Volume 9, Issue 2, February – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24FEB1527

Netflix Movies Recommendation System

Koppadi. Bhavani[1]; Kottu. Aslesha Lakshmi Sai[2]
Swarnandhra College of Engineering and Technology

Abstract:- Netflix is a leading global entertainment company with over 247 million paying customers who can access TV series, films, and games in over 190+ countries. Many languages and genres are offered for the content. As much as they like, whenever and wherever they wish, participants are free to change their plans and play, pause, and resume viewing.

With the exponential growth of content on streaming platforms like Netflix, users often face the challenge of finding relevant and enjoyable movies.The "Cold-Start" issue is a recommendation system issue wherein new users or items have little data, making it challenging to provide appropriate recommendations.

In response, this paper proposes a data-driven movie recommendation system tailored for Netflix. Leveraging machine learning techniques, user preferences, and historical data, the model aims to enhance the user experience by providing personalized movierecommendations. This study explores algorithms such as collaborative filtering, content-based filtering, and hybrid models to achieve accurate and effective movie recommendations.

While content-based filtering uses item features to suggest shows or movies a user has previously enjoyed, collaborative filtering examines user behavior and preferences to produce suggestions. Even with insufficient data, the system can deliver more accurate recommendations for new users or things by combining various approaches. By making recommendations based on both item features and user behavior, this hybrid approach helps to reduce the cold start issue.

Keywords:- *Collaborative Filtering, Content-Based Filtering, Hybrid Models, Movie Recommendation Systems, Machine Learning.*

I. INTRODUCTION

Streaming platforms like Netflix have too much content, making it difficult for users to find movies that match their interests. This article addresses this challenge by introducing Netflix movie recommendations using machine learning. The system uses algorithms that analyze the user's behavior, preferences and historical data to create personalized recommendations, thereby improving the overall user experience.

The world's data output is increasing faster than our resources are available. We all experience the inundation of new books, journals, and conferences published every year. Technology has reduced barriers to distribution and dissemination of information.

Combination of hybrid model and integration filter to ensure RMSE while accepting relevant data. The experiment shows how machine learning can change the way customers rent and watch videos on streaming services.

II. LITERATURE SURVEY

- "Netflix Recommendation: Algorithm, Market Cap, and Innovation" by Carlos Gomez-Uribe and Neil Hunt provides an overview of Netflix's recommendations, including its algorithm, market cap, and innovation. The authors discuss the problems of creating strategies for a wide and diverse range of users and strategies for solving these problems. They also explain the impact of the offering on Netflix's growth and success.
- "Matrix Factorization Techniques for Recommended Systems" by Yehuda Koren, Robert Bell, and Chris Volinsky provides detailed information on matrix factorization techniques for recommended systems, including Recommended use by Netflix. The authors discuss the advantages of matrix factorization over other optimal algorithms, as well as the problems in developing matrix factorization models for large data sets. They also provide a detailed description of the Netflix Prize, a competition to create the best recommendations for Netflix.
- Sarwar, B. Karypis, G.Konstan, J. and Riedl, J. (2001) proposed a project-based filtering recommendation algorithm using knowledge discovery technology to address the problem of self-validation of information, products, or services during interaction. These systems, especially those based on k-nearest neighbor filtering, are gaining popularity on the Internet. In recent years, the tremendous increase in the amount of information available and the number of website visitors has created some significant challenges to consensus. These include: generating recommendations, making multiple recommendations per second for millions of users and projects, and achieving high service levels when data is scarce.
- As the Netflix Grand Prix showed, the matrix factorization model outperforms the original model. Those closest to the creation of suggestions, such as Koren, Y., Bell, R., and Volinsky, allow for the integration of additional information and provide implicit feedback regarding body and confidence and are

popular for the matrix factorization process. who attends?

