SRM

INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

Movie Recommendation System

Project Report

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

Movie Recommendation System

Project Mentor:
Ayushi Pandey
Batch:
CSE-K

Name:
Harsh Sharma
Registration Number:
RA1711003030671

Acknowledgement

Now-a-days recommender systems are used in our day to day life. Yet, they are far from perfection. In this project, we will try to understand the various types of recommendation systems also comparing their output with other smaller datasets. We will be trying to develop a scalable model to Perform statistics. We commence by developing and comparing the different kinds of prototypes on a smaller dataset of 1000 ratings. Then, we try to gauge the system so that it is able to handle 200 ratings by using MS SQL server. We come to know that for a concise dataset, implementing user-based collaborative filtering results with better and efficient outputs.

Contents

- Acknowledgement
- Introduction
- Objectives and Problem Definition
- System Requirements
- System Model and design
- Source Code
- Output
- Implementation
- Conclusion
- References

Problem Definition/ Objective

Recommender systems are information filtering tools that aspire to predict the rating for users and items, predominantly from big data to recommend their likes. Movie recommendation systems provide a mechanism to assist users in classifying users with similar interests.

Recommender systems help the users to get personalized recommendations, helps users to take correct decisions in their online transactions, increase sales and redefine the users web browsing experience, retain the customers, enhance their shopping experience. ... Recommendation engines provide personalization.

