storted, cluster3_books_storted, cluster4_books_storted, cluster5_books_storted, cluster6_books_storted]


# for each user, find the 10 books the reader has rated that are the top-rated books of the cluster.
# get the reader's average score for those books
for index, row in test_set_ratings.iterrows():
    user_cluster = test_set_clusters.loc[index, 'cluster']
    favorites = cluster_favorites[user_cluster].index
    user_ratings_of_favorites = []
    # proceed in order down the cluster's list of favorite books
    for book in favorites:
        # if the user has given the book a rating, save the rating to a list
        if np.isnan(row[book]) == False:
            user_ratings_of_favorites.append(row[book])
        # stop when there are 10 ratings for the user
        if len(user_ratings_of_favorites) >= 10:
            break
    # get the mean for the user's rating of the cluster's 10 favorite books
    mean_rating_for_favorites = sum(user_ratings_of_favorites) / len(user_ratings_of_favorites)
    mean_ratings_for_cluster_favorites.append(mean_rating_for_favorites)


mean_favorites_rating = sum(mean_ratings_for_cluster_favorites) / len(mean_ratings_for_cluster_favorites)
```

```
print('Mean rating for 10 random books per test user: ', mean_benchmark_rating)

print('Mean rating for 10 books that are the cluster\'s favorites: ', mean_favorites_rating)

print('Difference between ratings: ', mean_favorites_rating-mean_benchmark_rating)


from sklearn.metrics import mean_squared_error

rmse = mean_squared_error(mean_ratings_for_random_10,mean_ratings_for_cluster_favorites,squared = False)

# taking root of mean squared error

print(rmse)

accuracy = 1.96 * rmse

print(accuracy)


import random

def recommend(cluster_assignments, user_id):
    user_cluster = cluster_assignments
    favorites = get_cluster_favorites(user_cluster).index
    favorites = random.choices(favorites, k=10)
    return favorites
recommendation8667 = recommend(5, 8667)

print(recommendation8667)
```

b) contentbasedrecommender.py

```
from google.colab import drive
```

```
drive.mount('/content/drive',force_remount = True)


# importing libraries

import pandas as pd

from sklearn.metrics.pairwise import linear_kernel

from sklearn.feature_extraction.text import TfidfVectorizer


# reading file

book_description = pd.read_csv('/content/drive/MyDrive/Datasets/book_data2.csv',
sep=',', error_bad_lines=False, encoding="latin-1")


# checking if we have the right data

book_description.head()

# removing the stop words

books_tfidf = TfidfVectorizer(stop_words='english')

# replace NaN with empty strings

book_description['book_desc'] = book_description['book_desc'].fillna('')

# computing TF-IDF matrix required for calculating cosine similarity

book_description_matrix = books_tfidf.fit_transform(book_description['book_desc'])


# Let's check the shape of computed matrix

book_description_matrix.shape


# compuing cosine similarity matrix using linear_kernal of sklearn

cosine_similarity = linear_kernel(book_description_matrix, book_description_matrix)
```

```
# Get the pairwsie similarity scores of all books compared to the book passed by index, sorting them and getting top 5

similarity_scores = list(enumerate(cosine_similarity[1]))

similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)

similarity_scores = similarity_scores[1:6]

# Get the similar books index

books_index = [i[0] for i in similarity_scores]

# printing the top 5 most similar books using integer-location based indexing (iloc)

print (book_description['book_title'].iloc[books_index])
```

4.5 Experimental analysis and Performance Measures

The books dataset which initially has 2080 features in it has been reduced to 200 features after using Truncated SVD in feature extraction. This reduced dataset has been used to build the models. By calculating the coefficient of silhouette, the quality of clustering of different trained models can be compared. A sample of the training dataset is shown in below fig-4.1.

```
[28] book_ratings_training.head()
```

user_id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
8189	22.572424	-6.820165	-5.953795	0.905107	11.310563	-1.428082	4.573312	-1.635167	6.082113	-4.096796	-0.497332	8.997144	-1.241497	-4.206573	0.641806	0.909021	2.211293	8
2250	18.133117	3.698193	-6.303387	4.812951	2.755407	-1.063095	-1.056664	-5.131466	5.679201	-1.436058	7.935320	-0.324121	3.868993	-0.609024	-0.444186	-2.080864	2.621088	1
5390	15.577047	12.559324	-0.427927	-7.224622	4.501250	2.297199	5.584461	7.343759	-7.907567	-7.821827	-4.379296	-2.841594	2.361561	-2.466796	0.225870	-11.129855	3.851802	0
1115	14.681320	-1.097166	-0.086704	-2.504046	-5.669789	4.067409	-1.097313	1.627067	2.789073	0.832866	-2.789175	-1.611497	-1.582967	-1.371035	0.002093	-1.508835	0.135221	2
5372	19.534503	-8.743696	5.176984	9.799347	5.380737	-0.341633	2.349641	-4.836639	-4.317742	0.123264	-2.351799	2.482326	-6.760441	-4.456905	-0.426022	-1.674257	-0.734462	-1

5 rows × 200 columns

Fig 4.1- Sample Training Data

After building the model using the training data for clustering, the next step is to measure the performance of the model. To evaluate the efficacy of the model, silhouette score is calculated.

4.5.1 Root Mean Square Error

Root mean squared error (RMSE) is the square root of the mean of the square of all of the error. The use of RMSE is very common, and it is considered an excellent general-purpose error metric for numerical predictions. The accuracy calculated by taking the RMSE value is known as Fundamental vertical accuracy whose value is computed by 1.96 * RMSE.

Root mean square error can be expressed as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

Where N is the number of data points, y(i) is the ith measurement, and y^(i) is its corresponding prediction.

By squaring errors and calculating a mean, RMSE can be heavily affected by a few predictions which are much worse than the rest. If this is undesirable, using the absolute value of residuals and/or calculating median can give a better idea of how a model performs on most predictions, without extra influence from unusually poor predictions.

4.5.2 Cosine Similarity Matrix

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, the higher the cosine similarity. You have to compute the cosine similarity matrix which contains the pairwise cosine similarity score for every pair of sentences (vectorized using tf-idf). Remember, the value corresponding to the ith row and jth column of a similarity matrix denotes the similarity score for the ith and jth vector. Use linear_kernel() and pass tfidf_matrix to compute the cosine similarity matrix cosine_sim.

```
[ ] # compuing cosine similarity matrix using linear_kernal of sklearn
    cosine_similarity = linear_kernel(book_description_matrix, book_description_matrix)
```

```
# Get the pairwsie similarity scores of all books compared to the book passed by index, sorting them and getting top 5
similarity_scores = list(enumerate(cosine_similarity[1]))
similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)
similarity_scores = similarity_scores[1:6]
# Get the similar books index
books_index = [i[0] for i in similarity_scores]

# printing the top 5 most similar books using integer-location based indexing (iloc)
print (book_description['book_title'].iloc[books_index])
```

Fig-4.2 Building Cosine similarity matrix using linear kernel

4.5.3 Performance Analysis and Models Comparison

Out of all the trained models we need to choose the best model. We need to analyze the performance of each model and then compare the silhouette scores of the two trained models.

```
[12] # trying with the training data after preprocessing
     from sklearn.cluster import KMeans

     clusterer_KMeans = KMeans(n_clusters=7).fit(book_ratings_training)
     preds_KMeans = clusterer_KMeans.predict(book_ratings_training)

     from sklearn.metrics import silhouette_score
     kmeans_score = silhouette_score(book_ratings_training, preds_KMeans)
     print(kmeans_score)

     0.02688953262460168
```

Fig-4.3 Silhouette score for K-means Clustering model

The above Figure 4.3 shows a screenshot of notebook with silhouette score for K-Means clustering model. This coefficient of silhouette of this model is found out to be 0.02688953262460168, which is positive that means the clustering is good and appropriate.

```
[13] # trying with the training data after preprocessing
     from sklearn.mixture import GaussianMixture

     clusterer_GMM = GaussianMixture(n_components=7).fit(book_ratings_training)
     preds_GMM = clusterer_GMM.predict(book_ratings_training)

     GMM_score = silhouette_score(book_ratings_training, preds_GMM)
     print(GMM_score)

     0.0225855911136862
```

Fig-4.4 Silhouette score for Gaussian mixture model

The above Figure 4.4 shows a screenshot of notebook with silhouette score for Gaussian mixture model. This coefficient of determination of this model is found out to be 0.025855911136862, which is also positive but less than that of K-Means clustering model.

By the above data and calculations, it is evident that K-Means Clustering model is more efficient for clustering. Then we have analyzed each cluster.