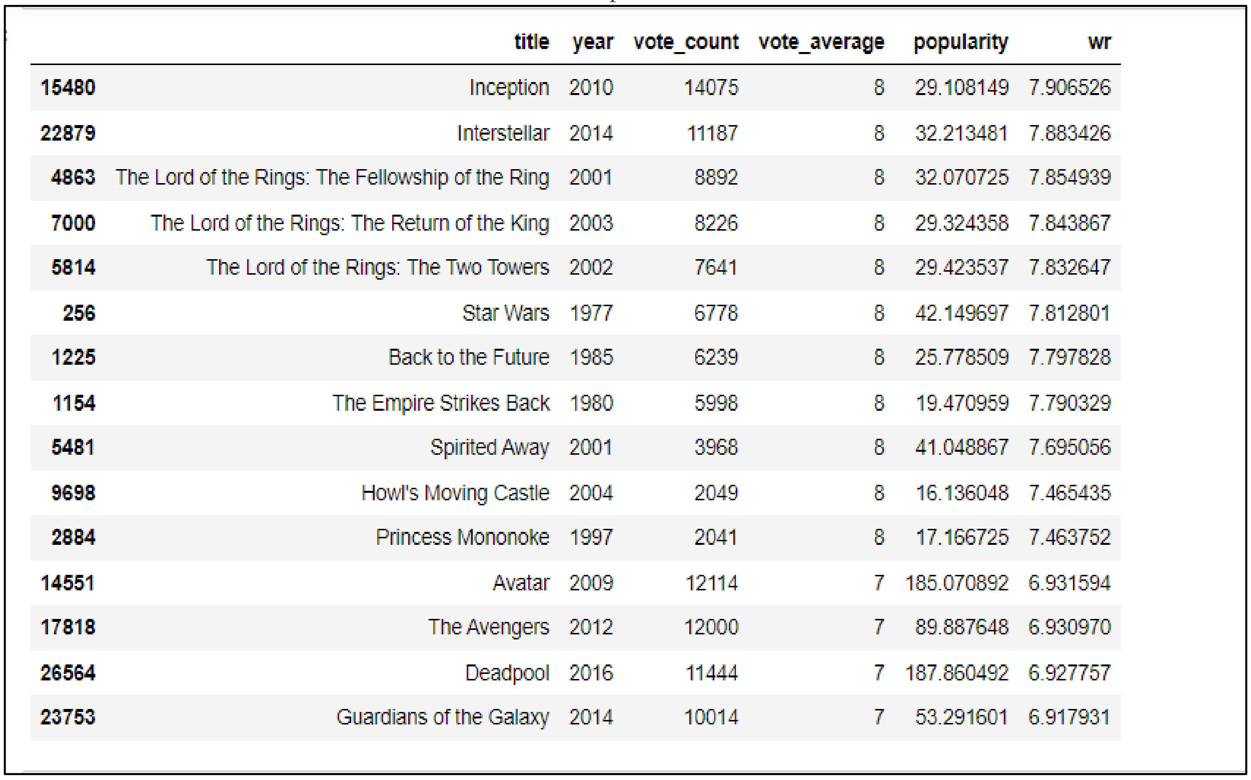
- Pazzani, M. J. and Billsus discuss contextual recommendations, systems that recommend products to users based on their descriptions and profile interests. Process-based recommendations can be used in many places, including recommending web pages, magazines, restaurants, TV shows, and products. Although the contents of various systems differ, content-based consensus sharing refers to creating user information that identifies objects that can be agreed upon, specifying products that the user likes, and creating a user profile that describes liked products. products. The type of product the user likes. kind of means. Compare the product with user data to determine what content to recommend. This profile is often created and updated in response to requests for products offered to users.

III. METHODOLOGY

This work adopts collaborative filtering, content-based filtering, and hybrid models to build consensus. Data from user interactions on Kaggle's Netflix site, including ratings, viewing history, and user data. The modeling process includes data prioritization, architecture design, and machine learning algorithms.

- *Procedure Model:* Collect data from Kaggle website and then pre-process (EDA) the data, which includes data maintenance and data analysis.

Table 1: Preprocessed Data



	title	year	vote_count	vote_average	popularity	wr
15480	Inception	2010	14075	8	29.108149	7.906526
22879	Interstellar	2014	11187	8	32.213481	7.883426
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	7.854939
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	7.843867
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.423537	7.832647
256	Star Wars	1977	6778	8	42.149697	7.812801
1225	Back to the Future	1985	6239	8	25.778509	7.797828
1154	The Empire Strikes Back	1980	5998	8	19.470959	7.790329
5481	Spirited Away	2001	3968	8	41.048867	7.695056
9698	Howl's Moving Castle	2004	2049	8	16.136048	7.465435
2884	Princess Mononoke	1997	2041	8	17.166725	7.463752
14551	Avatar	2009	12114	7	185.070892	6.931594
17818	The Avengers	2012	12000	7	89.887648	6.930970
26564	Deadpool	2016	11444	7	187.860492	6.927757
23753	Guardians of the Galaxy	2014	10014	7	53.291601	6.917931

NETFLIX movie recommendations can use 4 different recommendations based on different strategies and algorithms.