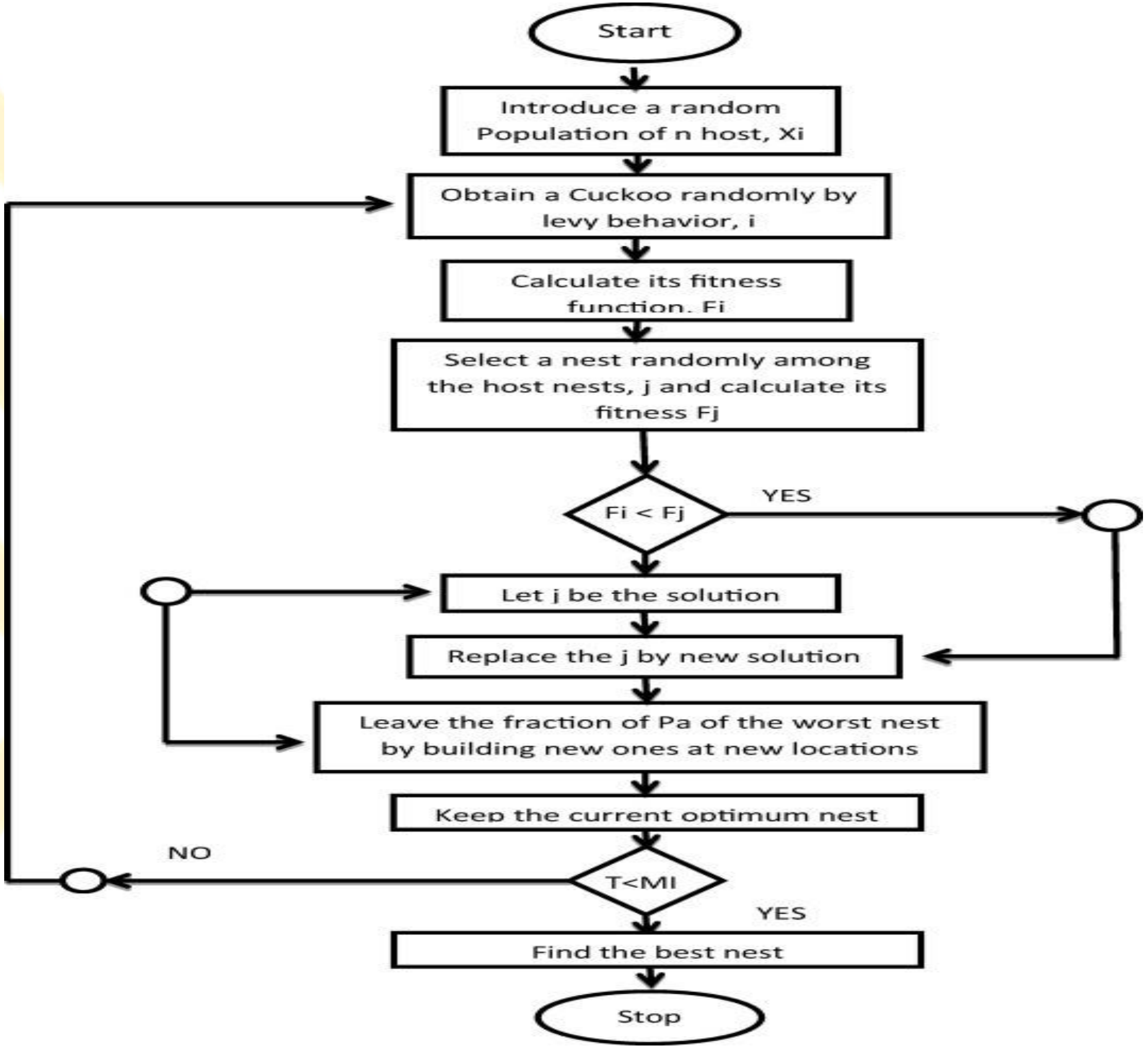
Introduction

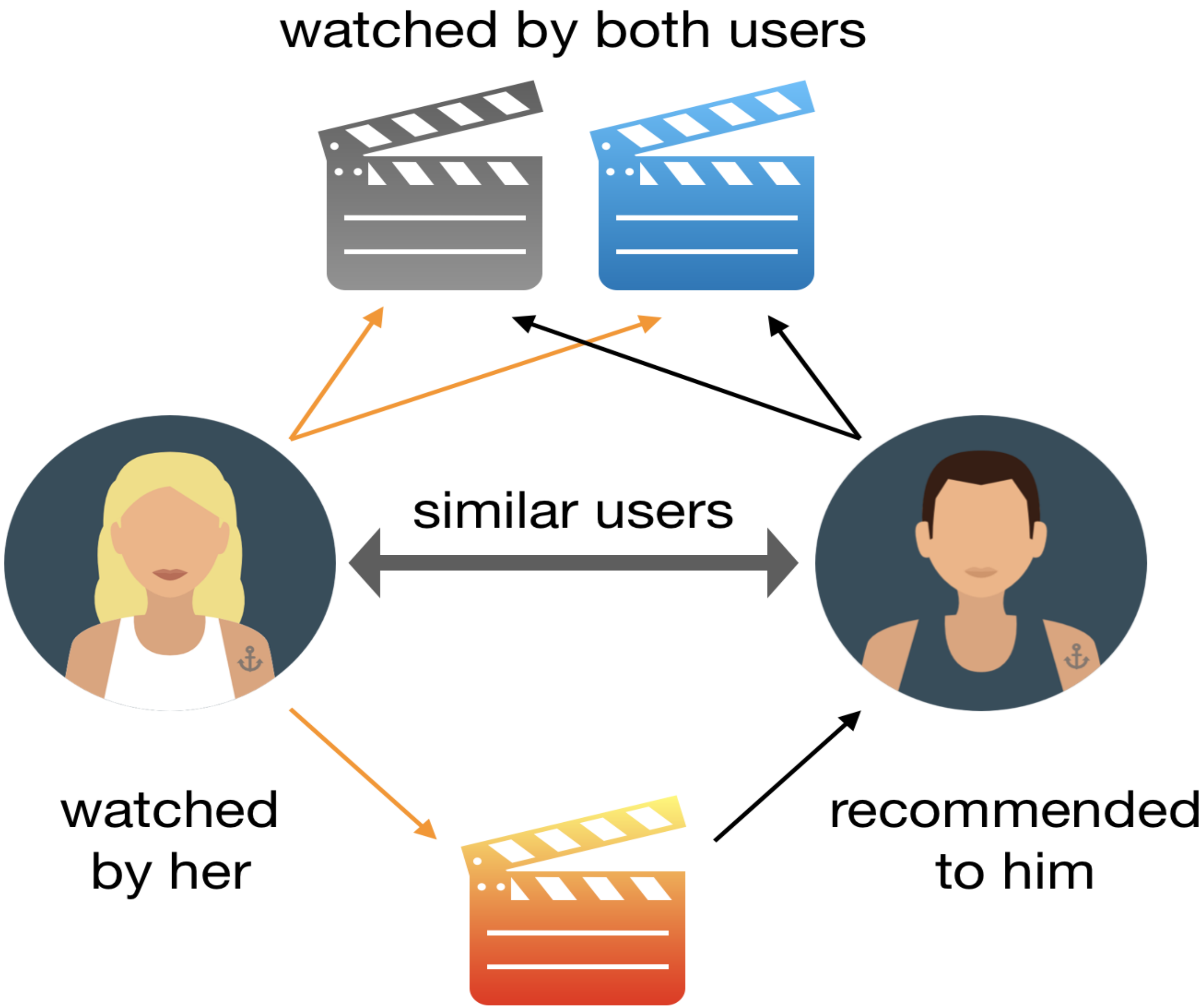
A recommendation system is a type of information filtering system which challenges to assume the priorities of a user, and make recommendations on the basis of user's priorities. Huge range of applications of recommendation systems are provided to the user. The popularity of recommendations systems have gradually increased and are recently implemented in almost all online platforms that people use. The content of such system differs from films, podcasts, books and videos, to colleagues and stories on social media, to commodities on e-commerce websites, to people on commercial and dating websites. Often, these systems are able to retrieve and filter data about a user's preferences, and can use this intel to advance their suggestions in the upcoming period. For an instance, Twitter can analyze your collaboration with several stories on your wall so as to comprehend what types of stories please you. Many a times, these systems can be improvised on the basis of activities of a large number of people. For example, if Flipkart notices that a large number of users who buy the modern laptop also buy a laptop bag. They can commend the laptop bag to a new customer who has just added a laptop to his cart. Due to the advances in recommender systems, users continuously expect good results. They have a low edge for services that are not able to make suitable recommendations. If a music streaming application is not able to foresee and play song that the user prefers, then the user will just stop using it. This has led to a high importance by technical corporations on refining their recommendation structures. However, the problem is more complicated than it appears. Every user has