IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Movie recommendation system based on Collaborative Filtering

[1]B. Venkataramana, [2]Guguloth Ravinder, [3]K. Helini

[1,2,3] Assistant Professor

[1,2,3] Department of Computer Science and Engineering,
[1,2] Holy Mary Institute of Technology & Science, Bogaram, Hyderabad, Telangana, India.
[3]Vignan's Institute of Management and Technology for Women, Bogaram, Hyderabad, Telangana, India.

Abstract: Today's web and app users request modified experiences. They anticipate the apps, news sites, social networks they engage with to evoke who they are and what they're fascinated in and make related, adjusted, and accurate commendations for new content and new goods based on their earlier deeds. This can be done using Recommended Systems in Machine Learning. In this paper we use Recommender System to recommend movies based on his previous ratings on movie he came across.

***Index Terms* - Recommendation system, Collaborative filtering, Movie, Data Preprocessing, Visualizations, Rating, Movies**

I. INTRODUCTION

As the business needs are accelerating, there is an increased dependence on extracting meaningful information from humongous amount of raw data to drive business solutions. The same is true for digital recommendation systems which are becoming a norm for consumer industries such as books, music, clothing, movies, news articles, places, utilities, etc. These systems collect information from the users to improve the future suggestions.

With the eruption of big data, practical recommendation schemes are now very important in various fields, including e-commerce, social networks, and a number of web-based services. Nowadays, there exist many personalized movie recommendation schemes utilizing publicly available movie datasets (e.g., MovieLens and Netflix), and returning improved performance metrics (e.g., Root-Mean-Square Error (RMSE)). However, two fundamental issues faced by movie recommendation systems are still neglected: first, scalability, and second, practical usage feedback and verification based on real implementation. In particular, Collaborative Filtering (CF) is one of the major prevailing techniques for implementing recommendation systems. However, traditional CF schemes suffer from a time complexity problem, which makes them bad candidates for real-world recommendation systems. Collaborative Filtering is the most common technique used when it comes to building intelligent recommender systems that can learn to give better recommendations as more information about users is collected.

II. TYPES OF RECOMMENDATION SYSTEM

A Recommendation System is a software tool designed to make and deliver suggestions for things or content a user would like to purchase. Using machine learning techniques and various data about individual products and individual users, the system creates an advanced net of complex connections between those products and those people. These are a collection of algorithms used to recommend items to users based on information taken from the user. These systems have become ubiquitous, and can be commonly seen in online stores, movies databases and job finders. There are 3 types of recommendation systems

1. Popularity based recommendation engine
2. Content based recommendation engine
3. Collaborative filtering based recommendation engine

Popularity based recommendation engine:

Pearson correlation is invariant to scaling, i.e. multiplying all elements by a nonzero constant or adding any constant to all elements. For example, if you have two vectors X and Y, then, pearson (X, Y) == pearson(X, 2 * Y + 3). This is a pretty important property in recommendation systems because for example two users might rate two series of items totally different in terms of absolute rates, but they would be similar users (i.e. with similar ideas) with similar rates in various scales.

Content based recommendation engine:

Content based recommendation engine takes in a movie that a user currently likes as input. Then it analyzes the contents of the movie to find out other movies which have similar content. Then it ranks similar movies according to their similarity scores and recommends the most relevant movies to the user.

Collaborative filtering based recommendation engine:

Collaborative filtering based recommendation engine first tries to find similar users based on their activities and preferences. Now, between these users (say, A and B) if user A has seen a movie that user B has not seen yet, then that movie gets recommended to user B and vice-versa. In other words, the recommend-dations get filtered based on the collaboration between similar user's preferences. One typical application of this algorithm can be seen in the Amazon e-commerce platform, where you get to see the "Customers who viewed this item also viewed" and "Customers who bought this item also bought" list.

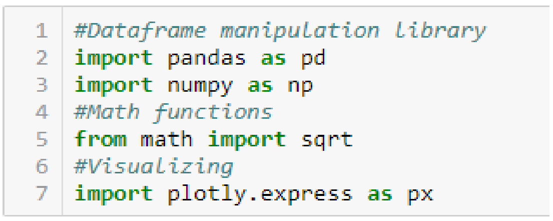
III. METHODOLOGY

In this section, we will invoke the data, preprocess the data and visualize the data.

