Movie Recommendation System using Python and Pandas

Group Members:

Rajdeep Maji
Asansol Engineering College
10800220046

Soumya Banerjee
Asansol Engineering College
10800220056

Subinaya Mukherjee
Asansol Engineering College
10800221140

Simran Tiwary
Asansol Engineering College
10800221142

Sudip Bhattacharjee
Asansol Engineering College
10800221134

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty, **Prof. Arnab Chakraborty** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

RajdeepMaji

Soumya Banerjee

Subinaya Mukherjee

Simran Tiwary

Sudip Bhattacharjee

Abstract

In this hustling world, entertainment is a necessity for each one of us to refresh our mood and energy. Entertainment regains our confidence for work and we can work more enthusiastically. For revitalizing ourselves, we can listen to our preferred music or can watch movies of our choice. For watching favourable movies online we can utilize movie recommendation systems, which are more reliable, since searching of preferred movies will require more and more time which one cannot afford to waste. In this paper, to improve the quality of a movie recommendation system, a Hybrid approach by combining content based filtering and collaborative filtering, using Support Vector Machine as a classifier and genetic algorithm is presented in the proposed methodology and comparative results have been shown which depicts that the proposed approach shows an improvement in the accuracy, quality and scalability of the movie recommendation system than the pure approaches in three different datasets. Hybrid approach helps to get the advantages from both the approaches as well as tries to eliminate the drawbacks of both methods.

What are Recommendation systems?

Recommendation systems are becoming increasingly important in today's extremely busy world. People are always short on time with the myriad tasks they need to accomplish in the limited 24 hours. Therefore, the recommendation systems are important as they help them make the right choices, without having to expend their cognitive resources.

The purpose of a recommendation system basically is to search for content that would be interesting to an individual. Moreover, it involves a number of factors to create personalised lists of useful and interesting content specific to each user/individual. Recommendation systems are Artificial Intelligence based algorithms that skim through all possible options and create a customized list of items that are interesting and relevant to an individual. These results are based on their profile, search/browsing history, what other people with similar traits/demographics are watching, and how likely are you to watch those movies. This is achieved through predictive modeling and heuristics with the data available.

Use-cases of Recommendation systems

Recommendations are not a new concept. Even when e-commerce was not that prominent, the sales staff in retail stores recommended items to the customers for the purpose of upselling and cross-selling, and ultimately maximise profit. The aim of recommendation systems is just the same.

Another objective of the recommendation system is to achieve customer loyalty by providing relevant content and maximising the time spent by a user on your website or channel. This also helps in increasing customer engagement.

On the other hand, ad budgets can be optimized by showcasing products and services only to those who have a propensity to respond to them.

Filtration Strategies for Movie Recommendation Systems

Movie recommendation systems use a set of different filtration strategies and algorithms to help users find the most relevant films. The most popular categories of the ML algorithms used for movie recommendations include content-based filtering and collaborative filtering systems.

— Content-Based Filtering

A filtration strategy for movie recommendation systems, which uses the data provided about the items (movies). This data plays a crucial role here and is extracted from only one user. An ML algorithm used for this strategy recommends motion pictures that are similar to the user's preferences in the past. Therefore, the similarity in content-based filtering is generated by the data about the past film selections and likes by only one user.

How does it work? The recommendation system analyzes the past preferences of the user concerned, and then it uses this information to try to find similar movies. This information is available in the database (e.g., lead actors, director, genre, etc.). After that, the system provides movie recommendations for the user. That said, the core element in content-based filtering is only the data of only one user that is used to make predictions.

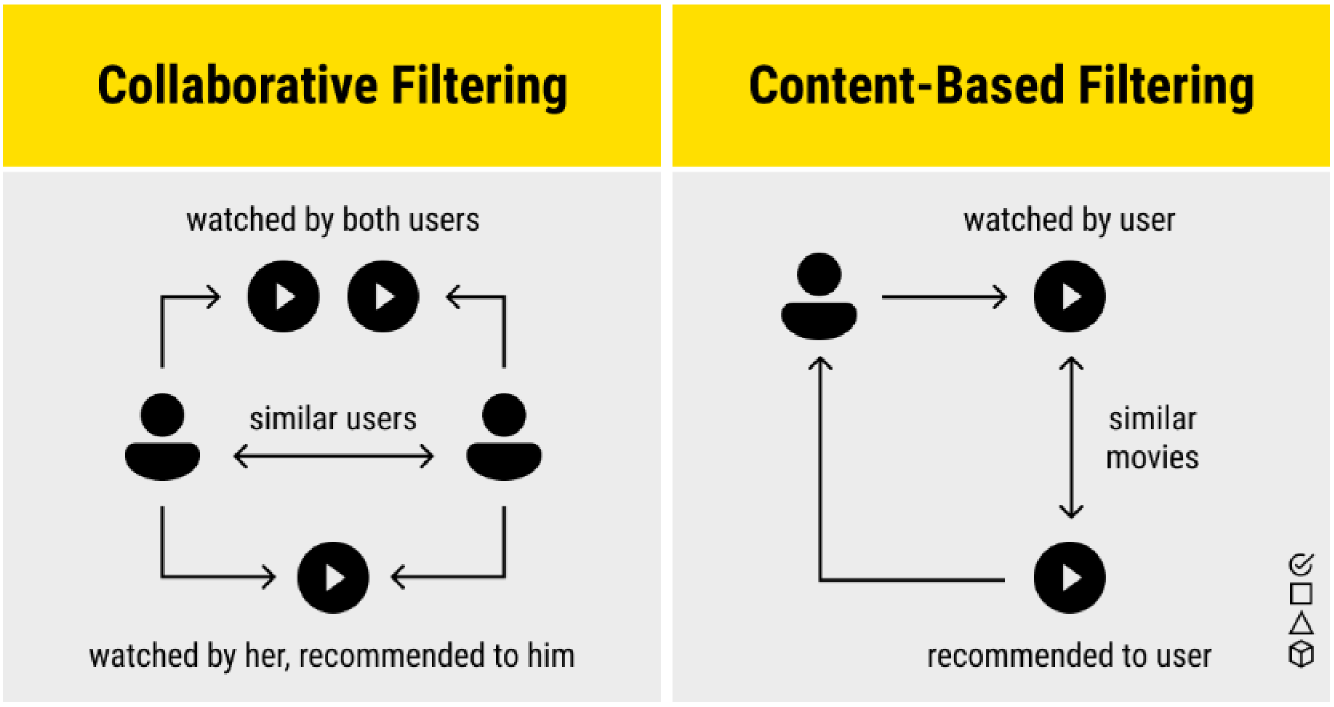
— Collaborative Filtering

As the name suggests, this filtering strategy is based on the combination of the relevant user's and other users' behaviors. The system compares and contrasts these behaviors for the most optimal results. It's a collaboration of the multiple users' film preferences and behaviors.