A. *Easy Expert:*

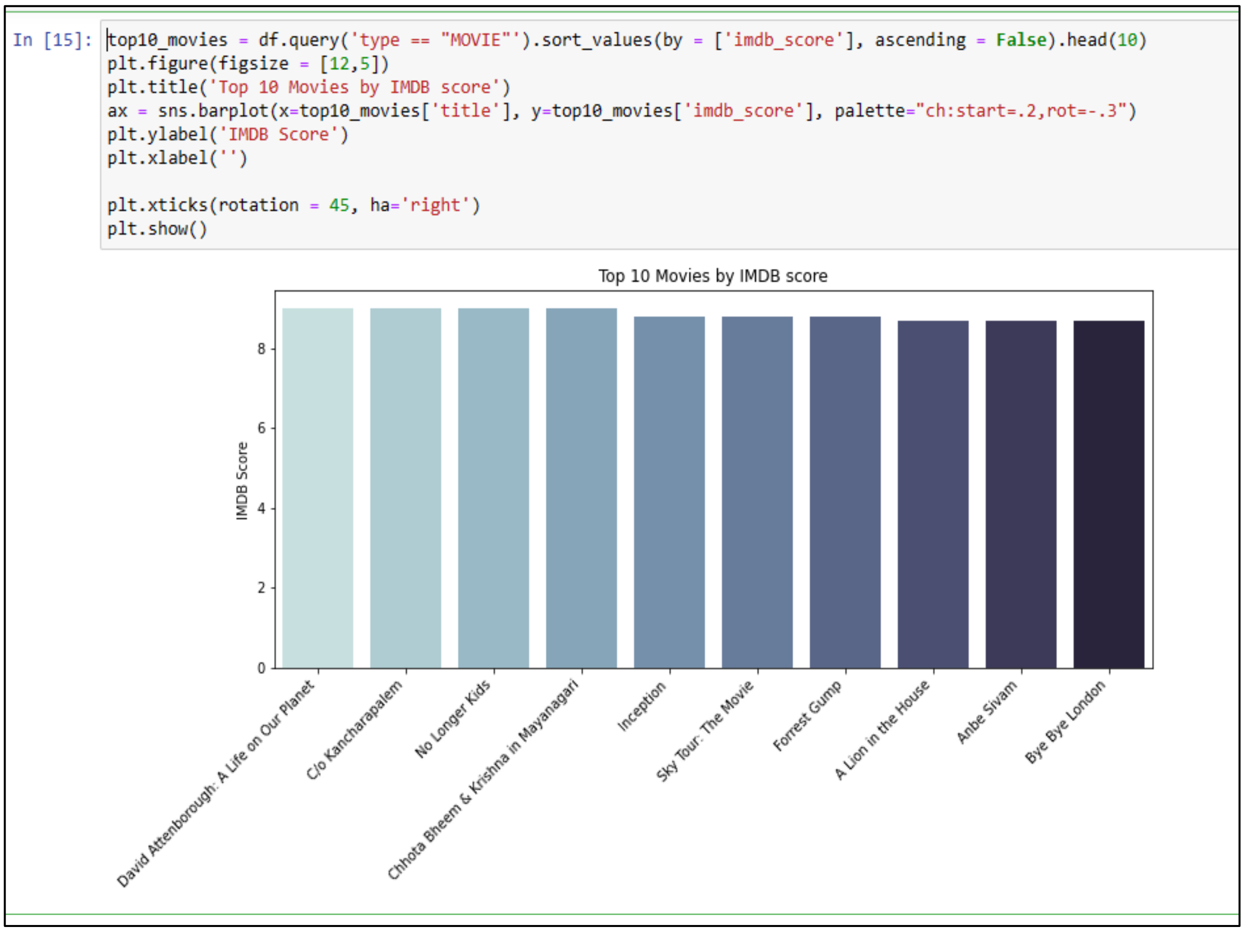
Use IMDB's weight index to calculate its score and finally rank it.