different likes and dislikes. In addition, even the taste of a single customer can differ depending on a large number of aspects, such as mood, season, or type of activity the user is performing. For an instance, the type of music one would prefer to listen during exercising varies critically from the type of music he would listen to while preparing dinner. They must discover new areas to determine more about the customer, whilst still determining almost all of what is already known about of the customer. Two critically important methods are widely used for recommender systems. One is content-based filtering, where we attempt to shape the users preferences using data retrieved, and suggest items based on that profile. The other is collaborative filtering, where in we try to cluster alike users together and use data about the group to make recommendations to the customer.

System Requirements

- Python >=3.5
- pandas
- numpy
- scipy
- scikit-learn
- scikit-surprise
- lightfm
- matplotlib
- seaborn
- jupyter notebook
- jupyter lab
- textblob

Flowchart


Start
Introduce a random Population of n host, Xi
Obtain a Cuckoo randomly by levy behavior, i
Calculate its fitness function, Fi
Select a nest randomly among the host nests, j and calculate its fitness Fj
Fi < Fj
YES
Let j be the solution
Replace the j by new solution
Leave the fraction of Pa of the worst nest by building new ones at new locations
Keep the current optimum nest
NO
T<MI
YES
Find the best nest
Stop



General Representation

Source Code

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
###### helper functions. Use them when needed #######
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]

def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
##################################################