```
[17] cluster0_books_storted = get_cluster_favorites(0)
     cluster0_mean = get_cluster_mean(0)

     print('The cluster 0 mean is:', cluster0_mean)
     cluster0_books_storted[0:10]
```

```
The cluster 0 mean is: 3.7488141202426917
book_title
Fifty Shades of Grey                    4.866667
The Clan of the Cave Bear               4.611111
Ready Player One                        4.583333
The Notebook                            4.526316
Lolita                                  4.517241
Harry Potter and the Goblet of Fire     4.500000
The Three Musketeers                    4.485714
A Million Little Pieces                 4.466667
The Art of Racing in the Rain           4.454545
2001: A Space Odyssey                   4.454545
dtype: float64
```

Fig-4.5 Analyzing clusters using model

We have performed our clustering on a dataset that included 200 composite features. It is difficult to create a visualization that effectively illustrates all of these features. Therefore, we have selected the two top features, which played the most significant role in the clustering, and created a scatterplot that illustrates the clusters across those features.

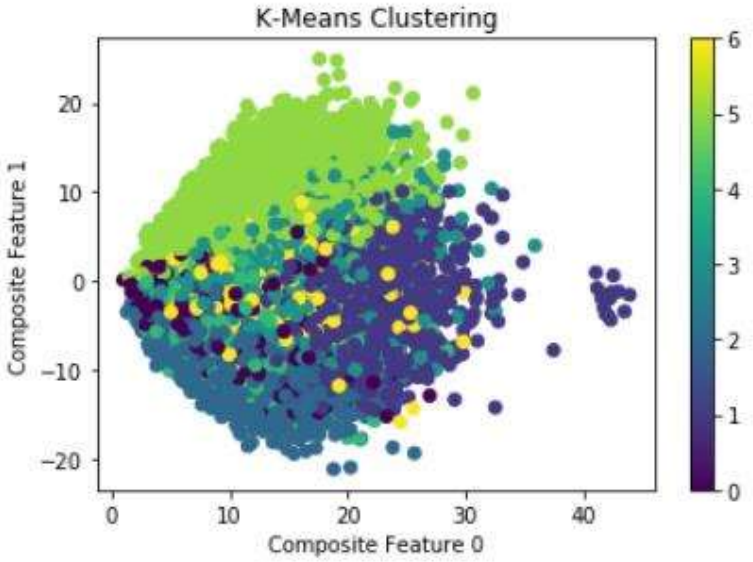


Fig-4.6 Visualizing clusters across two composite features

The fig 4.6 indicates that the clustering model was effective in grouping users based on their values for composite features 0 and 1. The scatterplot also indicates that a few users had outlier values for composite feature 0.

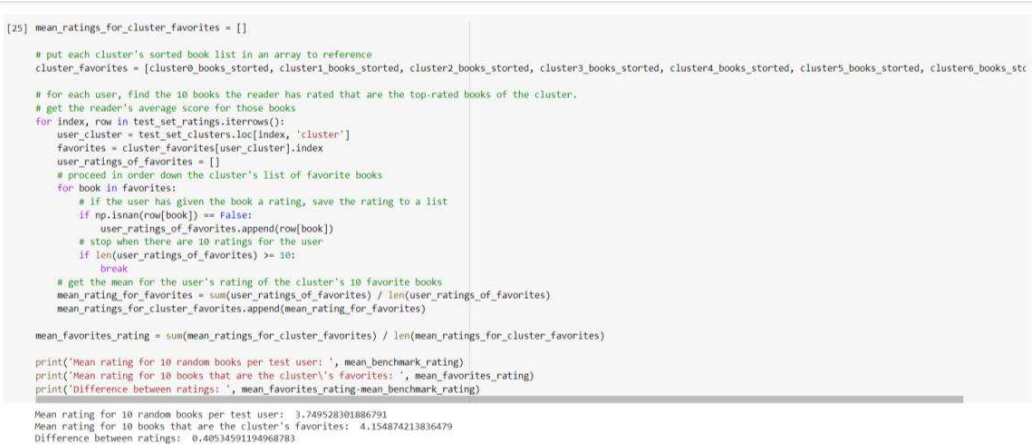
4.5.4 Experimental Analysis

The best fit model for our system has been found out through above mentioned model's comparison. The below table-4.1 shows silhouette coefficient for two models.

	K-Means	**Gaussian Mixture**
Silhouette score	0.02688953262460168	0.025855911136862

Table-4.1: Comparing Silhouette scores of two models

In the above table, we are checking the silhouette coefficient of two models. After checking the two scores, we consider the model using K-Means method with cluster count of 7 because of higher score. Then, this model is used to cluster the users.



```
[25] mean_ratings_for_cluster_favorites = []

    # put each cluster's sorted book list in an array to reference
    cluster_favorites = [cluster0_books_storted, cluster1_books_storted, cluster2_books_storted, cluster3_books_storted, cluster4_books_storted, cluster5_books_storted, cluster6_books_stc

    # for each user, find the 10 books the reader has rated that are the top-rated books of the cluster.
    # get the reader's average score for those books
    for index, row in test_set_ratings.iterrows():
        user_cluster = test_set_clusters.loc[index, 'cluster']
        favorites = cluster_favorites[user_cluster].index
        user_ratings_of_favorites = []
        # proceed in order down the cluster's list of favorite books
        for book in favorites:
            # if the user has given the book a rating, save the rating to a list
            if np.isnan(row[book]) == False:
                user_ratings_of_favorites.append(row[book])
            # stop when there are 10 ratings for the user
            if len(user_ratings_of_favorites) >= 10:
                break
        # get the mean for the user's rating of the cluster's 10 favorite books
        mean_rating_for_favorites = sum(user_ratings_of_favorites) / len(user_ratings_of_favorites)
        mean_ratings_for_cluster_favorites.append(mean_rating_for_favorites)

    mean_favorites_rating = sum(mean_ratings_for_cluster_favorites) / len(mean_ratings_for_cluster_favorites)

    print('Mean rating for 10 random books per test user: ', mean_benchmark_rating)
    print('Mean rating for 10 books that are the cluster\'s favorites: ', mean_favorites_rating)
    print('Difference between ratings: ', mean_favorites_rating-mean_benchmark_rating)

    Mean rating for 10 random books per test user:  3.749528301886791
    Mean rating for 10 books that are the cluster's favorites:  4.154874213836479
    Difference between ratings:  0.4053459119496878
```

Fig-4.7 Calculating difference between mean rating before and after clustering

The fig 4.7 indicates that users in the test set, on average, rated their clusters' favorite books higher than a random set of 10 books by 0.47 stars, or nearly half a star.

RMSE (Root mean square error) measures the error caused by the deviation between the sample values and predicted values by the model. The accuracy calculated by taking the RMSE value is known as Fundamental vertical accuracy whose value is computed by 1.96 * RMSE.

RMSE	0.5957791790493179
Accuracy	1.167727190936663

Table-4.2: RMSE and Accuracy for the obtained values

4.6 RESULTS (Screenshots)

To demonstrate the results of our project, we have determined the cluster favorites to recommend the book to the user of particular cluster. This recommendation is carried out in a python notebook.

```
import random
def recommend(cluster_assignments, user_id):
    user_cluster = cluster_assignments
    favorites = get_cluster_favorites(user_cluster).index
    favorites = random.choices(favorites, k=10)
    return favorites
recommendation8667 = recommend(5, 8667)
for i in recommendation8667:
  print(i)
```