3.1 Data and its features

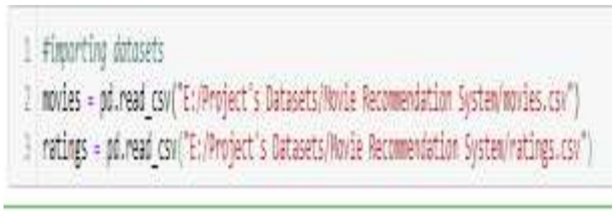
The datasets used in this system is taken from "GroupLens" namely "movie.csv" and "rating.csv". The movie.csv contains features like movieid, movie name and its genre. The rating dataset contains features like userid, movieid and rating given by the user to that movie.

First thing to do is to import necessary libraries as shown in Fig 1. Then we should invoke data from the datasets using pandas as shown in Fig 2. The features of the datasets are presented in Fig 3 and Fig 4.



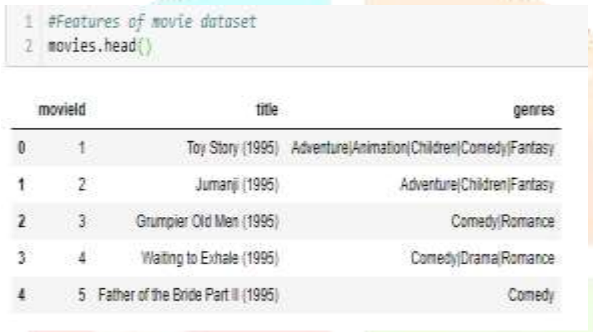
```
#Dataframe manipulation library
import pandas as pd
import numpy as np
#Math functions
from math import sqrt
#Visualizing
import plotly.express as px
```

Fig 1: Importing libraries



```
#importing datasets
movies = pd.read_csv("E:/Project's Datasets/Movie Recomendation System/movies.csv")
ratings = pd.read_csv("E:/Project's Datasets/Movie Recomendation System/ratings.csv")
```

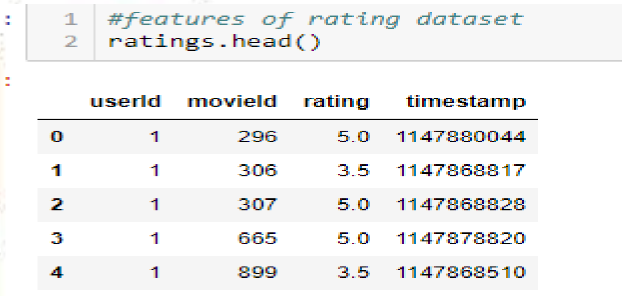
Fig 2: Invoking data



```
#Features of movie dataset
movies.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

Fig 3: Features of movie dataset



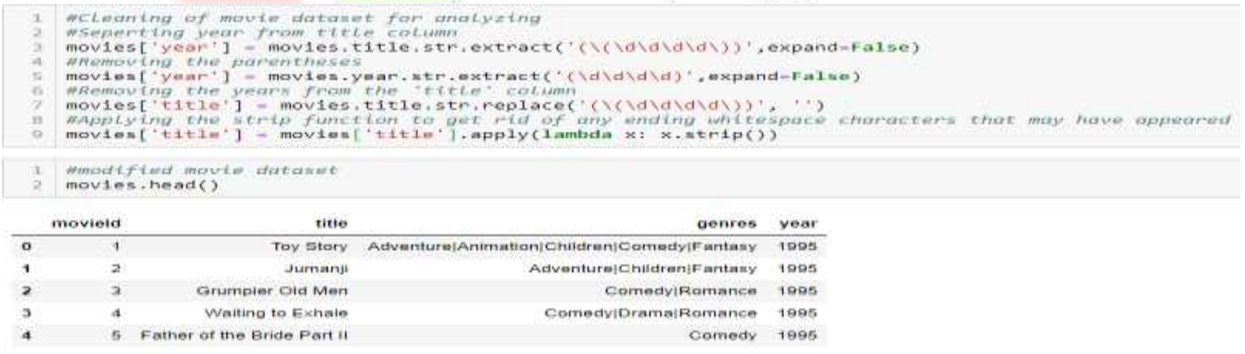
```
#features of rating dataset
ratings.head()
```

	userId	movieId	rating	timestamp
0	1	296	5.0	1147880044
1	1	306	3.5	1147868817
2	1	307	5.0	1147868828
3	1	665	5.0	1147878820
4	1	899	3.5	1147868510

Fig 4: Features of rating dataset

3.2 Data Prepossessing

Data preparation is the first step in data analytics projects and can include many discrete tasks such as loading data or data ingestion, data fusion, data cleaning, data augmentation, and data delivery. In movie data we can remove year from the title feature and add a column of it to the dataset. This is shown in Fig 5.



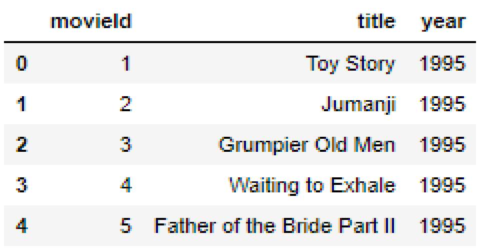
```
#Cleaning of movie dataset for analyzing
#Seperting year from title column
movies['year'] = movies.title.str.extract('(\(\d\d\d\d\))',expand=False)
#Removing the parentheses
movies['year'] = movies.year.str.extract('(\d\d\d\d)',expand=False)
#Removing the years from the 'title' column
movies['title'] = movies.title.str.replace('(\(\d\d\d\d\))', '')
#Applying the strip function to get rid of any ending whitespace characters that may have appeared
movies['title'] = movies['title'].apply(lambda x: x.strip())
```