The expert gives recommendations to all users based on favorite videos and (sometimes) genres. The main idea behind this suggestion is that more popular and better reviewed videos are more likely to be watched by the audience. The model does not provide individual user-based recommendations.

The mathematical expression is as follows:
Weighted rating (WR) = (vv+m.R) + (mv+m.C)

Where,

- v is the number of votes given to the movie
- m is the number of votes given to the movie in the picture is the number Minimum number of votes should be set
- R is the average rating of the movie
- C is the average number of votes for all videos in the report



```
In [15]: top10_movies = df.query('type == "MOVIE"').sort_values(by = ['imdb_score'], ascending = False).head(10)
         plt.figure(figsize = [12,5])
         plt.title('Top 10 Movies by IMDB score')
         ax = sns.barplot(x=top10_movies['title'], y=top10_movies['imdb_score'], palette="ch:start=.2,rot=-.3")
         plt.ylabel('IMDB Score')
         plt.xlabel('')

         plt.xticks(rotation = 45, ha='right')
         plt.show()
```

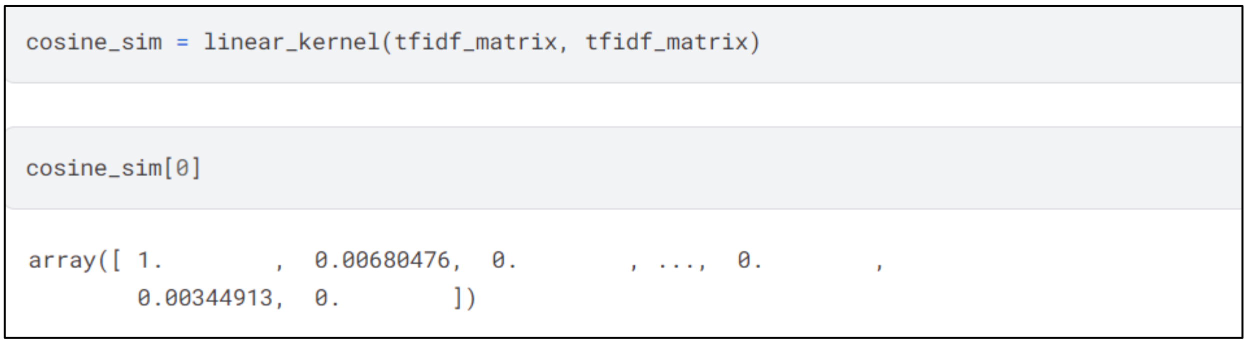
Fig. 1: Data Visualization

B. *Content-Based Recommendations:*

Simple recommendations have some limitations, so we can use content-based recommendations.

Calculates the similarity of videos based on some parameters and shows videos that are similar to certain videos the user liked. This is also called content filtering because we will use video metadata (or content) to create this engine.

- Cosine Similarity: I will use cosine similarity to calculate a numerical value that represents the similarity between two movies. Mathematically it is defined as: $\text{cosine}(x,y) = x.y^{\intercal}x.y$



```
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