##Step 1: Read CSV File

df = pd.read_csv("movie_dataset.csv")
#print (df.columns)

##Step 2: Select Features

features = ['keywords','cast','genres','director']
for feature in features:
df[feature] = df[feature].fillna('')

##Step 3: Create a column in DF which combines all selected features
```

Source Code

- features = ['keywords','cast','genres','director']
- for feature in features:
-     df[feature] = df[feature].fillna('')
- ##Step 3: Create a column in DF which combines all selected features
- def combined_features(row):
-     return row['keywords'] + " " + row['cast'] + " " +row['genres'] + " " +row['director']
-
- df["combined_features"] = df.apply(combined_features,axis=1)
- print ("Combined Features:" ,df["combined_features"].head)
- ##Step 4: Create count matrix from this new combined column
- cv = CountVectorizer()

count_matrix = cv.fit_transform(df["combined_features"])

- ##Step 5: Compute the Cosine Similarity based on the count_matrix

cosine_sim = cosine_similarity(count_matrix)

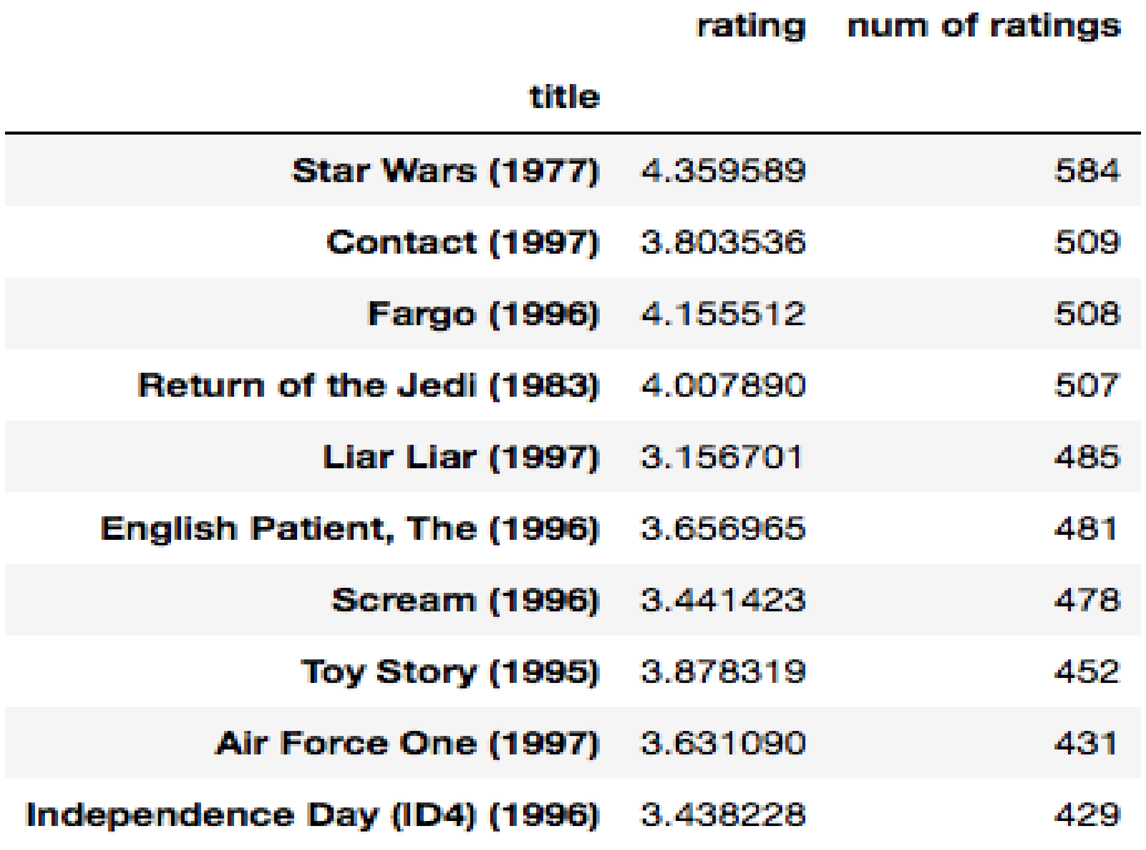
- movie_user_likes = "Avatar"

## Step 6: Get index of this movie from its title

movie_index = get_index_from_title(movie_user_likes)

- similar_movies = list(enumerate(cosine_sim[movie_index]))