```
#modified movie dataset
movies.head()
```

	movieId	title	genres	year
0	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995
1	2	Jumanji	Adventure|Children|Fantasy	1995
2	3	Grumpier Old Men	Comedy|Romance	1995
3	4	Waiting to Exhale	Comedy|Drama|Romance	1995
4	5	Father of the Bride Part II	Comedy	1995

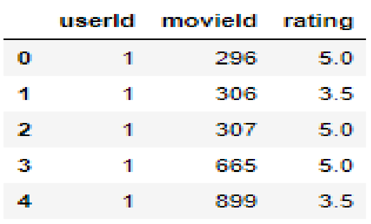
Fig 5: Preprocessing of movie dataset

In collaborative recommended system there is no need of genre so we can drop genre column from the movie dataset. After dropping the genre column from the movie dataset the final dataset in shown in Fig 6. Now coming to rating dataset we can drop a feature called 'timestamp' as it is not used in the system. The final rating dataset is shown in Fig 7.



	movieId	title	year
0	1	Toy Story	1995
1	2	Jumanji	1995
2	3	Grumpier Old Men	1995
3	4	Waiting to Exhale	1995
4	5	Father of the Bride Part II	1995

Fig 6: Final movie dataset



	userId	movieId	rating
0	1	296	5.0
1	1	306	3.5
2	1	307	5.0
3	1	665	5.0
4	1	899	3.5

Fig 7: Final rating dataset.

3.3 Data Visualization

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. The statistical analysis of no .of movies are rated for a rate can be show by using histogram as shown in Fig 8.

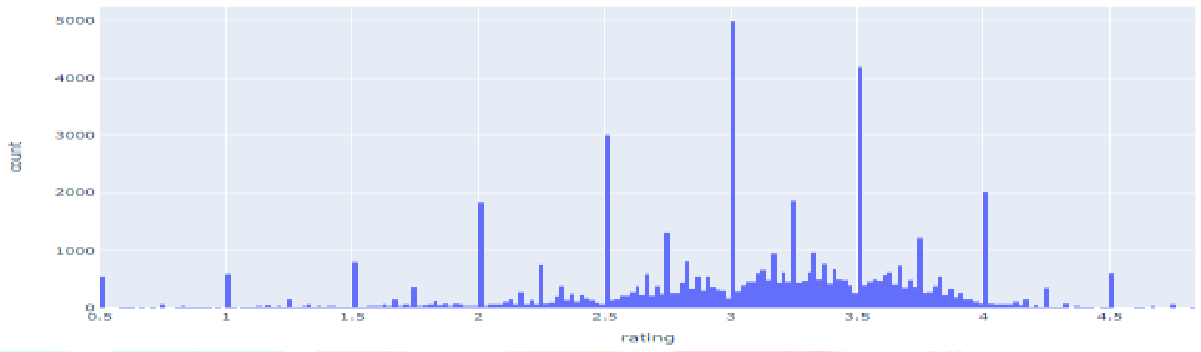


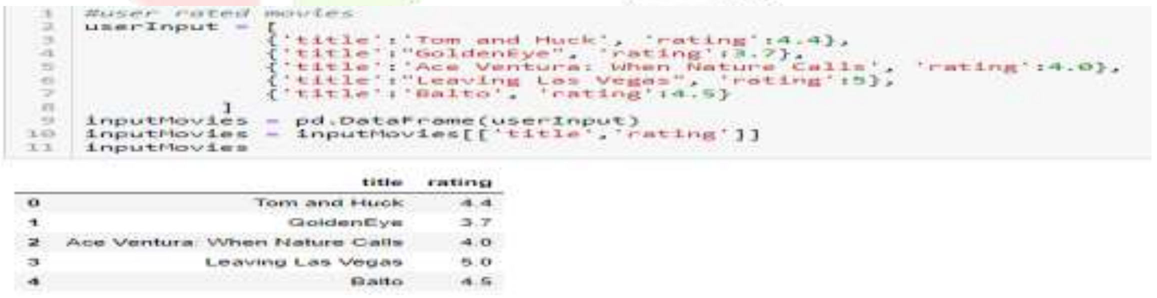
Fig 8: Statistical analysis of rating.

IV. COLLABORATIVE RECOMMENDATION SYSTEM

Now let’s start working on Collaborative recommendation System which is also known as User-User recommendation system. This technique uses other users to recommend items to the input user. It attempts to find users that have similar preferences and opinions as the input and then recommends items that they have liked to the input. There are several methods of finding similar users and the one we will be using here is going to be based on the Pearson Correlation Function. The process of creating a Collaborative system is as follows:

- Select a user with the movies the user has watched
- Based on his rating to movies, find the top X neighbors
- Get the watched movie record of the user for each neighbor.
- Calculate a similarity score using some formula
- Recommend the items with the highest score

Let's begin by creating an input user to recommend movies and is shown in the following Fig 9.



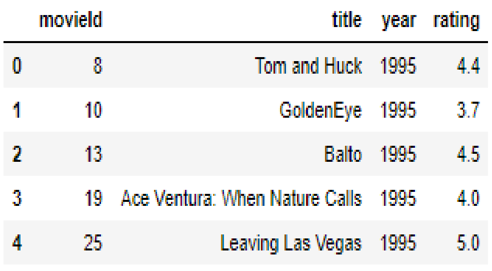
```
#user rated movies
userInput = [
                        {'title':'Tom and Huck', 'rating':4.4},
                        {'title':"GoldenEye", 'rating':3.7},
                        {'title':'Ace Ventura: When Nature Calls', 'rating':4.0},
                        {'title':"Leaving Las Vegas", 'rating':5},
                        {'title':'Balto', 'rating':4.5}
            ]
inputMovies = pd.DataFrame(userInput)
inputMovies = inputMovies[['title','rating']]
inputMovies
```

	title	rating
0	Tom and Huck	4.4
1	GoldenEye	3.7
2	Ace Ventura: When Nature Calls	4.0
3	Leaving Las Vegas	5.0
4	Balto	4.5

Fig 9: User’s data

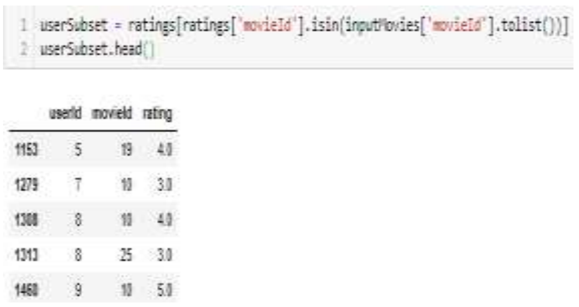
With the input complete, let's extract the input movies ID's from the movies Dataframe and add them into it. We can achieve this by first filtering out the rows that contain the input movies' title and then merging this subset with the input Dataframe. We also drop unnecessary columns for the input to save memory space. The DataFrame after adding the movie id is shown in Fig 10. With the help of specified user’s data we should find the users who gave the rating for the movies of the specified user’s data. This is shown in Fig 11.

The statistical way to represent he no of user rated for each of the movie selected by one user is shown in fig 12. The rating given by 3 users is shown in Fig 13.



	movieId	title	year	rating
0	8	Tom and Huck	1995	4.4
1	10	GoldenEye	1995	3.7
2	13	Balto	1995	4.5
3	19	Ace Ventura: When Nature Calls	1995	4.0
4	25	Leaving Las Vegas	1995	5.0

Fig 10: User’s data with movieid