```
Starship Troopers
The Handmaid's Tale
The Lies of Locke Lamora
Charlotte's Web
2001: A Space Odyssey
Jane Eyre
The Fault in Our Stars
An Ember in the Ashes
I, Claudius
Watchmen
```

Fig-4.8 Recommendation of books for a user

Now, we made recommendations for a user as shown in the fig 4.8. As like the above data, every time we want recommendations, we need to continue the whole process. But the users who are not familiar with programming or python notebooks will find difficult to do the whole thing. To get rid of this type of problems we need to have a user interface from which everyone can access and use our system at ease.

For our system, we have built User interface i.e., our front-end using Django web framework. We have created some Html pages and connected them to our python modules using Django. Users have login credentials which are used to login into website. To see how our interface works, we have taken some screenshots while checking the functionalities our system.

The following screenshots will demonstrate the final results of our project:

- A new user can be signed up through this page. In Figure 4.9, we can have a clear look at this page. There is button with the name register at the bottom of the page. After filling all the details which are required to register, click on the

register button. Then it will be redirected to a python module to which it is interfaced.

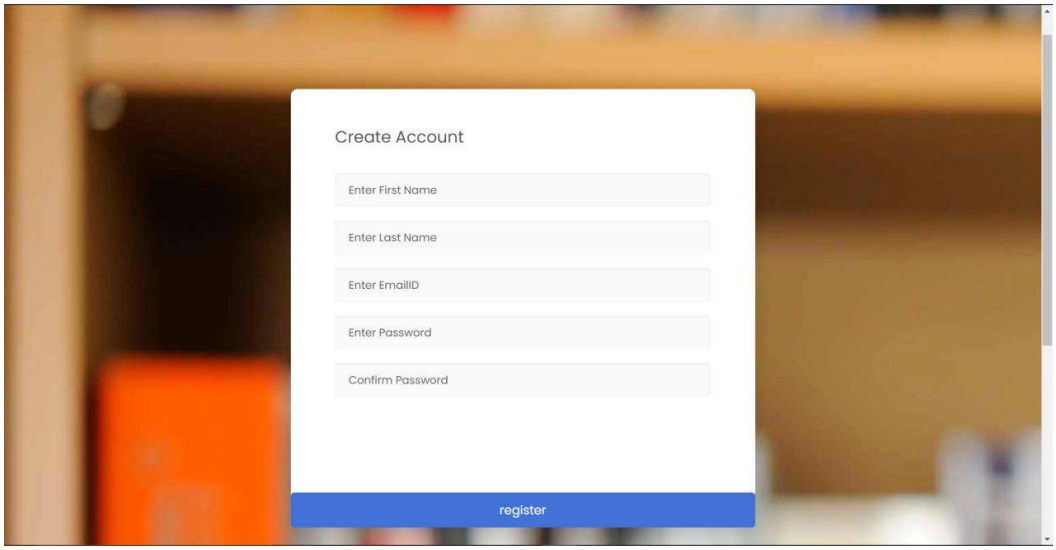


Fig-4.9 Register a user

- Using the page shown in Figure 4.9, we have registered a new user. In the figure 4.10 we can see the login page in which user can login into Home page with login credentials. If we didn't register, we cannot login to the website.

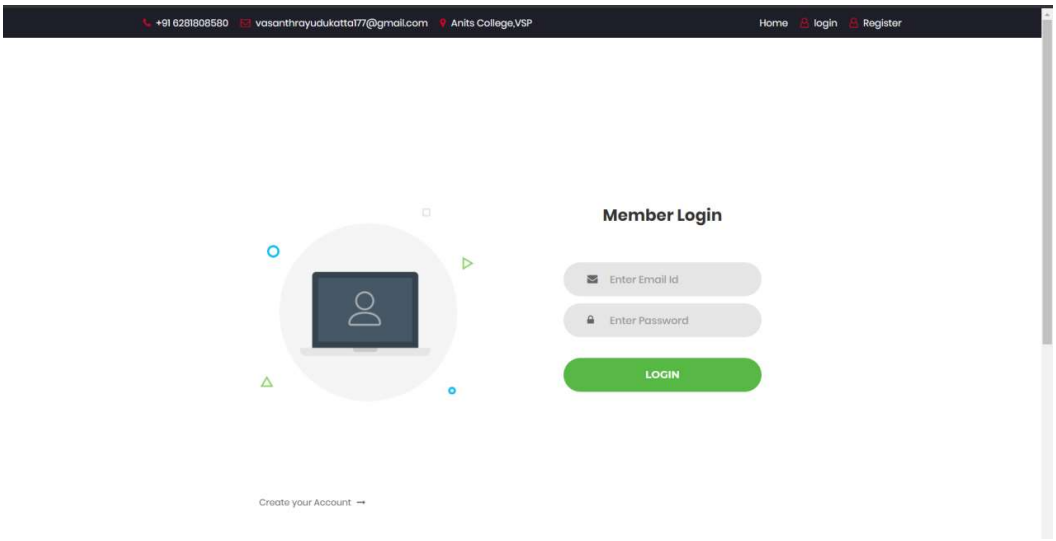


Fig-4.10 Login page

- The recommendation of books for users is shown in this page i.e. Home page. The recommendations shown in this figure based on collaborative filtering

system. This is the page that we have designed where our users can search for any book which available in our database. The page can be seen in the figure 4.11. This page can be accessed with user login.

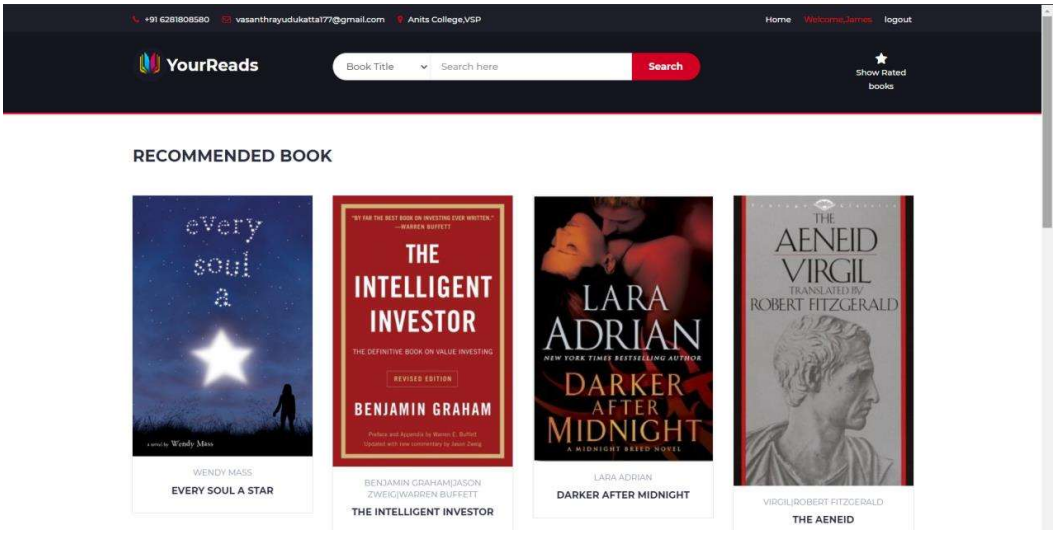


Fig-4.11 Recommended books in Home Page

- Under the Recommended books, there are popular books which are recommended by the Content Based Filtering system are also shown in the Home page, which are named as popular books. These books were shown to the new users also irrespective of the user data. We can see the books in Figure 4.12.

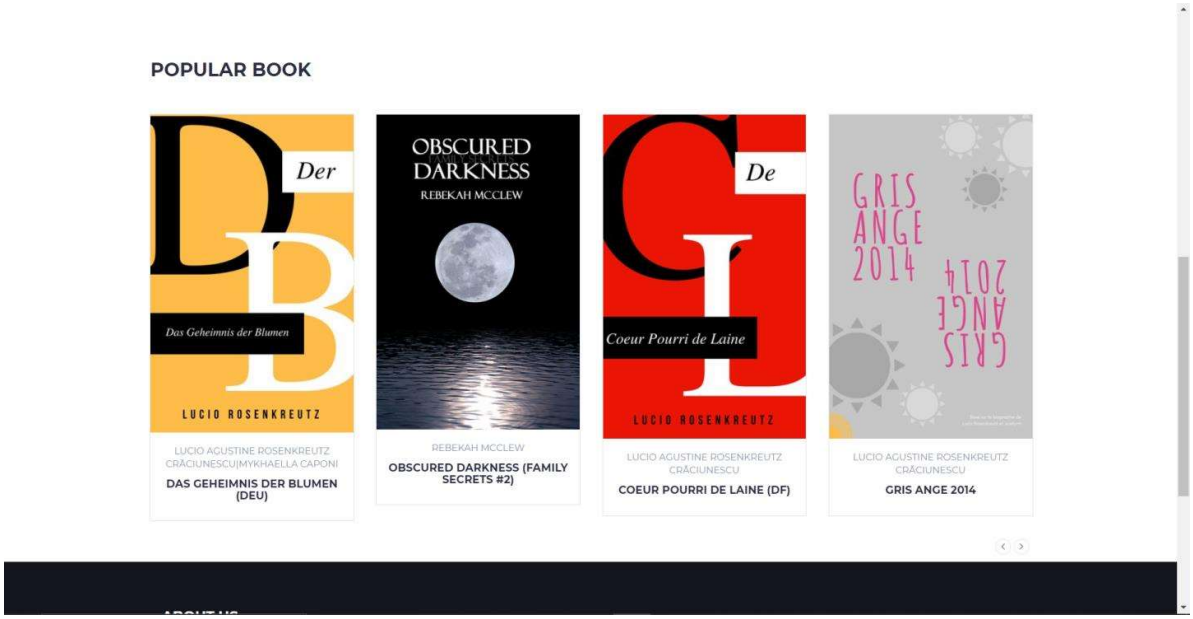


Fig-4.12 Popular books in Home Page

- After signing into our site, users can search for any book by entering the details of that particular book. Users can search book by using book author or book title. We can see the details of the book after searching with book title in fig 4.13.

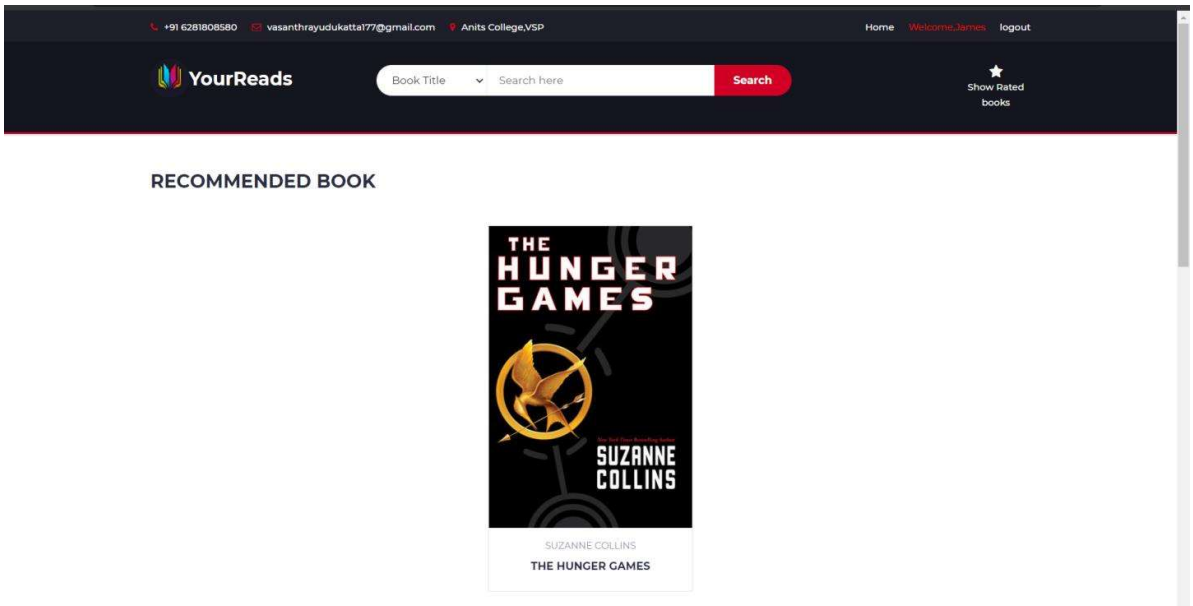


Fig-4.13 Searching book with book title

- Just like by searching the book with book title, user can also search the book by its author name. We can see the details of the book after searching with book author name in fig 4.14.

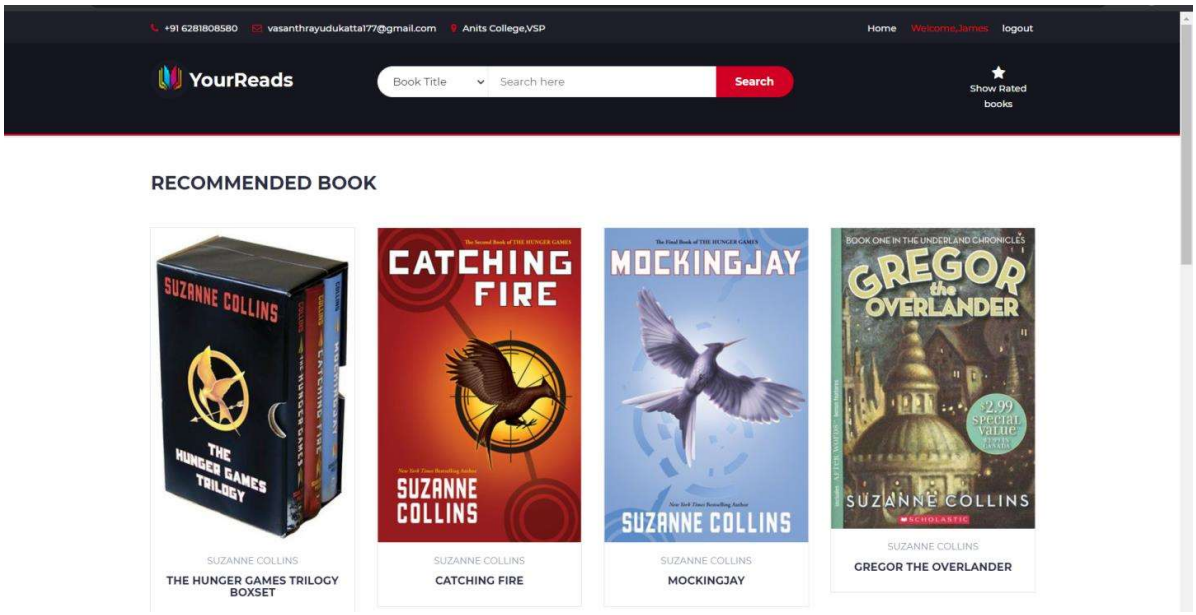


Fig-4.14 Searching book with book author name

- We can give rating to any book whichever we want after clicking on the book. After clicking on the book, we are redirected to another page where we can give the rating to the back. Figure 4.15 shows 5 black stars under the details of the book where we can give rating by clicking on the stars. After giving rating and clicking on the submit button, our rating to the book is stored.

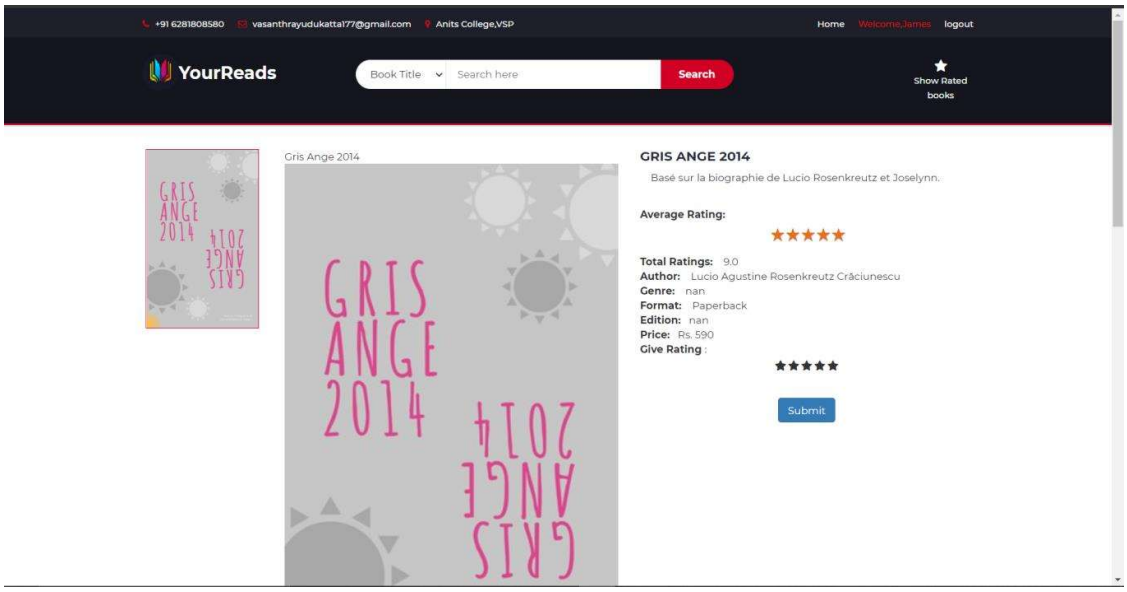


Fig-4.15 Giving Rating to the book

- After giving rating to the book, we can see that book in another page which is redirected after clicking on the Show rated books button. There we can see the books which are rated by us in sequential order. Figure 4.16 shows the screenshot of the books which are rated by user.

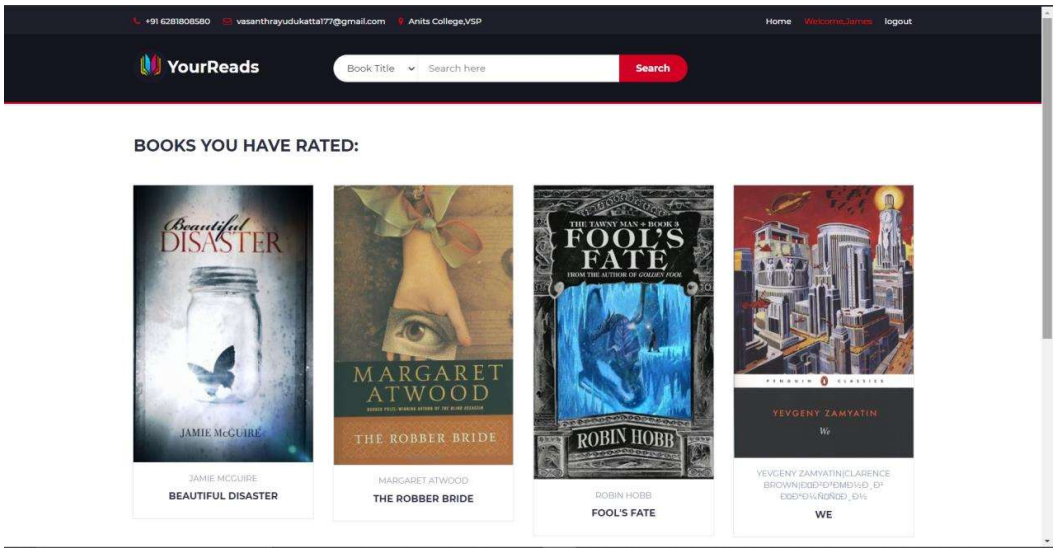


Fig-4.16 Books which are rated by user

CHAPTER-5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this project, we have recommended the books for a user using the model trained using K-Means Clustering which is a Collaborative Filtering Technique. We have also compared different models built using different methods and identified the best model and justifies why it has chosen that model. We have used the books dataset that is available in the Goodreads website which consists of more than 3000 books. The models are built using the reduced features which is done by Truncated SVD. Based on those features the author built a model that gives a positive Silhouette score. The model that is suggested by this paper is useful for book readers. The system we have developed can make recommendations for new users also.

5.2 Future Work

The System has adequate scope for modification in future if it is necessary. Development and launching of Mobile app and refining existing services and adding more service, System security, data security and reliability are the main features which can be done in future. The API for the shopping and payment gateway can be added so that we can also buy a book at the moment. In the existing system there are only some selected categories, so as an extension to the site we can add more categories as compared to existing site. Also we can add admin side with some functionalities like books management, User management etc.

REFERENCES

[1] Avi Rana and K. Deeba et.al, "Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)" in IOP ebooks 1362, 2019.

[2] G. Naveen Kishore, V. Dhiraj, Sk Hasane Ahammad, Sivaramireddy Gudise, Balaji Kummaraa and Likhita Ravuru Akkala, "Online Book Recommendation System" International Journal of Scientific & Technology Research vol.8, issue 12, Dec 2019.

[3] Uko E Okon, B O Eke and P O Asaga, "An Improved Online Book Recommender System using Collaborative Filtering Algorithm", International Journal of Computer Applications vol.179-Number 46, 2018.

[4] Jinny Cho, Ryan Gorey, Sofia Serrano, Shatian Wang, JordiKai Watanabe-Inouye, "Book Recommendation System" Winter 2016.

[5] Ms. Sushma Rjpurkar, Ms. Darshana Bhatt and Ms. Pooja Malhotra, "Book Recommendation System" International Journal for Innovative Research in Science & Technology vol.1, issue 11, April 2015.

[6] Abhay E. Patil, Simran Patil, Karanjit Singh, Parth Saraiya and Aayusha Sheregar, "Online Book Recommendation System using Association Rule Mining and Collaborative Filtering" International Journal of Computer Science and Mobile Computing vol.8, April 2019.

[7] Suhas Patil and Dr. Varsha Nandeo, "A Proposed Hybrid Book Recommender System" International Journal of Computer Applications vol.6 – No.6,Nov – Dec 2016.

[8] Ankit Khera, "Online Recommendation System" SJSU ScholarWorks, Masters Theorem and Graduate Research, Master's Projects, 2008.

[9] Anagha Vaidya and Dr. Subhash Shinde, “Hybrid Book Recommendation System” International Research Journal of Engineering and Technology (IRJET), July 2019.

[10] Dhirman Sarma, Tanni Mittra and Mohammad Shahadat Hossain, ”Personalized Book Recommendation System using Machine Learning Algorithm” The Science and Information Organization vol.12,2019.

[11] Ayub, M., Ghazanfar, M.A., Maqsood, M. and Saleem, A. (2018). A Jaccard base similarity measure to improve performance of CF based recommendation system. Proceedings of International Conference on Information Networking (ICOIN).

[12] Choi, S.H., Jeong, Y.S. and Jeong, M.K. (2010). A Hybrid Recommendation Method with Reduced Data for Large-Scale Application. IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews, Vol. 40, No.5, September 2010.

APPENDICES

BASE PAPER

Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)

Recent citations

- M. Arulmozhivarman and Gerard Deepak
- Vanita Jain *et al*
- Andrés García-Pérez *et al*

To cite this article: Avi Rana and K. Deeba 2019 *J. Phys.: Conf. Ser.* **1362** 012130

View the article online for updates and enhancements.