```
## Step 7: Get a list of similar movies in descending order of similarity score
 sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)
## Step 8: Print titles of first 50 movies
i=0
for movie in sorted_similar_movies:
    print (get_title_from_index(movie[0]))
    i=i+1
    if i>50:
        break
```



	rating	num of ratings
title		
Star Wars (1977)	4.359589	584
Contact (1997)	3.803536	509
Fargo (1996)	4.155512	508
Return of the Jedi (1983)	4.007890	507
Liar Liar (1997)	3.156701	485
English Patient, The (1996)	3.656965	481
Scream (1996)	3.441423	478
Toy Story (1995)	3.878319	452
Air Force One (1997)	3.631090	431
Independence Day (ID4) (1996)	3.438228	429

General Output

➢ Implementation

Python | Implementation of Movie Recommender System

Recommender System is a system that seeks to predict or filter preferences according to the user’s choices.

Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general.

Recommender systems produce a list of recommendations in any of the two ways –

Collaborative filtering: Collaborative filtering approaches build a model from user’s past behavior (i.e. items purchased or searched by the user) as

well as similar decisions made by other users.

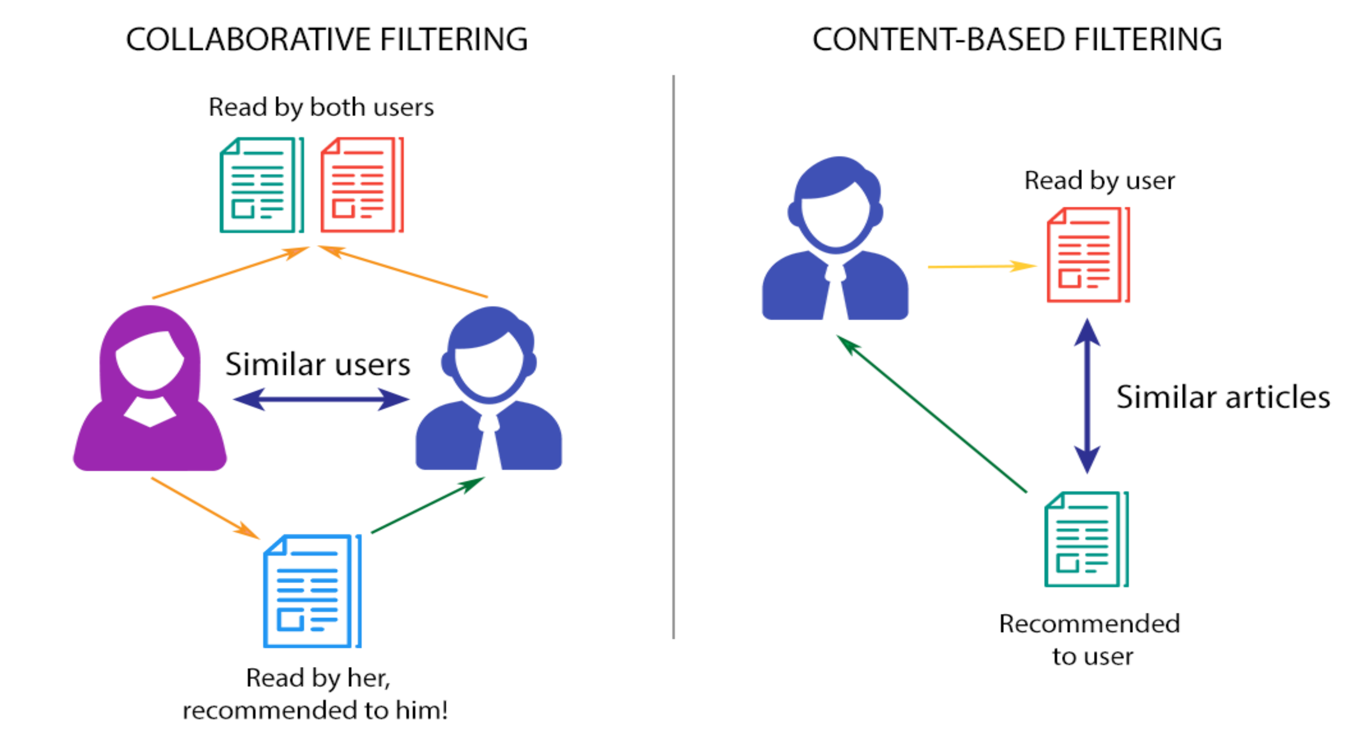
This model is then used to predict items (or ratings for items) that user may have an interest in.

Content-based filtering: Content-based filtering approaches uses a series of discrete characteristics of an item in order to recommend additional items with similar properties.

Content-based filtering methods are totally based on a description of the item and a profile of the user’s preferences. It recommends items based on user’s past preferences.

Working

- The movie recommendation system basically works by providing suggestions to the users by using the two re