```
userSubset = ratings[ratings['movieId'].isin(inputMovies['movieId'].tolist())]
userSubset.head()
```

	userId	movieId	rating
1153	5	19	4.0
1279	7	10	3.0
1308	8	10	4.0
1313	8	25	3.0
1460	9	10	5.0

Fig 11: Users how rated for the movies rated by one user

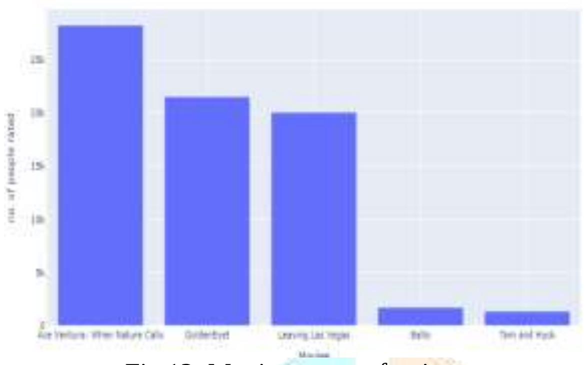
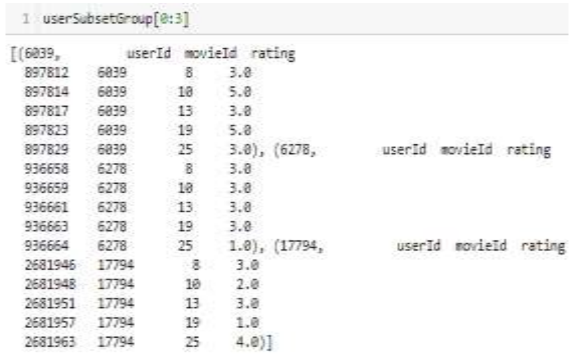


Fig 12: Movies vs no. of ratings