```
cosine_sim[0]
```

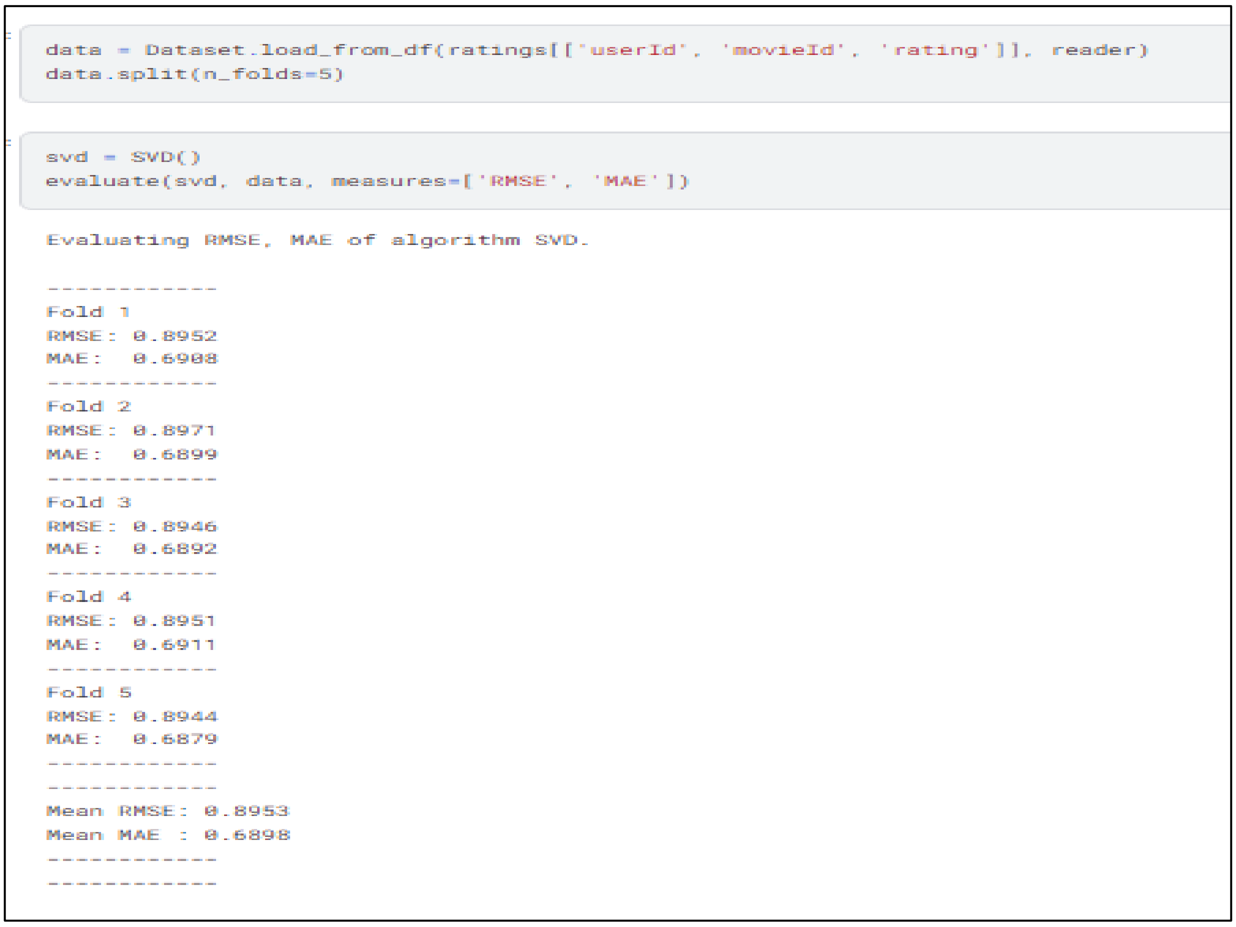
```
array([ 1.        ,  0.00680476,  0.        , ...,  0.        ,
        0.00344913,  0.        ])
```

Fig. 2: Sample Output

C. *Collaborative Filtering:*

Only movies that are close to certain movies can be recommended. So he can't capture the taste and can't give good recommendations on the product range.

Here we can use the Surprise library which uses very powerful techniques like Singular Value Decomposition (SVD) to reduce RMSE (Root Mean Square Error) and provide good recommendations.



```
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
data.split(n_folds=5)
```