IOP ebooks™

Bringing together innovative digital publishing with leading authors from the global scientific community.

Start exploring the collection–download the first chapter of every title for free.

This content was downloaded from IP address 157.47.89.119 on 18/06/2021 at 09:07

Journal of Physics: Conference Series **1362** (2019) 012130 doi:10.1088/17426596/1362/1/012130

Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)

Avi Rana[1], K. Deeba[2]

[1,2]SRM Institute of Science and Technology,Chennai,India

avijrana@gmail.com[1], deeba.k@ktr.srmuniv.ac.in[2]

Abstract-- Recommendation System (RS) is software that suggests similar items to a purchaser based on his/her earlier purchases or preferences. RS examines huge data of objects and compiles a list of those objects which would fulfil the requirements of the buyer. Nowadays most ecommerce companies are using Recommendation systems to lure buyers to purchase more by offering items that the buyer is likely to prefer. Book Recommendation System is being used by Amazon, Barnes and Noble, Flipkart, Goodreads, etc. to recommend books the customer would be tempted to buy as they are matched with his/her choices. The challenges they face are to filter, set a priority and give recommendations which are accurate.

RS systems use Collaborative Filtering (CF) to generate lists of items similar to the buyer's preferences. Collaborative filtering is based on the assumption that if a user has rated two books then to a user who has read one of these books, the other book can be recommended (Collaboration). CF has difficulties in giving accurate recommendations due to problems of scalability, sparsity and cold start. Therefore this paper proposes a recommendation that uses Collaborative filtering with Jaccard Similarity (JS) to give more accurate recommendations. JS is based on an index calculated for a pair of books. It is a ratio of common users (users who have rated both books) divided by the sum of users who have rated the two books individually. Larger the number of common users higher will be the JS Index and hence better recommendations. Books with high JS index (more recommended) will appear on top of the recommended books list.

Keywords: Similarity index, filtering techniques, recommender system, Jaccard Similarity

1. INTRODUCTION

Recommendation system filters information by predicting ratings or preferences of consumers for items that the consumer would like to use [1]. It tries to recommend items to the consumer according to his/her needs and taste. RS mainly uses two methods to filter information - Content-based and Collaborative filtering. Content-based filtering involves recommending those items to a consumer which are similar in content to the items that have already been used by him/her. First, it makes a profile of the consumer, which consists of his/her taste. Taste is based on the type of books rated by the consumer. The system analyses the books that were liked by the consumer with the books he had not rated and looks for similarity. Out of these unrated books, the books with the maximum value of similarity index will be recommended to the consumer. Paul Resnick and Hal Varian were the ones who suggested Collaborative filtering algorithm in 1997. It became popular amid the various frameworks available at that time. [2] [3] [4]

A complete RS contains three main things: user resource, item resource and the recommendation algorithm. In the user model, the consumers' interests are analysed, similarly, the item model analyses the items' features. Then, the characteristics of the consumer are matched with the item characteristics to estimate which items to recommend using the recommendation algorithm. The performance of this algorithm is what affects the performance of the whole system.

In memory-based CF, the book ratings are directly used to assess unknown ratings for new books. This method can be subdivided into two ways: User-based approach and Item-based approach.

1. User-based approach: As per this method, customers with alike choices form a neighbourhood. If an item is not rated by the buyer, but it has been rated highly by other members of the neighbourhood, then it can be endorsed to the buyer. Hence the buyer's choices can be predicted based on the neighbourhood of similar buyers. [2] [3] [4]
2. Item-based approach: In this method, similarity between the group of objects rated emphatically by the buyer and the required object is calculated. The items which are very alike are selected. Recommendation is computed by finding the weighted means of user's ratings of the alike objects. [5]

To obtain accurate and faster recommendations researchers have combined the different recommender technologies, these are known as Hybrid recommendation systems. Some Hybrid recommendation approaches are:

- Separate execution of procedures and connecting the results.
- Using some content filtering guidelines with community CF.
- Using some principles of CF in content filtering recommender
- Using both Content and Collaborative filtering in a recommender

Additionally there is on-going research on Semantic-based, Context-aware, Cross-lingual, Cross- domain and peer-to-peer methods [6].

2. LITERATURE SURVEY

Okon et.al. (2018) [7] proposed a model that generates recommendations to buyers, 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.

Kurmashov et.al. (2015) [8] used Pearson correlation coefficient based CF to provide internet based recommendations to book readers and evaluated the system through an online survey.

Mathew et.al. (2016) [9] proposed a system that saves details of books purchased by the user. From these Book contents and ratings, a hybrid algorithm using collaborative filtering, content-based filtering and association rule generates book recommendations. Rather than Apriori, they recommended the use of Equivalence class Clustering and bottom up Lattice Transversal (ECLAT) as this algorithm is faster due to the fact that it examines the entire dataset only once.

Parvatikar et.al. (2015) [10] proposed item-based collaborative filtering and association rule mining to give recommendations. Similarity between different users was computed through Adjusted Cosine Vector Similarity function. Better recommendations were obtained as through this method data sparsity problem was removed.

Ayub et.al. (2018) [11] proposed a similarity function similar to Jaccard Similarity to locate alike items and users for the enquiring item and user in nearest neighbour based collaborative filtering. They proposed that absolute value of ratings should be taken as against the ratio of co-rated items taken in Jaccard Similarity. They also compared performance of their method with other similarity measures.

Gogna and Majumdar [12] suggested the use of buyer's demographic and item category to overcome data sparsity and cold start problems in their movie recommendation system. Latent Factor Model (LFM) was used. They developed a matrix to match the buyer and user information to get a dense user and dense item matrix. Label Consistency map, the outcome of this system was used to suggest unrated and other items to new buyers.

Chatti et.al. (2013) [13] 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.

Choi et.al. (2010) [14] proposed RS based on HYRED, a hybrid algorithm using both content and collaborative filtering on a compact dataset (by reducing user interest items) and neighbor data. HYRED used altered Pearson Coefficient based Collaborative filtering and distance-to-boundary (DTB) Content filtering. This would result in better and faster recommendation for large amount of data.

Liu et.al. (2012) [15] added the dimension of user-interest. They proposed iExpand, a 3 tier model i.e. user, user-interest and item. This helped in overcoming the issues of overspecialization and cold-start as well as reducing computation costs.