What's the mechanism behind this strategy? The core element in this movie recommendation system and the ML algorithm it's built on is the history of all users in the database. Basically, collaborative filtering is based on the interaction of all users in the system with the items (movies). Thus, every user impacts the final outcome of this ML-based recommendation system, while content-based filtering depends strictly on the data from one user for its modeling.

Collaborative filtering algorithms are divided into two categories:

User-based collaborative filtering. The idea is to look for similar patterns in movie preferences in the target user and other users in the database.
Item-based collaborative filtering. The basic concept here is to look for similar items (movies) that target users rate or interact with.
The modern approach to the movie recommendation systems implies a mix of both strategies for the most gradual and explicit results.



Popularity based recommendation system

Let us take an example of a website that streams movies. The website is in its nascent stage and has listed all the movies for the users to search and watch. What the website misses here is a recommendation system. This results in users browsing through a long list of movies, with no suggestions about what to watch. This, in turn, reduces the propensity of a user to engage with the website and use its services. Therefore, the simplest way to fix this issue is to use a popularity based recommendation system. Top review websites like IMDb and Rotten Tomatoes maintain a database of movies and their popularity in terms of reviews and ratings. Utilising this data to recommend the most popular movies to users based on their star ratings, could increase their content consumption.

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

Demerits of the popularity based recommendation system

– Recommendations are not personalized as per user attributes and all users see the same recommendations irrespective of their preferences

– Another problem is that the number of reviews (which reflects the number of people who have viewed the movie) will vary for each movie and hence the average star rating will have discrepancies.

– The system doesn’t take into account the regional and language preferences and might recommend movies in languages that a regional dialect speaking individual might not understand

A popularity based recommendation system when tweaked as per the needs, audience, and business requirement, it becomes a hybrid recommendation system. Additional logic is added to include customization as per the business needs.

Introduction

1.1 Relevance of the Project

A recommendation system or recommendation engine is a model used for information filtering where it tries to predict the preferences of a user and provide suggests based on these preferences. These systems have become increasingly popular nowadays and are widely used today in areas such as movies, music, books, videos, clothing, restaurants, food, places and other utilities. These systems collect information about a user's preferences and behaviour, and then use this information to improve their suggestions in the future.

Movies are a part and parcel of life. There are different types of movies like some for entertainment, some for educational purposes, some are animated movies for children, and some are horror movies or action films. Movies can be easily differentiated through their genres like comedy, thriller, animation, action etc. Other way to distinguish among movies can be either by releasing year, language, director etc. Watching movies online, there are a number of movies to search in our most liked movies . Movie Recommendation Systems helps us to search our preferred movies among all of these different types of movies and hence reduce the trouble of spending a lot of time searching our favourable movies. So, it requires that the movie recommendation system should be very reliable and should provide us with the recommendation of movies which are exactly same or most matched with our preferences.

A large number of companies are making use of recommendation systems to increase user interaction and enrich a user's shopping experience. Recommendation systems have several benefits, the most important being customer satisfaction and revenue. Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffers with poor recommendation quality and scalability issues.

1.2 **Problem Statement:**

The goal of the project is to recommend a movie to the user.

Providing related content out of relevant and irrelevant collection of items to users of online service providers.

1.3 **Objective of the Projects**

• Improving the Accuracy of the recommendation system

• Improve the Quality of the movie Recommendation system

• Improving the Scalability.

• Enhancing the user experience.

1.4 **Scope of the Project**

The objective of this project is to provide accurate movie recommendations to users. The goal of the project is to improve the quality of movie recommendation system, such as accuracy, quality and scalability of system than the pure approaches. This is done using Hybrid approach by combining content based filtering and collaborative filtering, To eradicate the overload of the data, recommendation system is used as information filtering tool in social networking sites .Hence, there is a huge scope of exploration in this field for improving scalability, accuracy and quality of movie recommendation systems Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffers with poor recommendation quality and scalability issues.

1.5 **Methodology for Movie Recommendation**

The hybrid approach proposed an integrative method by merging fuzzy k-means clustering method and genetic algorithm based weighted similarity measure to construct a movie recommendation system. The proposed movie recommendation system gives finer similarity metrics and quality than the existing Movie recommendation system but the computation time which is

taken by the proposed recommendation system is more than the existing recommendation system. This problem can be fixed by taking the clustered data points as an input dataset

The proposed approach is for improving the scalability and quality of the movie recommendation system .We use a Hybrid approach , by unifying Content-Based Filtering and Collaborative Filtering, so that the approaches can be profited from each other. For computing similarity between the different movies in the given dataset efficiently and in least time and to reduce computation time of the movie recommender engine we used cosine similarity measure.

Agile Methodology:

1. collecting the data sets: Collecting all the required data set from Kaggle web site.in this project we require movie.csv,ratings.csv,users.csv.

2. Data Analysis: make sure that that the collected data sets are correct and analysing the data in the csv files. i.e. checking whether all the column Felds are present in the data sets.

3. Algorithms: in our project we have only two algorithms one is cosine similarity and other is single valued decomposition are used to build the machine learning recommendation model.

4. Training and Testing the model: once the implementation of algorithm is completed . we have to train the model to get the result. We have tested it several times the model is recommend different set of movies to different users.

5. Improvements in the project: In the later stage we can implement different algorithms and methods for better recommendation

Literature Survey

Over the years, many recommendation systems have been developed using either collaborative, content based or hybrid filtering methods. These systems have been implemented using various big data and machine learning algorithms.