```
svd = SVD()
evaluate(svd, data, measures=['RMSE', 'MAE'])
```

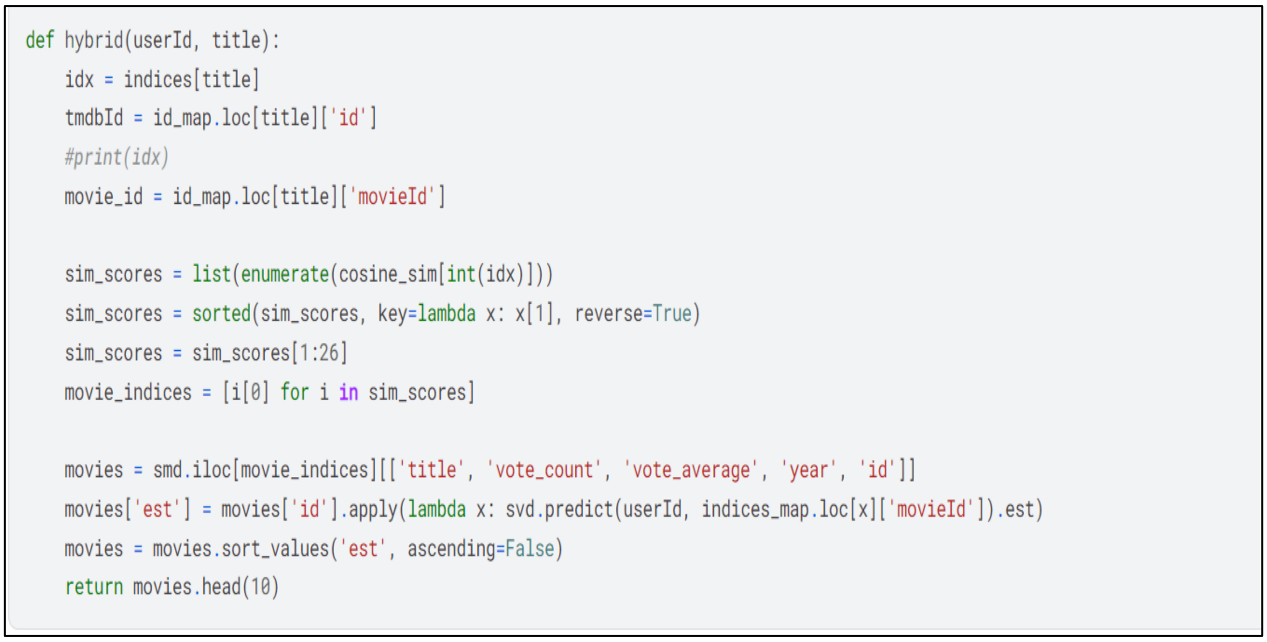
```
Evaluating RMSE, MAE of algorithm SVD.

------------
Fold 1
RMSE: 0.8952
MAE:  0.6908
------------
Fold 2
RMSE: 0.8971
MAE:  0.6899
------------
Fold 3
RMSE: 0.8946
MAE:  0.6892
------------
Fold 4
RMSE: 0.8951
MAE:  0.6911
------------
Fold 5
RMSE: 0.8944
MAE:  0.6879
------------
------------
Mean RMSE: 0.8953
Mean MAE : 0.6898
------------
------------
```

- Output: We obtained a root mean square error of 0.8963, which is good enough for our case.

D. Hybrid Recommender:

A simple hybrid recommender that combines ideas we use in content-based and collaborative filter-based engines.



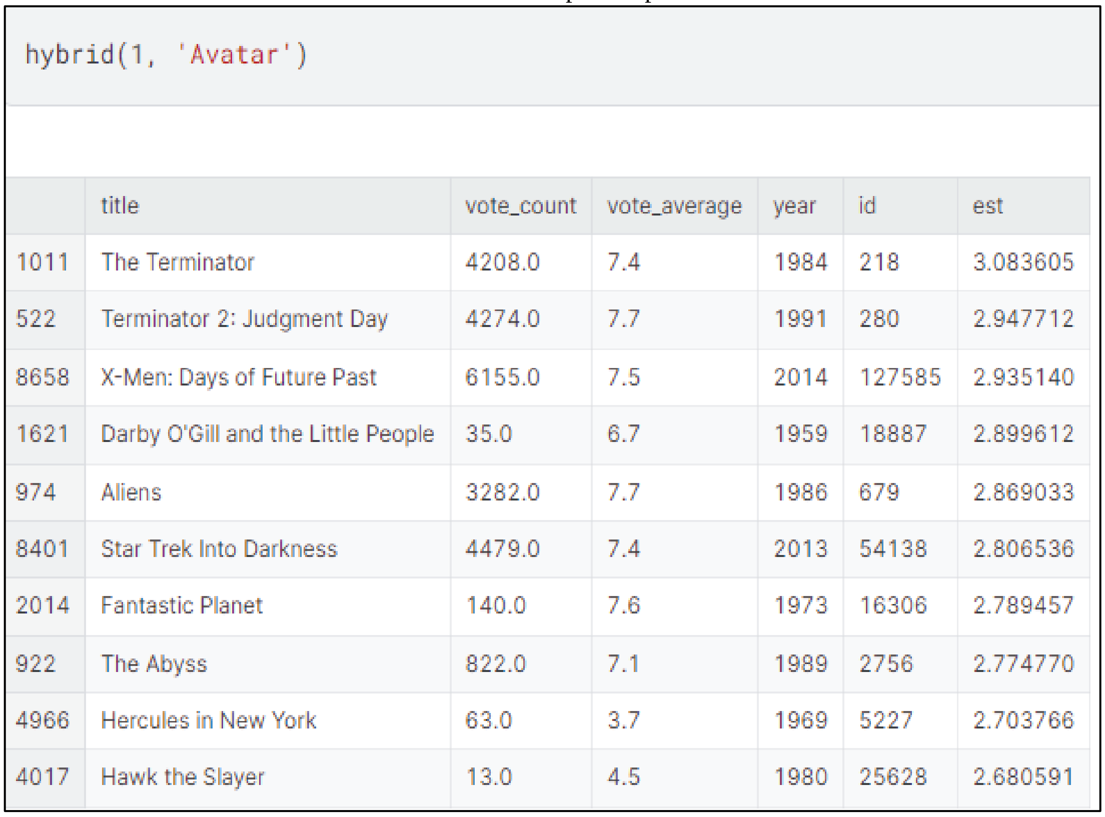
```
def hybrid(userId, title):
    idx = indices[title]
    tmdbId = id_map.loc[title]['id']
    #print(idx)
    movie_id = id_map.loc[title]['movieId']