Feng et.al. (2018) [16] proposed a RS for movies based on a similarity model constituted of factors S_1 (similarity between users), S_2 (ratio of co-rated items) and S_3 (user's rating choice weight). This RS was particularly useful for sparse datasets.

Literature survey suggested that recommendation systems are being used by large number of online marketers to increase their sales by offering products to customers which match their tastes.

These RS suffer from many problems such as data sparsity, cold start, trust, scalability and privacy. Therefore there is need for improved recommendation systems which solve these issues.

Most researchers use Adjusted Cosine Vector Similarity function to compute similarity among book ratings to recommend books, to take into account that users have different rating schemes. Some rate items high usually while others usually rate low.

3. PROPOSED WORK

According to Aggarwal, C.C., Collaborative Filtering method is used in recommendation systems to develop recommendations based on ratings provided by the other users of the system [17]. If buyer's ratings of items match, it is likely that their ratings of other items will also match, this is the basic assumption of CF. Computers cannot gauge qualitative factors such as taste or quality, therefore recommendations based on the ratings of humans who can rate on the basis of qualitative factors, i.e. collaboration, will give a better outcome.

Soanpet Sree Lakshmi and Dr. T. Adi Lakshmi in their work have described in detail the problems like overspecialization, data sparsity, cold start, scalability, ranking of the recommendation, etc. that are related to the CF recommendation system [6]. Buyers do not like to spend too much time on rating items online. Hence rating data is usually sparse, which in turn reduces the recommendation quality. As new users have not made any explicit or implicit ratings, therefore it is difficult to find similarity and hence provide recommendations. This is known as cold start problem.

This paper proposes the use of Jaccard Similarity [11]. Jaccard Similarity for item-based recommendation calculates the similarity between two books by taking the number of users who have rated both books in the numerator and the number of users who have rated either of the two books in the denominator. This does not consider the absolute ratings rather it considers number of items rated. Two items will be more similar, when they have been rated by more common users. Jaccard Similarity does not consider the actual rating given by the user to a book, but only goes by the number of users who have rated the books. Incorporating a weightage to the similarity index based on the rating given by user, overcomes this issue. For the item-based algorithm, Jaccard Similarity is given as

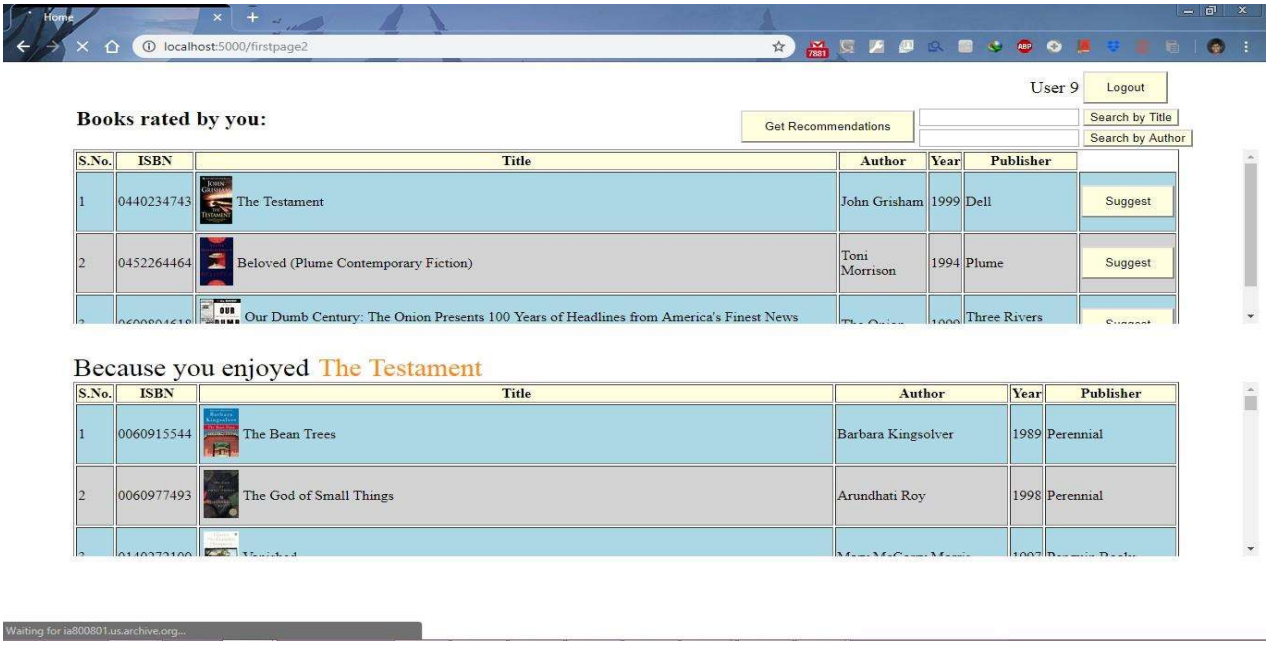
$$jaccard\ sim(A, B) = \frac{|n_A \cap n_B|}{|n_A \cup n_B|}$$

Where n_A is the set of users who have rated item A and n_B is the set of users who have rated item B.

In this approach, the recommendation system will calculate the similarity between books that have been rated by the user already and the books available in the Book Crossing dataset. This dataset was compiled by Cai-Nicolas Ziegler, it contains data of 2,71,379 books, along with 11,49,780 user ratings for these books, given by 2,78,858 users. Datasets are available in CSV (Comma Separated Values) and SQL formats. Since large amount of data is to be handled, SQL database was used for faster and more efficient processing.

Login module for the proposed system:

Once the similarities have been computed for all the books, they are sorted and the books most similar to the books rated by the user are recommended to him. Recommendations to the user can also be based on a particular book chosen by the user. In this case, the similarity calculation will be done only for this book and the most similar books will be recommended to him. Books in the entire dataset can be searched by author or by title, and the user can give ratings to these books for more accurate results.



The interaction between the application server and the user is shown in fig 1. Fig. 2 displays the architecture of the proposed system

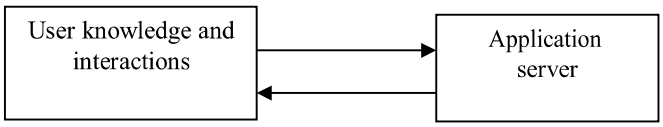


Fig1: Block diagram

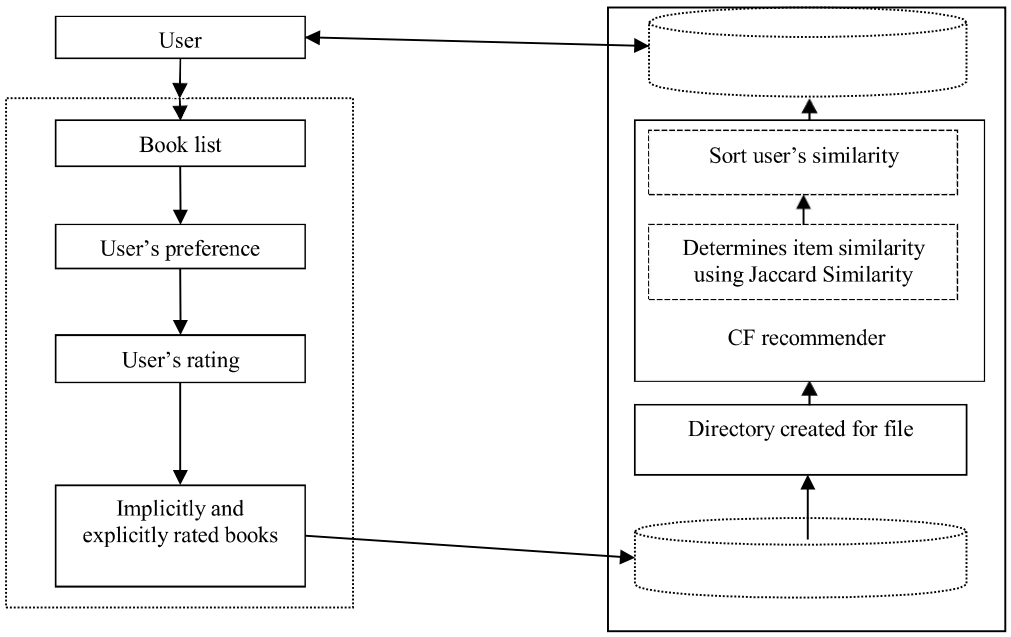
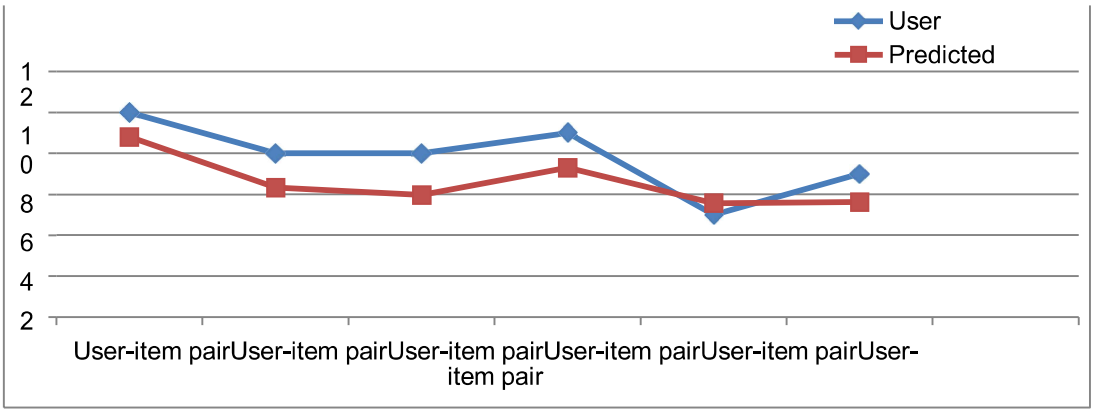


Fig. 2: Architecture of proposed system

4. EXPERIMENTAL RESULTS

This paper used Root Mean Square Error (RMSE) statistical accuracy metrics to evaluate the online book recommendation system. RMSE measures the variation between the predicted value and actual user rating. The predicted rating is calculated as follows:

$$P_{u,i} = \frac{\sum_{all\ similar\ items,N}(s_{i,N} * R_{u,N})}{\sum_{all\ similar\ items,N}(|s_{i,N}|)}$$



Hence, smaller the value of RMSE, better the recommendation system. The computed value of RMSE for this system is 1.504.

Fig. 3: Comparison of Ratings and Predictions for user-item pairs

5. CONCLUSION

Even with the adaptation of a fitting algorithm for recommendation, the RS faces an obstacle because of large quantity of data that needs to be handled. According to the experimental results, the proposed algorithm with compact dataset was more accurate than existing algorithms with full datasets. In addition, JS uses the number of common users as a basis for measuring similarity, rather than the absolute ratings, as used by most existing algorithms, which gives a more accurate result.

6. FUTURE WORK

The recommendation system proposed here takes the number of users who have rated the books into account, without factoring in the absolute rating. Due to this, a recommendation might arise from a book that a user has given low rating to, in which case a book might be recommended from a genre that the user dislikes.

This recommendation system relies on the ratings given by users. So, trust is a major issue, like whether the feedback and rating given by the user is genuine or not. This recommendation system does not solve the trust issue.

Therefore future research should focus on resolving both these issues.

7. REFERENCES

[1] Elgohary, A., Nomir, H., Sabek, I., Samir, M., Badawy, M. and Yousri, N.A. (2010). Wiki-rec: A semantic-based recommendation system using Wikipedia as ontology. In 10th International Conference on Intelligent Systems Design and Applications (ISDA).

[2] Oku, K., Nakajima, S., Miyazaki, J. and Uemura, S. (2006). Context-aware SVM for context- dependent information recommendation. In MDM'06 proceedings of International Conference on Mobile Data Management.

[3] Ghani, R. and Fano, A. (2002). Building recommender systems using a knowledge base of product semantics. In proceedings of 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems.

[4] Miyahara, K. and Pazzani, M.J. (2000) Collaborative filtering with the simple Bayesian classifier. In proceedings of Pacific Rim International Conference on Artificial Intelligence.

[5] Asanov D. (2011) Algorithms and Methods in Recommender Systems. Berlin Institute of Technology Berlin, Germany

[6] Lakshmi, S.S. and Lakshmi, T.A. (2014). Recommendation Systems: Issues and challenges. (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4), 2014, 5771-5772

[7] 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

[8] Kurmashov, N., Konstantin, L., Nussipbekov, A. (2015). Online book recommendation System. Proceedings of Twelve International Conference on Electronics Computer and Computation (ICECC)

[9] Mathew, P., Kuriakose, B. And Hegde, V. (2016). Book Recommendation System through content based and collaborative filtering method. Proceedings of International Conference on Data Mining and Advanced Computing (SAPIENCE)

[10] Parvitikar, S. and Dr. Joshi, B. (2015). Online book recommendation system by using collaborative filtering and association mining. Proceedings of IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)

[11] Ayub, M., Ghazanfar, M.A., Maqsood, M. and Saleem, A. (2018). A Jaccard base similarity measure to improve performance of CF based recommendation system. Proceedings of International Conference on Information Networking (ICOIN)