2.1 Movie Recommendation System by K-Means Clustering AND K-Nearest Neighbour

A recommendation system collect data about the user's preferences either implicitly or explicitly on different items like movies. An implicit acquisition in the development of movie recommendation system uses the user's behaviour while watching the movies. On the other hand, a explicit acquisition in the development of movie recommendation system uses the user's previous ratings or history. The other supporting technique that are used in the development of recommendation system is clustering. Clustering is a process to group a set of objects in such a way that objects in the same clusters are more similar to each other than to those in other clusters. K- Means Clustering along with K-Nearest Neighbour is implemented on the movie lens dataset in order to obtain the best-optimized result. In existing technique, the data is scattered which results in a high number of clusters while in the proposed technique data is gathered and results in a low number of clusters. The process of recommendation of a movie is optimized in the proposed scheme. The proposed recommender system predicts the user's preference of a movie on the basis of different parameters. The recommender system works on the concept that people are having common preference or choice. These users will influence on each other's opinions. This process optimizes the process and having lower RMSE.

2.2 Movie Recommendation System Using Collaborative Filtering: By Ching-Seh (Mike) Wu,Deepti Garg,Unnathi Bhandary

Collaborative filtering systems analyse the user's behaviour and preferences and predict what they would like based on similarity with other users. There are two kinds of collaborative filtering systems; user-based recommender and item-based recommender.

1. Use-based filtering: User-based preferences are very common in the field of designing personalized systems. This approach is based on the user's likings. The process starts with users giving ratings (1-5) to some movies. These ratings can be implicit or explicit. Explicit ratings are when the user explicitly rates the item on some scale or indicates a thumbs-up/thumbs-down to the item. Often explicit ratings are hard to gather as not every user is much interested in providing feedbacks. In these scenarios, we gather implicit ratings based on their behaviour. For instance, if a user buys a product more than once, it indicates a positive preference. In context to movie systems, we can imply that if a user watches the entire movie, he/she has some likeability to it. Note that there are no clear rules in determining implicit ratings. Next, for each user, we first find some defined number of nearest neighbours. We calculate correlation between users' ratings using Pearson Correlation algorithm. The assumption that if two users' ratings are highly correlated, then these two users must enjoy similar items and products is used to recommend items to users.

2. Item-based filtering: Unlike the user-based filtering method, item- based focuses on the similarity between the item’s users like instead of the users themselves. The most similar items are computed ahead of time. Then for recommendation, the items that are most similar to the target item are recommended to the user.

SYSTEM REQUIREMENTS SPECIFICATION

This chapter involves both the hardware and software requirements needed for the project and detailed explanation of the specifications.

3.1 Hardware Requirements

- A PC with Windows/Linux OS
- Processor with 1.7-2.4gHz speed
- Minimum of 8gb RAM
- 2gb Graphic card

3.2 Software Specification

- Text Editor (VS-code/WebStorm)
- Anaconda distribution package (PyCharm Editor)
- Python libraries

3.3 Software Requirements

3.3.1 Anaconda distribution:

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system conda. The anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.3

3.3.3 Python libraries:

For the computation and analysis we need certain python libraries which are used to perform analytics. Packages such as SKlearn, Numpy, pandas, Matplotlib, Flask framework, etc are needed.

SKlearn: It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

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. Pandas: Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Data frame.

SYSTEM ANALYSIS AND DESIGN

4.1 System Architecture of Proposed System:

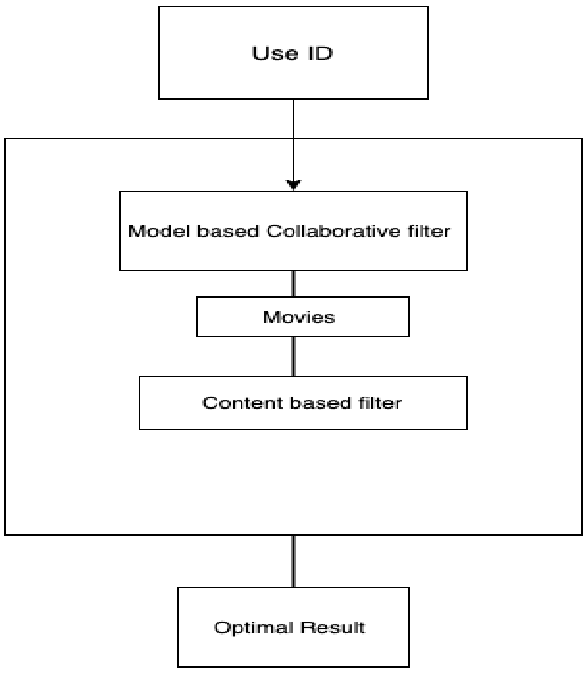


Fig:-4.1 Architecture for hybrid approach

For each different individual use different list of movies are recommended ,as user login or enters the user id based on two different approaches used in the project each will recommend the set of movies to the particular user by combining the both the set of movie based on the user the hybrid model will recommend the single list of movie to the user.