```
userSubsetGroup[0:3]
```

```
[(6039,         userId  movieId  rating
  897812    6039        8     3.0
  897814    6039       10     5.0
  897817    6039       13     3.0
  897823    6039       19     5.0
  897829    6039       25     3.0), (6278,         userId  movieId  rating
  936658    6278        8     3.0
  936659    6278       10     3.0
  936661    6278       13     3.0
  936663    6278       19     3.0
  936664    6278       25     1.0), (17794,          userId  movieId  rating
  2681946   17794        8     3.0
  2681948   17794       10     2.0
  2681951   17794       13     3.0
  2681957   17794       19     1.0
  2681963   17794       25     4.0)]
```

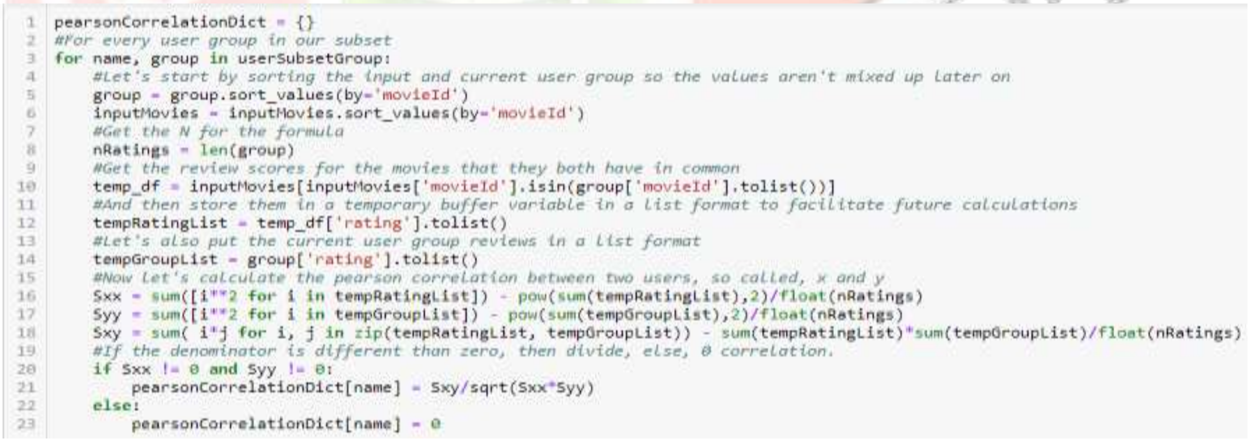
Fig 13: Ratings of 3 users

Next, we are going to compare all users to our specified user and find the one that is most similar we're going to find out how similar each user is to the input through the Pearson Correlation Coefficient. It is used to measure the strength of a linear association between two variables. The formula for finding this coefficient between sets X and Y with N values can be seen in the Fig 14.

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[n\Sigma x^2 - (\Sigma x)^2][n\Sigma y^2 - (\Sigma y)^2]}}$$

Fig 14: Pearson Correlation formulae

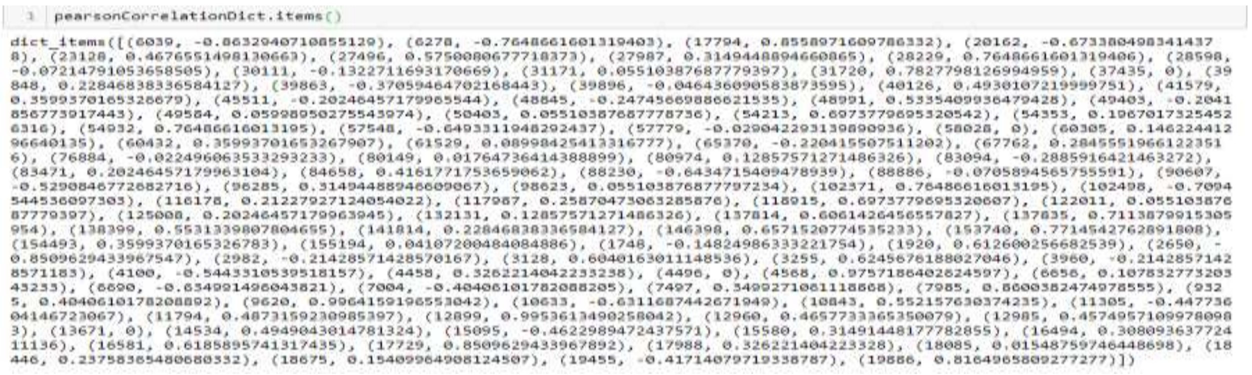
Now, we calculate the Pearson Correlation between input user and subset group, and store it in a dictionary, where the key is the user Id and the value is the coefficient. This process is shown in Fig 15 and the output dictionary is shown in Fig 16.