[12] Gogna, A., Majumdar, A. (2015). A Comprehensive Recommender System Model: Improving Accuracy for Both Warm and Cold Start Users. IEEE Access Vol. 3, 2803-2813, 2015

[13] 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

[14] Choi, S.H., Jeong, Y.S. and Jeong, M.K. (2010). A Hybrid Recommendation Method with Reduced Data for Large-Scale Application. IEEE Transactions on Systems, Man

and Cybernetics-Part C: Applications and Reviews, Vol. 40, No.5, September 2010.

[15] Liu, Q., Chen, E., Xiong, H., Ding, C.H.Q. and Chen, J. (2012). Enhancing Collaborative Filtering by User Interest Expansion via Personalised Ranking. IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 42, No.1, February 2012.

[16] Feng, J., Fengs, X., Zhang, N. and Peng, J. (2018). An improved collaborative filtering method based on similarity. PLOS ONE 13 (9): e0204003, September 2018.

[17] Aggarwal, C.C. (2016). Recommendation System: The Textbook. XXI, 29 p. ISBN 978-3-319- 29657-9

[18] NHK K. ISMAIL*,"Estimation Of Reliability Of D Flip-Flops Using Mc Analysis", Journal of VLSI Circuits And Systems 1 (01), 10-12,2019.

[19] MN BORHAN,"Design Of The High Speed And Reliable Source Coupled Logic Multiplexer",Journal of VLSI Circuits And Systems 1 (01), 18-22,2019.

PUBLISHED PAPER

Online Book Recommender System Using Collaborative Filtering Algorithm

Mr. S. Joshua Johnson[1], Katta Vasantha Rayudu[2], Katta Sunil Kumar[3], Gurugubelli Sai Kumar[4]

[1]Assistant professor, Department of Computer Science and Engineering, Anil Neerukonda Institute of Technology and Sciences, Visakhapatnam-531162, India; [2,3,4,5]Student, Department of Computer Science and Engineering, Anil Neerukonda Institute of Technology and Sciences, Visakhapatnam-531162, India.

ABSTRACT

A recommender system is a kind of filtering system that predicts a user's rating of an item. Recommender systems recommend items to users by filtering through a large database of information using a ranked list of predicted ratings of items. When selecting a book to read, individuals read and rely on the book ratings and reviews that previous users have written. In this paper, Hybrid Recommender system is used in which Collaborative Filtering and Content-Based Filtering techniques are used. The datasets are used which are downloaded from the Goodreads Website which contains the features of users and books. The content- based filtering system uses only book features for recommendation whereas the Collaborative Filtering system uses Books and Users features and the results are displayed in the Front end.

Keywords: Book Recommender System; Truncated-SVD; Clustering; Root Mean Square Error

1. INTRODUCTION

A Hybrid recommender system is used to boost our recommendations. The technique used by recommender systems is Collaborative filtering. This technique filters information by collecting data from other users. Collaborative filtering systems apply the similarity index-based technique. The ratings of those items by the users who have rated both items determine the similarity of the items. The similarity of users is determined by the similarity of the ratings given by the users to an item. Content-based filtering uses the description of the items and gives recommendations which are similar to the description of the items. With these two filtering systems, books are recommended not only based on the behaviour of user but also with the content of the books. So, our recommendation system recommends books to the new users also. In this paper, we used two methods i.e., K-means and Gaussian mixture for clustering the users. Root Mean square Error is used to measure the error between the absolute values and obtained values. That RMSE value is used to find the fundamental accuracy.

2. LITERATURE SURVEY

Most researchers used Pearson's Correlation Coefficient function to calculate similarity among book ratings to recommend books.

Avi Rana and K. Deeba, et.al. (2019) [1] proposed a paper "Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)". In this paper, the author used CF with Jaccard similarity to get more accurate recommendations because general CF difficulties are scalability, sparsity, and cold start. So to overcome these difficulties, they used CF with Jaccard Similarity. JS is based on pair of books index which is a ratio of common users who have rated both books divided by the sum of users who have rated books individually. Books with a high JS index are highly recommended.

G. Naveen Kishore, et.al. (2019) [2] proposed a paper "Online Book Recommendation System". The dataset used in this paper was taken from the website "good books-10k dataset" which contains ten thousand unique books. Features are book_id, user_id, and rating. In this paper, the author adopted a Keras deep learning framework model to create neural network embedding.

Uko E Okon, et.al. (2018) [3] proposed a paper "An Improved Online Book Recommender System using Collaborative Filtering Algorithm". The authors designed and developed a recommendation model by using a quick sort algorithm, collaborative filtering, and object-oriented analysis and design methodology (OOADM). This system produces an accuracy of 90-95%.

Jinny Cho, et.al. (2016) [4] proposed a paper "Book Recommendation System". In this paper, the author uses two approach methods which are Content-based (CB) and Collaborative Filtering (CF). They used two algorithms as UV-Decomposition and K Nearest Neighbors (KNN). They obtained a result with an accuracy of 85%.

Sushma Rjpurkar, et.al. (2015) [5] proposed a paper "Book Recommendation System". In this paper, the author used Associative Rule Mining to find association and correlation relationships among a dataset of items. They used CB and CF approaches to build a system.

Abhay E. Patil, et.al. (2019) [6] proposed a paper "Online Book Recommendation System using Association Rule Mining and Collaborative Filtering". The author detected recurrently occurring patterns, correlations and uses various databases such as relational databases, transactional databases to form associations. They used two approaches i.e., User-based and Item-based Collaborative Filtering, and used the Pearson correlation coefficient to find similarity between the items.

Suhas Patil, et.al. (2016) [7] proposed a paper "A Proposed Hybrid Book Recommender System". In this paper, the author used techniques such as Demographic, Collaborative Filtering, Content-based to build a system and rarely they combined the features of these techniques to make a better recommendation system.

Ankit Khera, et.al. (2008) [8] proposed a paper "Online Recommendation System". In this paper, the author used the User similarity matrix, Vogoo which is PHP-based CF, Fuggy logic, Context Engine for building recommendation systems. Pearson Correlation is a similarity function in this paper.

Anagha Vaidya and Dr. Subhash Shinde, et.al. (2019) [9] proposed a paper "Hybrid Book Recommendation System". In this paper, the author used techniques such as

Collaborative Filtering etc. and used the Pearson correlation coefficient. It was published in International Research Journal of Engineering and Technology (IRJET).

Dhirman Sarma,Tanni Mittra and Mohammad Shahadat Hossain, et.al. (2019) [10] proposed a paper "Personalized Book Recommendation System using Machine Learning Algorithm". It was published in The Science and Information Organization vol.12.

3. PROPOSED SYSTEM

This paper is divided into five sections. In Section 3.1, Dataset was collected from Good Reads Website in which three datasets are present i.e. Books Dataset, Ratings Dataset, Users Dataset. In Section 3.2, Datasets were pre-processed to make suitable for developing the Recommendation system. Truncated-SVD is used to reduce the features of the dataset. In Section 3.3, Content Based Filtering System is developed in which book description is taken as an input. In Section 3.4, Data splitting is done in which training dataset and testing dataset are divided into 80:20 ratio.