  This movie recommendation system recommends movies to a user or client by evaluating IMDB ratings. The software and language which we have used for designing our interface and front end is Visual Basic Asp.Net. For creating database, we have used SQL Server since it is convenient. This system collaborates with IMDB ratings and displays a list of movies which are highly rated by a user based on category of the movie. This approach asks the user to provide 2 inputs – 1. Category of the movie (for e.g. comedy) 2. Year in which the movie is released (for e.g. 2016) The algorithm segregates the list of movies from the dataset according to the inputs provided by user and finally displays the list of movies. The algorithm compares the inputs with the traits of the dataset and formulates the list. A user may select more than one category according to his fancies. A bright feature of allowing the user to rate movies has enhanced the beauty of this recommender system. This is achieved by using collaborative filtering approach, wherein the system will provide recommendations to other likeminded users which have the same taste.

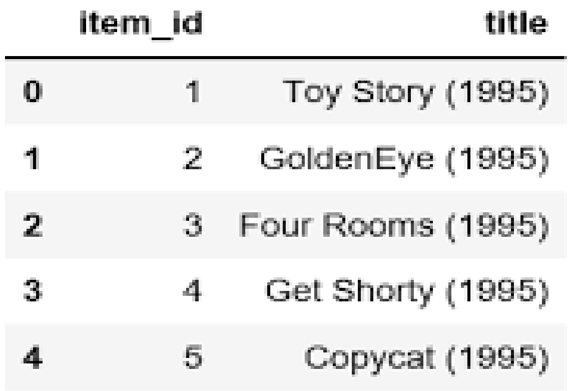


CONCLUSIONS

Movie recommendation systems which are existing have poor efficiency due to which movies are suggested in view of aspects for example - movie rated & evaluated by the User. They have almost same viewing tastes, by means of data mining and insisting movies based on juncture of the three methods mentioned above that is - User Based Collaborative filtering, Content-based algorithm & data mining because of which the user will not only be recommended movies but this scheme also delivers the user with additionally advanced and sophisticated endorsements as movies which have a poor rating score in any of the Movie features produced based on data mining will be refined out during the significant allocation platform of the expected three way hybrid movie recommendation system.

REFERENCES

[1] Microsoft SQL Server 2012 Management and Administration Second, Kindle Edition
Book by Ross Mistry.
[2] ASP.NET: The Complete Reference
Book by Matthew MacDonald (2002).



	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

Thank you