    sim_scores = list(enumerate(cosine_sim[int(idx)]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:26]
    movie_indices = [i[0] for i in sim_scores]

    movies = smd.iloc[movie_indices][['title', 'vote_count', 'vote_average', 'year', 'id']]
    movies['est'] = movies['id'].apply(lambda x: svd.predict(userId, indices_map.loc[x]['movieId']).est)
    movies = movies.sort_values('est', ascending=False)
    return movies.head(10)
```

➢ *Sample Code for Hybrid Recommendation:*

Table 2: Sample Output



```
hybrid(1, 'Avatar')
```

	title	vote_count	vote_average	year	id	est
1011	The Terminator	4208.0	7.4	1984	218	3.083605
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	2.947712
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	2.935140
1621	Darby O'Gill and the Little People	35.0	6.7	1959	18887	2.899612
974	Aliens	3282.0	7.7	1986	679	2.869033
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	2.806536
2014	Fantastic Planet	140.0	7.6	1973	16306	2.789457
922	The Abyss	822.0	7.1	1989	2756	2.774770
4966	Hercules in New York	63.0	3.7	1969	5227	2.703766
4017	Hawk the Slayer	13.0	4.5	1980	25628	2.680591

IV. CONCLUSION

In this example, we create 4 different recommendations based on different strategies and algorithms. These are as follows:

- Simple Recommendations: This system uses all TMDB vote counts and average votes to create charts of general videos and specific videos. IMDB's weighting system is used to evaluate ratings and final ratings.
- Content-Based Recommendations: We create two content-based engines; one uses movie content and tagline as input, while the other uses metadata like actors, staff, genre, and content to make predictions. We've also created simple filters to give higher priority to videos with more votes and higher ratings.
- Collaborative Filter: We use the powerful Surprise library to create a collaborative filter based on the value of parsing. The resulting RMSE is less than 1 and provides a predictive rating for the engine, user, and video.
- Hybrid Engine: We combined ideas from content and collaborative filtering to create an engine that provides recommendations to specific users based on their estimated audience numbers.

REFERENCES

[1]. "Netflix Recommendation System: Algorithms, Business Value, and Innovation" by Carlos Gomez-Uribe and Neil Hunt (https://dl.acm.org/doi/10.1145/2827872)
[2]. "Matrix Factorization Techniques for Recommender Systems" by Yehuda Koren, Robert Bell, and Chris Volinsky (https://ieeexplore.ieee.org/document/5197422)
[3]. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (WWW '01).
[4]. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
[5]. Pazzani, M. J., &Billsus, D. (2007). Content-based recommendation systems. In The Adaptive Web (pp. 325-341). Springer.