```
pearsonCorrelationDict = {}
#For every user group in our subset
for name, group in userSubsetGroup:
    #Let's start by sorting the input and current user group so the values aren't mixed up later on
    group = group.sort_values(by='movieId')
    inputMovies = inputMovies.sort_values(by='movieId')
    #Get the N for the formula
    nRatings = len(group)
    #Get the review scores for the movies that they both have in common
    temp_df = inputMovies[inputMovies['movieId'].isin(group['movieId'].tolist())]
    #And then store them in a temporary buffer variable in a list format to facilitate future calculations
    tempRatingList = temp_df['rating'].tolist()
    #Let's also put the current user group reviews in a list format
    tempGroupList = group['rating'].tolist()
    #Now let's calculate the pearson correlation between two users, so called, x and y
    Sxx = sum([i**2 for i in tempRatingList]) - pow(sum(tempRatingList),2)/float(nRatings)
    Syy = sum([i**2 for i in tempGroupList]) - pow(sum(tempGroupList),2)/float(nRatings)
    Sxy = sum( i*j for i, j in zip(tempRatingList, tempGroupList)) - sum(tempRatingList)*sum(tempGroupList)/float(nRatings)
    #If the denominator is different than zero, then divide, else, 0 correlation.
    if Sxx != 0 and Syy != 0:
        pearsonCorrelationDict[name] = Sxy/sqrt(Sxx*Syy)
    else:
        pearsonCorrelationDict[name] = 0
```

Fig 15: Process of Pearson Correlation

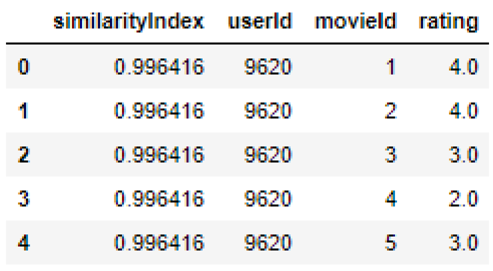
Now we should extract top 50 users who are similar to the specified user. After extracting the users we start recommending movies to the input user. This is done by taking the weighted average of the ratings of the movies using the Pearson Correlation as the weight. But to do this, we first need to get the movies watched by the users and store their correlation in a new column called “_similarityIndex" as shown in Fig 17.