In Section 3.5, Collaborative Filtering System is developed by building a model using K- Means Algorithm over Gaussian Mixture after comparing with Silhouette scores.

An architecture diagram gives the high-level overview of major system components and relationships among them. It represents the flow of execution of paper. The architecture diagram of our recommendation system is shown in fig 1.

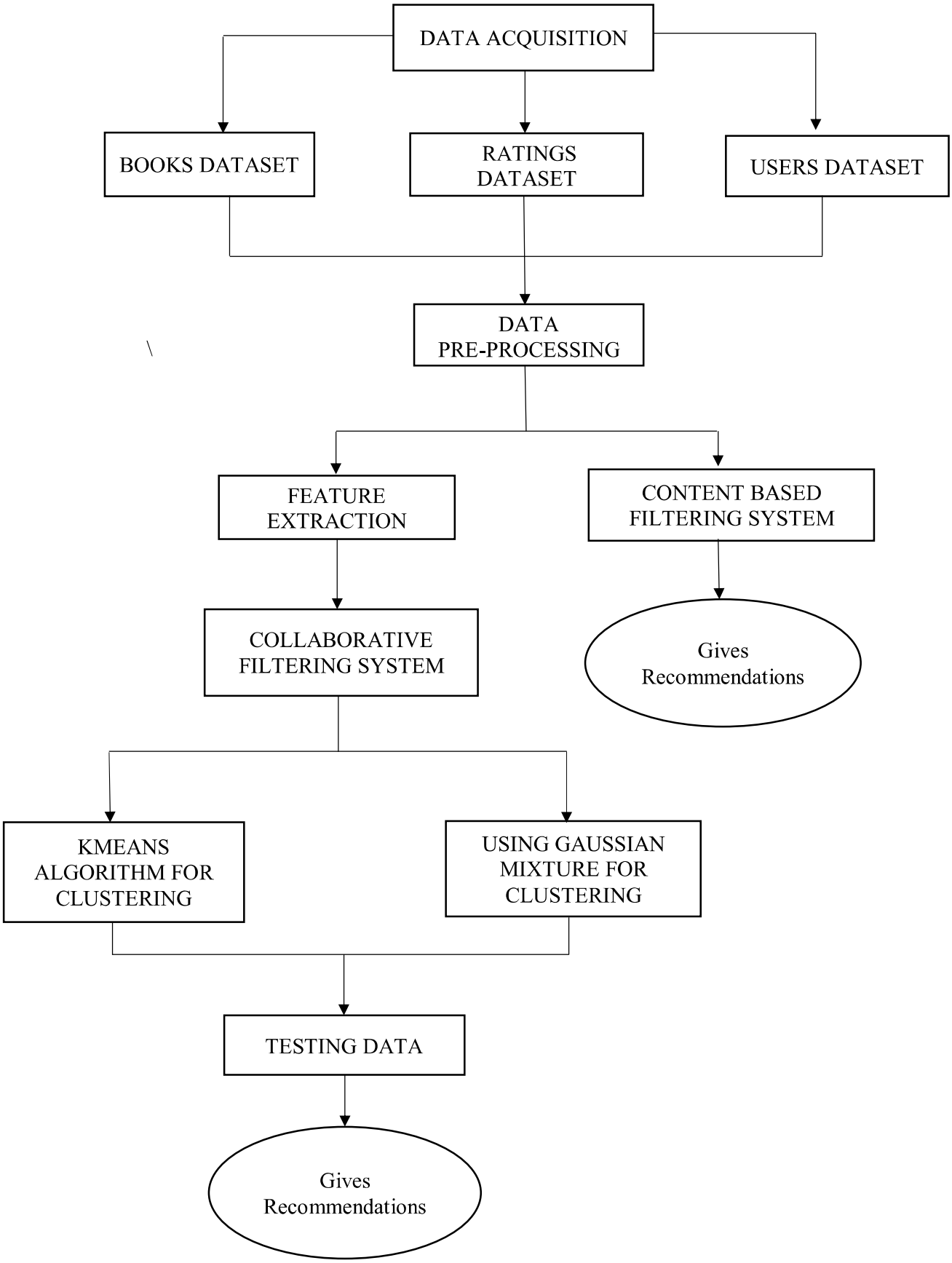


Fig.1. System Architecture

3.1 Datase
3.1 Dataset

The first step is to find the Dataset for building our system. We have to identify the dataset which gives efficient results for our system. In this step, we used to collect different datasets that are available in data repositories, and the data might be raw data that is stored in the form of zip files and databases. Important features to identify the data are quality and quantity. Results are more accurate if we have more data points in the database. We collected the three datasets which contain features required for the system. The three datasets are the Books dataset, Users Dataset, and Ratings Dataset. The Books dataset contains features namely book_authors which gives the name of the author, book_desc which gives a description of books, book_format which tells about the packing of book, book_rating which shows an average rating of the book, book_rating_count which gives a total number of rating count, book_title which shows the title of the book, genres which shows genres of book, image_url which contains URL of book cover page and the Rating Dataset contains features namely User-ID, Book-ID, Rating.

3.2 Pre-Processing data

In this step, the main aim is to understand the features of data that the author has to work with. In this paper, we had applied Truncated-SVD which is available for sparse matrices to reduce the dimensionality. By using the Truncated-SVD, features of the dataset are reduced. The originally collected data contains noises, missing values, and duplicate values which cannot be directly used for machine learning algorithms. Data pre-processing is the method of cleaning the data, removing duplicates, and filling the null values with the average of that attribute. In that way, the Final dataset was prepared for building the model which gives an efficient performance. We split the data into 80:20 i.e. 80% of the dataset is used to train the developed model and 20% of the data are used to test the model.

3.2Content Based filtering System

A Content-Based filtering system recommends items that are similar to the content of the item. This System uses the description of the items and gives the recommendations that are similar to the description. We used Cosine similarity as a similarity function for this system. The Item-Content Matrix which describes the attributes of the features is taken as an input. Based on the angle between the vectors, Cosine similarity is calculated. We improve the quality of the content-based system by normalizing and tuning the attributes with the use of the TF-IDF vectorizer. TF (Term Frequency) is termed as a word frequency in a document. IDF (Inverse Document Frequency) is universe document frequency.

3.3Training data

In Data processing, we split the data into training data and testing data so that we can train and test the model. By this method, we can get more performance of our machine learning model. If the total dataset is used to train the model and then we have to test the model with a completely different dataset. So, we would get more difficulties to understand the correlation between the models with different sets which decreases the performance. That's why we have to use the same data for training and testing to get better performance. Training is very essential to understand various features and patterns.

3.4Collaborative Filtering System

Collaborative Filtering System collected and analysed a large amount of user-based information and predicts the user's preference based on their similarity with other users. The User-Based Collaborative Filtering approach is finding similar users to the current user U based on ratings for given items and then predicts the rating for an Item I by calculating the ratings given by the similar users for that Item I. Model-Based Collaborative Filtering uses Dimensionality reduction by constructing a user-item matrix by taking users as rows, items as columns, and rating of user for an item as values. In this paper, we used Truncated-SVD for Dimensionality reduction.

3.4.1 K-Means Clustering

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters in such a manner that each dataset belongs to only one group that has similar features. Here K defines the number of pre-defined clusters. We have to associate each cluster with a centroid in this algorithm. The sum of distances between the data point and their corresponding clusters should be minimized. The unlabelled dataset is taken as an input and the dataset into k-number of clusters is divided, and the process is repeated until it does not find the best clusters. We have to predetermine the k value in this algorithm. Elbow method is used to find the value of k which decides the number of clusters. This method uses the Within Cluster Sum of Squares (WCSS) value that defines the total variations within a cluster.

The Formula for calculating the value of WCSS for n clusters is given by eq-1.

$$WCSS = \sum_{Pi \text{ in Cluster1distance}} (P_i\ C_1)^2 + \sum_{Pi \text{ in Cluster2distance}} (P_i\ C_2)^2 + \ldots\ldots + \sum_{Pi \text{ in Cluster n distance}} (P_i\ C_n)^2 \quad \text{eq-1}$$

Algorithm steps:

Step-1: Select the number K which gives the number of clusters.

Step-2: Select random K number of points or centroids.

Step-3: Each data point to their nearest centroid should be assigned, which forms the predefined K clusters.

Step-4: Calculate the variance and place a new centroid for each cluster.

Step-5: We have to repeat the step-3, each data-point to the new closest centroid of each cluster should be reassigned.

Step-6: If reassignment happens, then go to step-4 or else go to step-7.

Step-7: Stop.

3.4.2 Gaussian Mixture

Gaussian Mixture models are powerful clustering algorithms. It assumes that there are a certain number of Gaussian distributions where each distribution represents a cluster. This model groups the data points together into a single distribution. These models used the soft clustering technique for assigning data points to Gaussian distributions.
In a one dimensional space, the probability density function of a Gaussian distribution (univariate) is given by eq-2.

$P(x \mid \mu, \sigma 2) = N(x; \mu, \sigma 2) = 1/Z\ [\exp(-(x-\mu)\ 2/\ 2\sigma 2)]$ eq-2

Where Z is the normalization constant i.e., $Z = \sqrt{2\pi\sigma 2}$, μ is the mean i.e., $\mu = E[x]$, and $\sigma 2$ is the variance of the distribution i.e., $\sigma 2 = var[x]$.

In a multidimensional space, the probability density function of a Gaussian distribution (multivariate) is given by eq-3.

$P(x \mid \mu, \Sigma) = N(x; \mu, \Sigma) = 1/Z\ [\exp(-\ 1/2\ (x-\mu)T\Sigma-1(x-\mu))]$…… eq-3

Where X is the input vector, μ is the 2D mean vector, and Σ is the 2×2 covariance matrix. Thus, we would have K (number of clusters) Gaussian distributions.

4 PERFORMANCE ANALYSIS AND RESULTS

In this step, we analyse the performance of a model and checks the results. While clustering, we have applied K-Means and Gaussian mixture to cluster the users. The model which gave the best silhouette score would be used for building the system. Silhouette score is used to test the efficiency of clustering. Positive value of Silhouette score indicates good clustering whereas negative value indicates bad clustering. First, we built the model by using the K-means method and then we fit and predict the model with the training dataset. Then, we calculate the silhouette score with the developed model.