4.3 Dataflow:

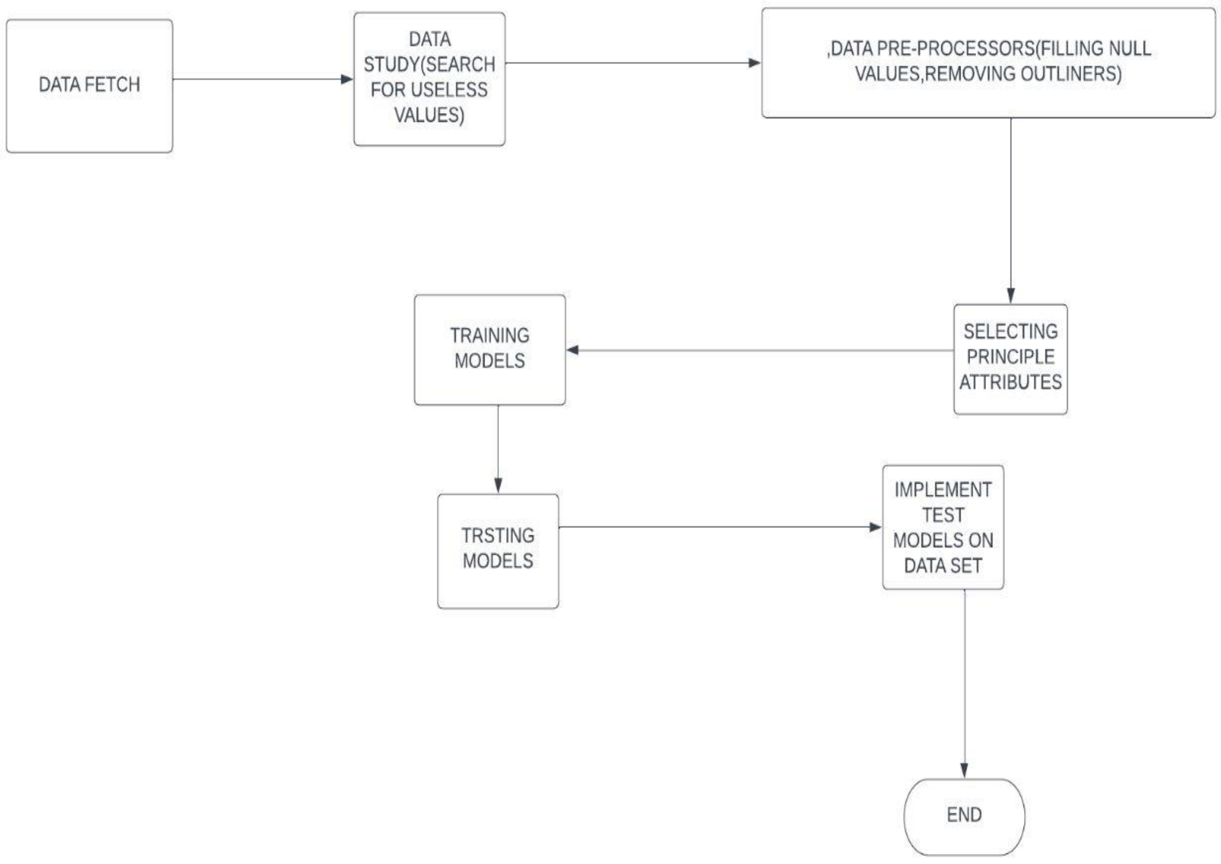

DATA FETCH
DATA STUDY(SEARCH FOR USELESS VALUES)
,DATA PRE-PROCESSORS(FILLING NULL VALUES,REMOVING OUTLINERS)
SELECTING PRINCIPLE ATTRIBUTES
TRAINING MODELS
TRSTING MODELS
IMPLEMENT TEST MODELS ON DATA SET
END

Fig:-4.3 Data Flow Diagram

Initially load the data sets that are required to build a model the data set that are required in this project are movies.csv, rating.csv, all the data sets are available in the Kaggle.com. Basically, two models are built in this project content based and collaborative filtering each produce a list of movies to a particular user by combining both based on the useid a single final list of movies are recommended to the particular user

IMPLEMENTATION

The Proposed System Make Use Different Algorithms and Methods for the implementation of Hybrid Approach

5.1 **Cosine Similarity**: Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

Formula:

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

5.2 **Singular Value Decomposition (SVD):**

Let A be an n*d matrix with singular vectors v1, v2, . . . , vr and corresponding singular values σ1, σ2, . . . , σr. Then ui = (1/σi)Avi , for i = 1, 2, . . . , r, are the left singular vectors and by Theorem 1.5, A can be decomposed into a sum of rank one matrices a

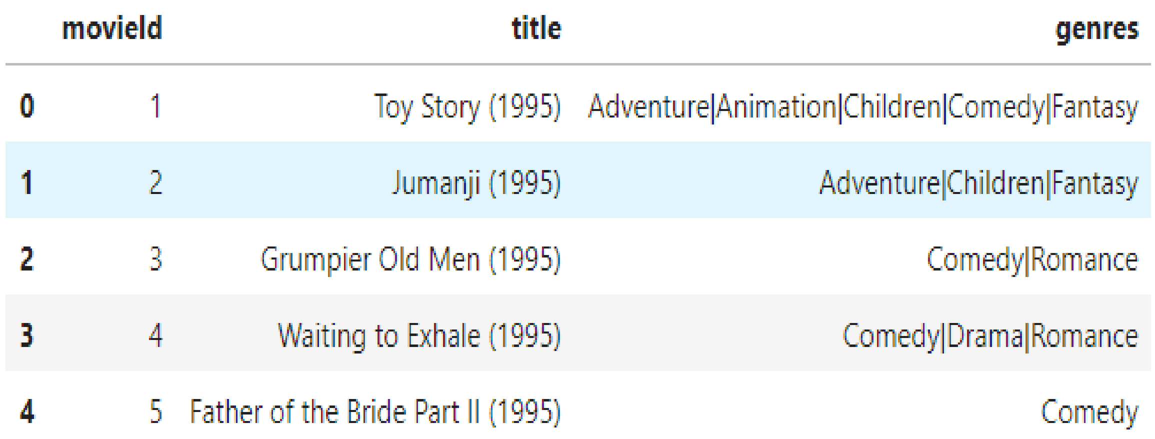
$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T .$$

We first prove a simple lemma stating that two matrices A and B are identical if Av = Bv for all v. The lemma states that in the abstract, a matrix A can be viewed as a transformation that maps vector v onto Av