```
pearsonCorrelationDict.items()
```

```
dict_items([(6039, -0.8632940710855129), (6278, -0.7648661601319403), (17794, 0.8558971609786332), (20162, -0.673380498341437
8), (23128, 0.4676551498130663), (27496, 0.5750080677718373), (27987, 0.3149448894660865), (28229, 0.7648661601319406), (28598,
-0.07214791053658505), (30111, -0.1322711693170669), (31171, 0.05510387687779397), (31720, 0.7827798126994959), (37435, 0), (39
848, 0.22846838336584127), (39863, -0.37059464702168443), (39896, -0.046436090583873595), (40126, 0.4930107219999751), (41579,
0.3599370165326679), (45511, -0.20246457179965544), (48845, -0.24745669886621535), (48991, 0.5335409936479428), (49403, -0.2041
856773917443), (49584, 0.05998950275543974), (50403, 0.05510387687778736), (54213, 0.6973779695320542), (54353, 0.1967017325452
6316), (54932, 0.76486616013195), (57548, -0.6493311948292437), (57779, -0.029042293139890936), (58028, 0), (60305, 0.146224412
96640135), (60432, 0.3599370165326790 7), (61529, 0.08998425413316777), (65370, -0.2204155075112O2), (67762, 0.284555196612235l
6), (76884, -0.022496063533293233), (80149, 0.017647364143888899), (80974, 0.12857571271486326), (83094, -0.2885916421463272),
(83471, 0.20246457179963104), (84658, 0.4161771753659062), (88230, -0.6434715409478939), (88886, -0.0705894565755591), (90607,
-0.5290846772682716), (96285, 0.3149448894660 9067), (98623, 0.05510387687779 7234), (102371, 0.76486616013195), (102498, -0.7094
544536097303), (116178, 0.21227927124054022), (117987, 0.25870473063285876), (118915, 0.6973779695320607), (122011, 0.055103876
87779397), (125008, 0.20246457179963945), (132131, 0.12857571271486326), (137814, 0.6061426456557827), (137835, 0.7113879915305
954), (138399, 0.5531339807804655), (141814, 0.22846838336584127), (146398, 0.6571520774535233), (153740, 0.7714542762891808),
(154493, 0.3599370165326783), (155194, 0.04107200484084886), (1748, -0.14824986333221754), (1920, 0.612600256682539), (2650, -
0.8509629433967547), (2982, -0.21428571428570167), (3128, 0.6040163011148536), (3255, 0.6245676188027046), (3960, -0.2142857142
8571183), (4100, -0.5443310539518157), (4458, 0.3262214042233238), (4496, 0), (4568, 0.9757186402624597), (6656, 0.107832773203
43233), (6690, -0.634991496043821), (7004, -0.40406101782088205), (7497, 0.3499271061118868), (7985, 0.8600382474978555), (932
5, 0.4040610178208892), (9620, 0.9964159196553042), (10633, -0.6311687442671949), (10843, 0.552157630374235), (11305, -0.447736
04146723067), (11794, 0.4873159230985397), (12899, 0.9953613490258042), (12960, 0.4657733365350079), (12985, 0.4574957109978098
3), (13671, 0), (14534, 0.4949043014781324), (15095, -0.4622989472437571), (15580, 0.3149144817778285 5), (16494, 0.308093637724
11136), (16581, 0.6185895741317435), (17729, 0.8509629433967892), (17988, 0.326221404223328), (18085, 0.01548759746448698), (18
446, 0.23758385480680332), (18675, 0.15409964908124507), (19455, -0.41714079719338787), (19886, 0.8164965809277277)])
```

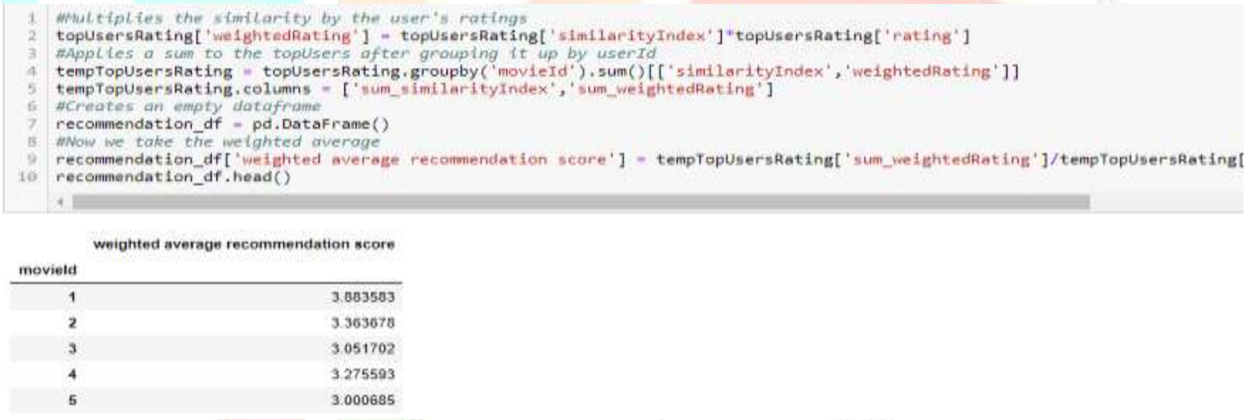
Fig 16: Dictionary containing userid and coefficients



	similarityIndex	userId	movieId	rating
0	0.996416	9620	1	4.0
1	0.996416	9620	2	4.0
2	0.996416	9620	3	3.0
3	0.996416	9620	4	2.0
4	0.996416	9620	5	3.0

Fig 17: Finding similarity index

Now all we need to do is simply multiply the movie rating by its weight (The similarity index), then sum up the new ratings and divide it by the sum of the weights. The entire process of finding the movies according to weighted average recommendation score is shown in Fig 18.



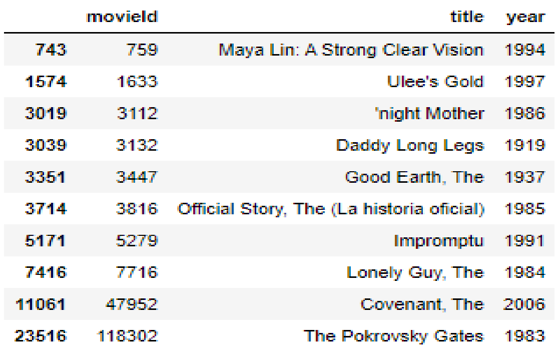
```
#Multiplies the similarity by the user's ratings
topUsersRating['weightedRating'] = topUsersRating['similarityIndex']*topUsersRating['rating']