```
Clusterer_KMeans=KMeans(n_clusters=7)
.fit(book_ratings_training)

Preds_KMeans=Clusterer_KMeans.predict
(book_ratings_training)

Kmeans_score=silhouette_score(book_rati
ngs_training,preds_KMeans)

Print(Kmeans_score)
```

Now, we built the model by using Gaussian mixture and then we fit and predict the model using same training dataset. Then, we calculate the silhouette score with the developed model.

```
Clusterer_GMM=GaussianMixture(n_com
ponents=7).fit(book_ratings_training)

Preds_GMM=Clusterer_GMM.predict(boo
k_ratings_training)

GMM_score=silhouette_score(book_ratin
gs_training,preds_GMM)

Print (GMM_score)
```

	K-Means	Gaussian mixture
Silhouette Score	0.0433968332584411	0.016778872313688933

Table-1: Silhouette scores of two models

After checking the two scores, we consider the model using K-Means method with cluster count of 7 because of higher score. Then, this model is used to cluster the users.

The difference between average mean ratings of books per test user and average mean rating of books for cluster's favourite is calculated. The average mean rating is calculated by: Average mean rating = Sum(ratings)/len(ratings)

Mean rating for 10 random books	3.8876949740034736
Mean rating for 10 books of cluster's favourites	4.3735008665511135

Table-2: Comparing Mean ratings

Difference between ratings: 0.48580589254763984.

RMSE (Root mean square error) measures the error caused by the deviation between the sample values and predicted values by the model. The accuracy calculated by taking the RMSE value is known as Fundamental vertical accuracy whose value is computed by 1.96 * RMSE.

RMSE	0.5957791790493179
Accuracy	1.167727190936663

Table-3: RMSE and Accuracy values

5 CONCLUSION

In this paper, we recommended the books for a user using K-Means model clustering which is an unsupervised machine learning algorithm. We compared the two methods for clustering and identified the best model by calculating silhouette score. We used the datasets which are downloaded from the Good reads website. We implemented the functionalities in the system according to the requirements after understanding all the modules of the system.

6 REFERENCES

[1] Avi Rana and K. Deeba et.al, "Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)" in IOP ebooks 1362, 2019.

[2]G. Naveen Kishore, V. Dhiraj, Sk Hasane Ahammad, Sivaramireddy Gudise, Balaji Kummaraa and Likhita Ravuru Akkala, "Online Book Recommendation System" International Journal of Scientific & Technology Research vol.8, issue 12, Dec 2019.

[3] Uko E Okon, B O Eke and P O Asaga, "An Improved Online Book Recommender System using Collaborative Filtering Algorithm", International Journal of

Computer Applications vol.179-Number 46, 2018.

[4] Jinny Cho, Ryan Gorey, Sofia Serrano, Shatian Wang, JordiKai Watanabe-Inouye, "Book Recommendation System" Winter 2016.

[5] Ms. Sushma Rjpurkar, Ms. Darshana Bhatt and Ms. Pooja Malhotra, "Book Recommendation System" International Journal for Innovative Research in Science & Technology vol.1, issue 11, April 2015.

[6] Abhay E. Patil, Simran Patil, Karanjit Singh, Parth Saraiya and Aayusha Sheregar, "Online Book Recommendation System using Association Rule Mining and Collaborative Filtering" International Journal of Computer Science and Mobile Computing vol.8, April 2019.

[7] Suhas Patil and Dr. Varsha Nandeo, "A Proposed Hybrid Book Recommender System" International Journal of Computer Applications vol.6 – No.6,Nov – Dec 2016.

[8] Ankit Khera, "Online Recommendation System" SJSU ScholarWorks, Masters Theorem and Graduate Research, Master's Projects, 2008.

[9] Anagha Vaidya and Dr. Subhash Shinde, "Hybrid Book Recommendation System" International Research Journal of Engineering and Technology (IRJET), July 2019.

[10] Dhirman Sarma, Tanni Mittra and Mohammad Shahadat Hossain, "Personalized Book Recommendation System using Machine Learning Algorithm" The Science and Information Organization vol.12,2019.