Code Requirements

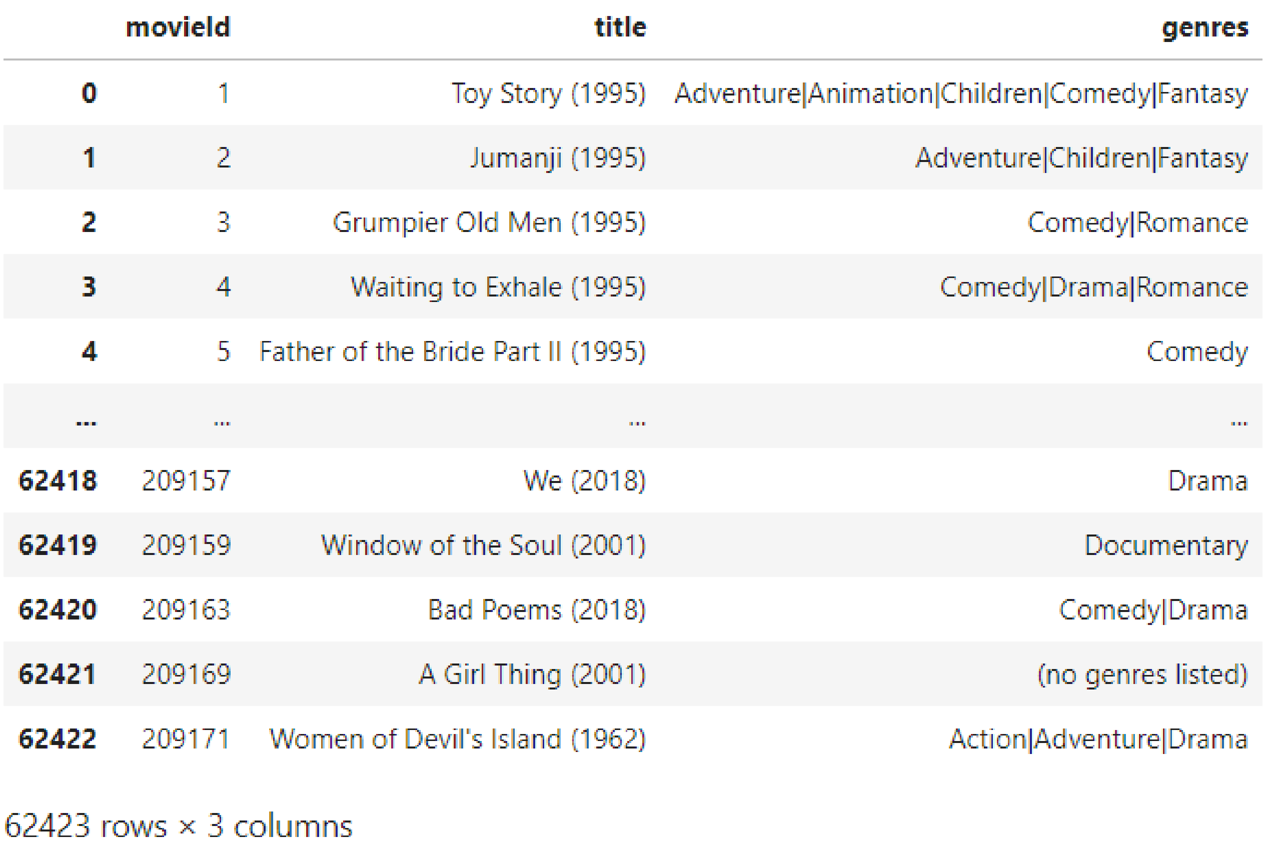
In this project we have used python programming language and pandas library for working with datasets.

```
import pandas as pd
movies = pd.read_csv("movies.csv")
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

```
import re
#cleans title of movies(data cleaning)
def clean_title(title):
title = re.sub("[^a-zA-Z0-9 ]", "", title)
return title
movies["clean_title"] = movies["title"].apply(clean_title)
movies
```

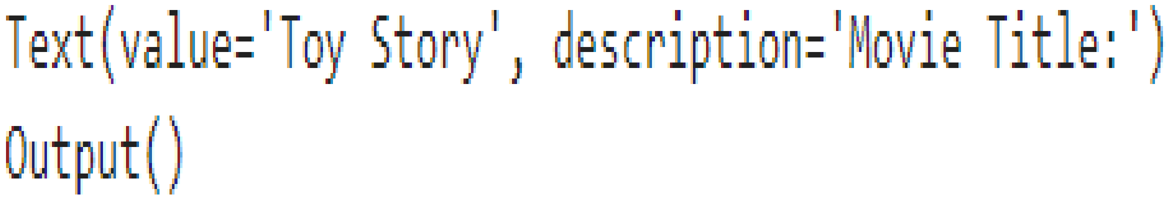


	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
62418	209157	We (2018)	Drama
62419	209159	Window of the Soul (2001)	Documentary
62420	209163	Bad Poems (2018)	Comedy|Drama
62421	209169	A Girl Thing (2001)	(no genres listed)
62422	209171	Women of Devil's Island (1962)	Action|Adventure|Drama

62423 rows × 3 columns

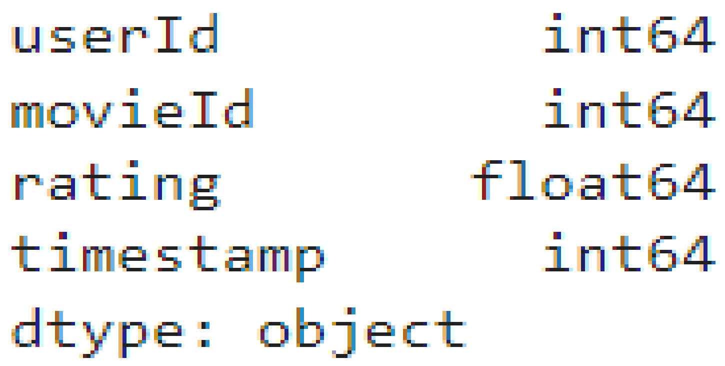
```
from sklearn.feature_extraction.text import TfidfVectorizer
 vectorizer = TfidfVectorizer(ngram_range=(1,2))

 tfidf = vectorizer.fit_transform(movies["clean_title"])
 from sklearn.metrics.pairwise import cosine_similarity
 import numpy as np
 def search(title):
 title = clean_title(title)
```