#Applies a sum to the topUsers after grouping it up by userId
tempTopUsersRating = topUsersRating.groupby('movieId').sum()[['similarityIndex','weightedRating']]
tempTopUsersRating.columns = ['sum_similarityIndex','sum_weightedRating']
#Creates an empty dataframe
recommendation_df = pd.DataFrame()
#Now we take the weighted average
recommendation_df['weighted average recommendation score'] = tempTopUsersRating['sum_weightedRating']/tempTopUsersRating[
recommendation_df.head()
```

	weighted average recommendation score
movieId	
1	3.883583
2	3.363678
3	3.051702
4	3.275593
5	3.000685

Fig 18: process of finding the movies for recommendation.

V. Conclusion

By using Content-Based Recommended System we predicted top 20 movies based on the requirements of the user and the output is shown in Fig 19. This is to conclude that collaboration recommended system in this paper is very helpful to study the priorities of the customer and recommend movies based on similar users who rated the movies of the customer.



	movieId	title	year
743	759	Maya Lin: A Strong Clear Vision	1994
1574	1633	Ulee's Gold	1997
3019	3112	'night Mother	1986
3039	3132	Daddy Long Legs	1919
3351	3447	Good Earth, The	1937
3714	3816	Official Story, The (La historia oficial)	1985
5171	5279	Impromptu	1991
7416	7716	Lonely Guy, The	1984
11061	47952	Covenant, The	2006
23516	118302	The Pokrovsky Gates	1983

Fig 19: Top 10 recommended movies

REFERENCES

[1] Zan Wanga, Xue Yub, Nan Fengb, Zhenhua Wangc (2014), An improved collaborative movie recommendation system using computational intelligence, Journal of Visual Languages & Computing, Volume 25, Issue 6, December 2014, Pages 667-675.

[2] Bushra Alhijawi, Yousef Kilani (2020), A collaborative filtering recommender system using genetic algorithm, Information Processing & Management, Volume 57, Issue 6, November 2020, 102310.

[3] Jamilu Maaruf Musa and Xu Zhihong (2020), Item Based Collaborative Filtering Approach in Movie Recommendation System Using Different Similarity Measures. In Proceedings of the 2020 6th International Conference on Computer and Technology Applications (ICCTA '20). Association for Computing Machinery, New York, NY, USA, 31–34. DOI:https://doi.org/10.1145/3397125.3397148

[4] 2017. A personalised movie recommendation system based on collaborative filtering. Int. J. High Perform. Comput. Netw. 10, 1–2 (January 2017), 54–63.

[5] J. Zhang, Y. Wang, Z. Yuan and Q. Jin,(2020), "Personalized real-time movie recommendation system: Practical prototype and evaluation," in Tsinghua Science and Technology, vol. 25, no. 2, pp. 180-191, April 2020, doi: 10.26599/TST.2018.9010118.

[6] A. V. Dev and A. Mohan,(2016), "Recommendation system for big data applications based on set similarity of user preferences", 2016 International Conference on Next Generation Intelligent Systems (ICNGIS), pp. 1-6, 2016.

[7] Kumar Manoj, D.K. Yadav, Singh Ankur and Kr Vijay,(2015), "A Movie Recommender System: MOVREC", 2015 International Journal of Computer Applications, vol. 124, pp. 7-11.

[8] S. G Wa1unj and K Sadafa1e, (2017), "An online recommendation system for e-commerce based on Apache Mahout framework", 2017 ACM SIGMIS International Conference on Computers and People Research, pp. 153-158, 2013.

[9] H. W. Chen, Y. L. Wu, M. K. Hor and C. Y. Tang, (2017), "Fully content-based movie recommender system with feature extraction using neural network", 2017 International Conference on Machine Learning and Cybernetics (ICMLC), pp. 504-509, 2017.

[10] Z. D Sarwar and M. S Shang, (2016), "User-Based collaborative filtering recommendation algorithms on Hadoop", Proc. of Third International Workshop on Knowledge Discovery and Data Mining, pp. 478-481, 2016.