```
query_vec = vectorizer.transform([title])
similarity = cosine_similarity(query_vec, tfidf).flatten()
indices = np.argpartition(similarity, -5)[-5:]
results = movies.iloc[indices].iloc[::-1]

return results
import ipywidgets as widgets
from IPython.display import display
#search box
movie_input = widgets.Text(
 value='Toy Story',
description='Movie Title:',
disabled=False
)
movie_list = widgets.Output()
def on_type(data):
with movie_list:
movie_list.clear_output()
title = data["new"]
if len(title) > 5:
display(search(title))
movie_input.observe(on_type, names='value')
display(movie_input, movie_list)
```



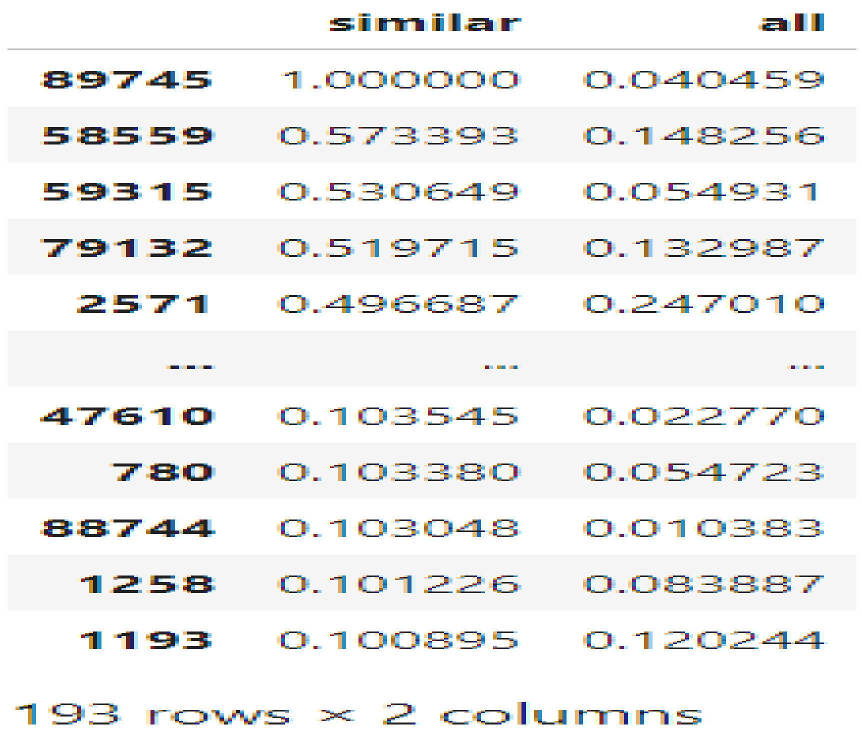
```
Text(value='Toy Story', description='Movie Title:')
Output()
```

```
movie_id = 89745
#def find_similar_movies(movie_id):
movie = movies[movies["movieId"] == movie_id]
ratings = pd.read_csv("https://www.dropbox.com/s/i4jcaay0tbpk6ed/ratings.csv?dl=1")
ratings.dtypes
```



```
userId         int64
movieId        int64
rating       float64
timestamp      int64
dtype: object
```

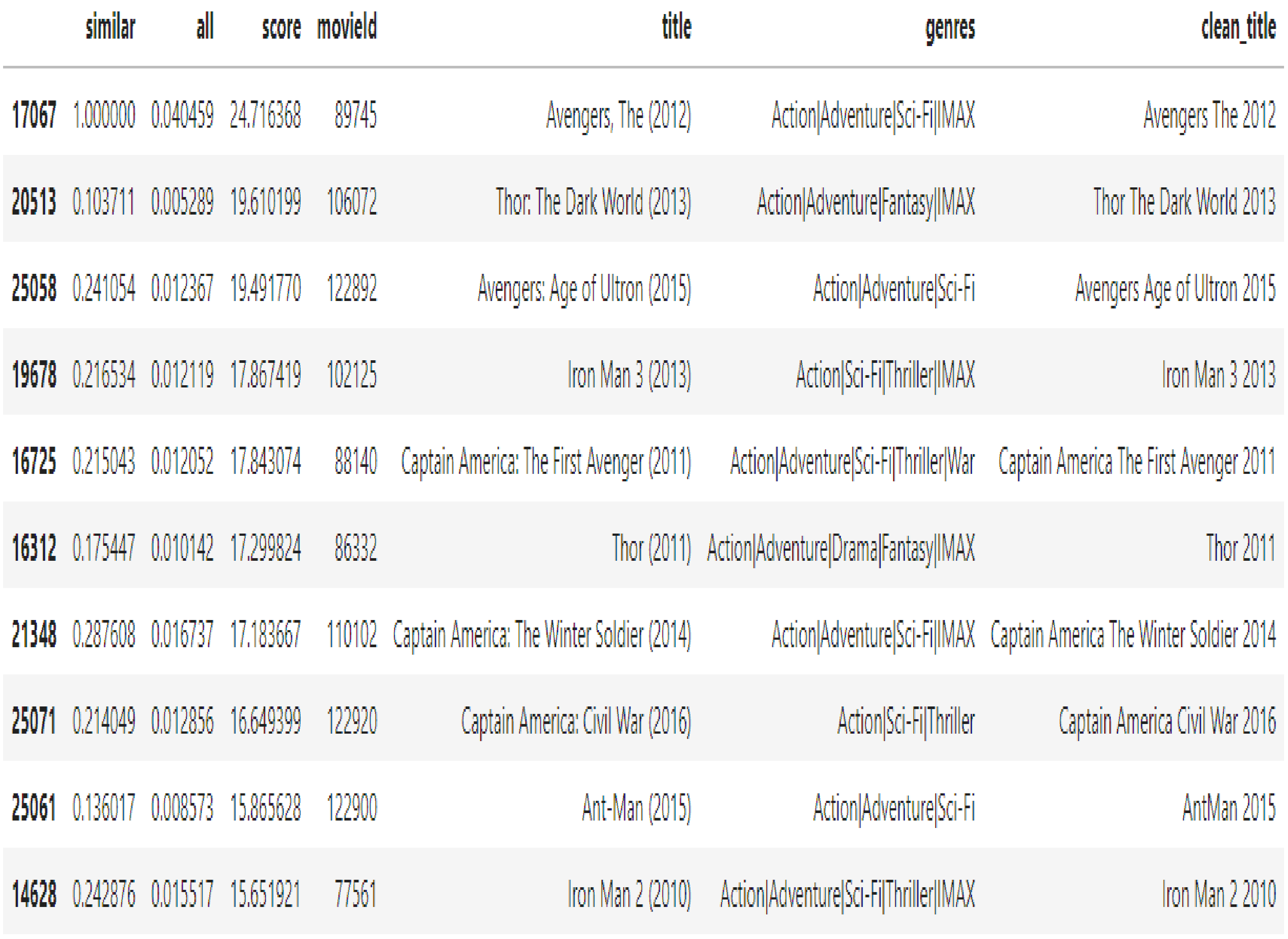
```
  similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] >
4)]["userId"].unique()

  similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings["rating"] >
4)]["movieId"]

  similar_user_recs = similar_user_recs.value_counts() / len(similar_users)
  similar_user_recs = similar_user_recs[similar_user_recs > .10]
  all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"] > 4)]
  all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].unique())
  rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
  rec_percentages.columns = ["similar", "all"]
  rec_percentages
```



	similar	all
89745	1.000000	0.040459
58559	0.573393	0.148256
59315	0.530649	0.054931
79132	0.519715	0.132987
2571	0.496687	0.247010
...	...	...
47610	0.103545	0.022770
780	0.103380	0.054723
88744	0.103048	0.010383
1258	0.101226	0.083887
1193	0.100895	0.120244

193 rows × 2 columns

```
rec_percentages["score"] = rec_percentages["similar"] / rec_percentages["all"]
rec_percentages = rec_percentages.sort_values("score", ascending=False)
rec_percentages.head(10).merge(movies, left_index=True, right_on="movieId")
```

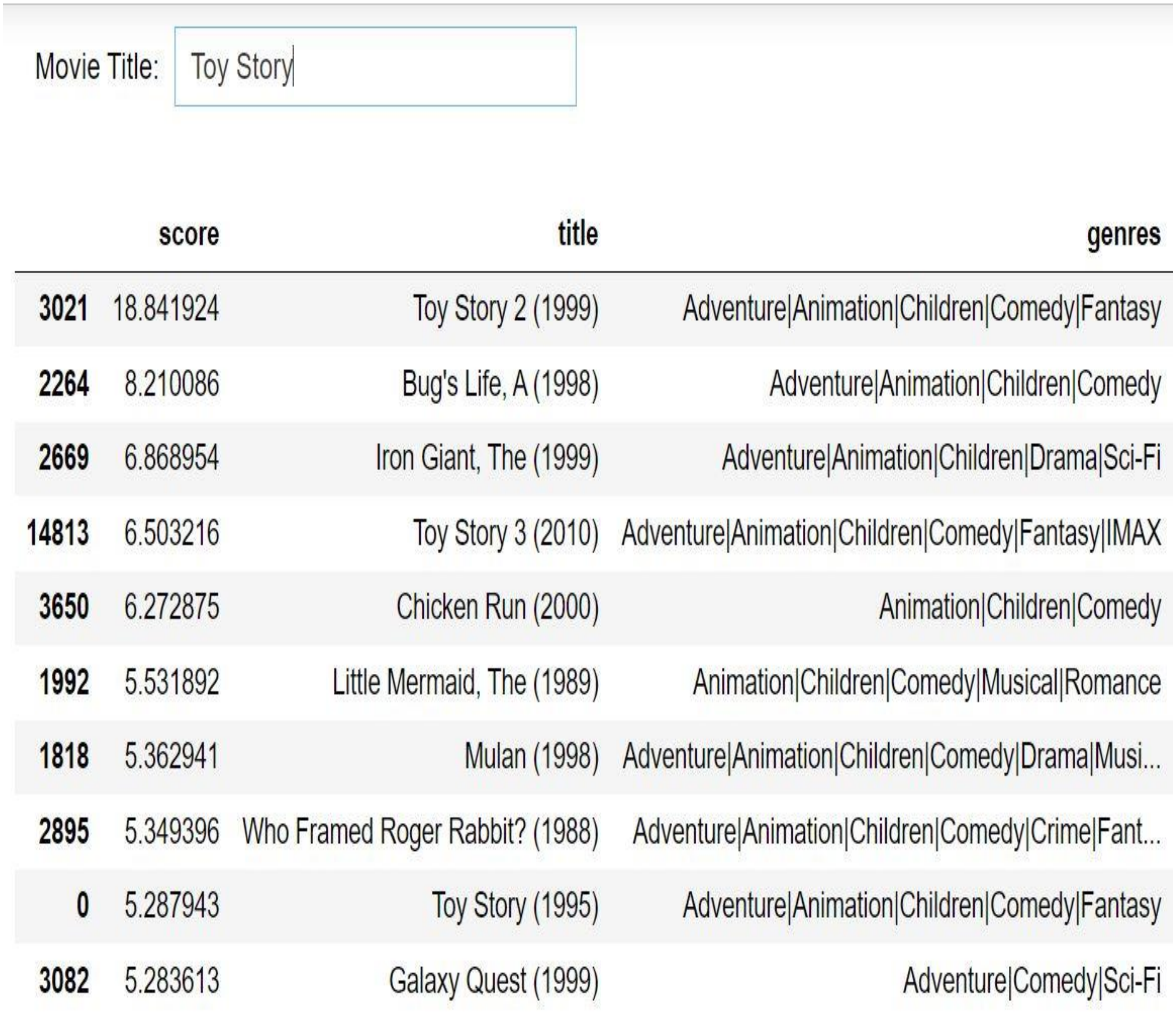


	similar	all	score	movieId	title	genres	clean_title
17067	1.000000	0.040459	24.716368	89745	Avengers, The (2012)	Action|Adventure|Sci-Fi|IMAX	Avengers The 2012
20513	0.103711	0.005289	19.610199	106072	Thor: The Dark World (2013)	Action|Adventure|Fantasy|IMAX	Thor The Dark World 2013
25058	0.241054	0.012367	19.491770	122892	Avengers: Age of Ultron (2015)	Action|Adventure|Sci-Fi	Avengers Age of Ultron 2015
19678	0.216534	0.012119	17.867419	102125	Iron Man 3 (2013)	Action|Sci-Fi|Thriller|IMAX	Iron Man 3 2013
16725	0.215043	0.012052	17.843074	88140	Captain America: The First Avenger (2011)	Action|Adventure|Sci-Fi|Thriller|War	Captain America The First Avenger 2011
16312	0.175447	0.010142	17.299824	86332	Thor (2011)	Action|Adventure|Drama|Fantasy|IMAX	Thor 2011
21348	0.287608	0.016737	17.183667	110102	Captain America: The Winter Soldier (2014)	Action|Adventure|Sci-Fi|IMAX	Captain America The Winter Soldier 2014
25071	0.214049	0.012856	16.649399	122920	Captain America: Civil War (2016)	Action|Sci-Fi|Thriller	Captain America Civil War 2016
25061	0.136017	0.008573	15.865628	122900	Ant-Man (2015)	Action|Adventure|Sci-Fi	AntMan 2015
14628	0.242876	0.015517	15.651921	77561	Iron Man 2 (2010)	Action|Adventure|Sci-Fi|Thriller|IMAX	Iron Man 2 2010

```
        def find_similar_movies(movie_id):
        similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] >
4)]["userId"].unique()

        similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings["rating"] >
4)]["movieId"]

        similar_user_recs = similar_user_recs.value_counts() / len(similar_users)
        similar_user_recs = similar_user_recs[similar_user_recs > .10]
        all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"] > 4)]
        all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].unique())
        rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
        rec_percentages.columns = ["similar", "all"]
        rec_percentages["score"] = rec_percentages["similar"] / rec_percentages["all"]
        rec_percentages = rec_percentages.sort_values("score", ascending=False)
         return rec_percentages.head(10).merge(movies, left_index=True,
        right_on="movieId")[["score", "title", "genres"]]

        import ipywidgets as widgets
        from IPython.display
        import display   movie_name_input = widgets.Text(value='Toy Storydescription='MovieTitle:',
        disabled=False)
        recommendation_list = widgets.Output()
        def on_type(data):
        with recommendation_list:
        recommendation_list.clear_output()
        title = data["new"]
        if len(title) > 5:
        results = search(title)
        movie_id = results.iloc[0]["movieId"]
        display(find_similar_movies(movie_id))
        movie_name_input.observe(on_type, names='value')
        display(movie_name_input, recommendation_list)
```

```
Text(value='Toy Story', description='Movie Title:')
Output()
```

Final Output



Movie Title: Toy Story

	score	title	genres
3021	18.841924	Toy Story 2 (1999)	Adventure|Animation|Children|Comedy|Fantasy
2264	8.210086	Bug's Life, A (1998)	Adventure|Animation|Children|Comedy
2669	6.868954	Iron Giant, The (1999)	Adventure|Animation|Children|Drama|Sci-Fi
14813	6.503216	Toy Story 3 (2010)	Adventure|Animation|Children|Comedy|Fantasy|IMAX
3650	6.272875	Chicken Run (2000)	Animation|Children|Comedy
1992	5.531892	Little Mermaid, The (1989)	Animation|Children|Comedy|Musical|Romance
1818	5.362941	Mulan (1998)	Adventure|Animation|Children|Comedy|Drama|Musi...
2895	5.349396	Who Framed Roger Rabbit? (1988)	Adventure|Animation|Children|Comedy|Crime|Fant...
0	5.287943	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3082	5.283613	Galaxy Quest (1999)	Adventure|Comedy|Sci-Fi

Data sets

Here we used datasets for two different things, first data set we used for movies and the second one is for giving us the ratings

Link for movies datasets: https://github.com/Rajdeep-2002/Movie-Recommendation/blob/main/movies.csv

Link for ratings datasets:

https://www.dropbox.com/s/i4jcaay0tbpk6ed/ratings.csv?dl=1

Advantages & Disadvantages

Advantages

Since it is based on unsupervised learning, it automatically updates the features and functions [53]. It is flexible to new input because it learns by itself. It is suitable for unidentified new inputs, for example, new movies that have no ratings or new users where there is no data about them. The new movies may be recommended when the system extracts their features, and the new users will not experience a cold start because they have somewhere to begin on the output feature map. It is also faster in computation since it easily organizes complex data and makes a good representation of the mapping for easy interpretation.

Disadvantages

The major drawback is that feature classification may not be according to the expected output; therefore, the unsupervised learning classification algorithms have to be initialized often to maintain the relevance of the clustering

Future Scope

The bigger the choice, the harder it is to make the final decision. This is especially true for modern movie fans, who have thousands of movies to pick from. But thanks to machine learning, we now have recommendation systems based on its complex algorithms and techniques.

Today, movie recommendation systems are widely used by the most popular streaming services, enabling a more personalized experience and increased user satisfaction across the platforms. Why do we need them? It’s estimated that the world cinema has released more than 500,000 movies — a number beyond one person’s control. With such an enormous number of motion pictures to choose from, developing and improving recommendation systems with ML was a crucial step to make this process easier and feasible.

Once again, ML proves to be a vital technological solution that makes our lives easier. And the more these systems evolve, the more advanced ML techniques we have at our disposal that generate the most accurate content for users and give them what they are looking for.

Conclusion

In this documentation, movie recommender systems have been described and classified. The various types of recommender systems are introduced and discussed. Special emphasis is given to explain in detail the various machine learning and metaheuristic algorithms commonly deployed in movie recommendation research. The various model metrics that summarize the quality of the model are discussed at length. The problems associated with movie recommender systems are also summarized in a structured way and discussed.

Certificate

This is to certify that Mr. Rajdeep Maji of Asansol Engineering College, registration number: 10800220046, has successfully completed a project Movies Reccomendation system using Python and Pandas under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Mr. Soumya Banerjee of Asansol Engineering College, registration number: 10800220056, has successfully completed a project on Movies Reccomendation system using Python and Pandas under the guidance of Prof. Arnab Chakraborty.

-- -

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Mr. Subinaya Mukherjee of Asansol Engineering College, registration number: 10800221140, has successfully completed a project on Movies Reccomendation system using Python and Pandas under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Mr. Simran Tiwary of Asansol Engineering College, registration number: 10800221142, has successfully completed a project on Movies Reccomendation system using Python and Pandas under the guidance of Prof. Arnab Chakraborty.

-- -

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Mr. Sudip Bhattacharjee of Asansol Engineering College, registration number: 10800221134, has successfully completed a project on Movies Reccomendation system using Python and Pandas under the guidance of Prof. Arnab Chakraborty.

--- -

Prof. Arnab Chakraborty

Globsyn Finishing School

THANKYOU