# Movie Recommendation Using Content-Based and Collaborative Filtering Approach

Anjali Jha[1], Nidhi Agarwal[1,2], Devendra K. Tayal[1], Vrinda Abrol[1(✉)], Deepakshi[1], Yashica Garg[1], and Anushka[1]

[1] Department of Computer Science and Engineering, Indira Gandhi Delhi Technical University for Women, Kashmere Gate, New Delhi, India
{anjali148btcse20,vrinda047btcse21,deepakshi120btcse21, yashica048btcse21,anushka017btcse21}@igdtuw.ac.in

[2] Department of Computer Science and Engineering, Delhi Technical Campus, Greater Noida, India
n.agarwal@delhitechnicalcampus.ac.in

**Abstract.** A recommender system is one that tries to anticipate or filter preferences based on the user's selections. Films, music, journalism, publications, scientific papers and items in general all make use of recommender systems. We're building a recommendation system with Python and Pandas in this model. Utilizing an approach called content-based filtering, which is based on the information provided about the items, the system makes suggestions for movies that are comparable to those that a user has previously enjoyed. The information we know about the items and the user's previous choices are used to calculate this similarity. Combining collaborative filtering and content-based filtering is used to overcome the shortcomings of these two types of filtering generally so that a better recommendation system may be created.

**Keywords:** Pandas · Python · Recommendation System · Content-Based Filtering · Collaborative filtering · Movie Recommendation

## 1 Introduction

Users in the modern world, where the internet has become an integral part of daily life, struggle to make decisions since there are so many options available. There is too much information available online, whether you're looking for a hotel or solid investment opportunities. To assist users companies have implemented recommendation systems [1] for dealing with the information explosion directing their clients.

Numerous businesses, including Facebook, which suggests friends, LinkedIn, which suggests jobs, Pandora, which suggests music, Netflix, which suggests shows, and Amazon, which suggests purchases, among others, employ recommendation systems to boost their profits and help their clients. [2, 14, 15].


S. Nandan Mohanty et al. (Eds.): ICISML 2022, LNICST 470, pp. 439–450, 2023.
https://doi.org/10.1007/978-3-031-35078-8_37

Recommendation systems help the users to find and select items (e.g., books, movies, restaurants) from the wide collection available on the web or in other electronic information sources [3].

A movie recommendation system's fundamental idea is pretty straightforward. Every recommender system primarily consists of two components: users and items. Users receive movie predictions from the system, and the actual movies are the items [4]. This approach relies on user ratings for specific persons. It determines the commonalities among various users and then suggests movies to individuals with similar likes based on their ratings. [5, 16, 17].

In content-based recommendations, products that match the user's preferences are suggested, while in collaborative suggestions, individuals with similar tastes are found and recommended content they have enjoyed. Later, as the recommender system developed, a hybrid method that combines two or more techniques was created [6, 18, 19].

The movies dataset [7] includes the metadata for 45,000 films that were released on or before July 2017 and are listed in the MovieLens [8] database. Features include genre, title, price, gross, release dates, languages, production firms and nations, vote counts and averages from TMDB [9, 20, 21], and other information.

We use the dataset's significant attributes to create a recommender system that puts forward the most suitable movies for new customers streaming them from a particular platform. Due to the fact that new members lack a watch history, we must find and recommend highly regarded films that new users are likely to enjoy and stick with the streaming service [22].

Our goal is to create three straightforward recommender systems that suggest:

1. 20 most popular movies, in order.
2. The top 20 films by average weighted rankings.
3. 20 most highly rated films in a particular genre.

## 2 Experiment and Analysis

### 2.1 Data Cleaning

We will produce a subset of this dataset with crucial features like genres, id, popularity, title, vote average, and vote counts because our dataset includes many features that are not helpful in creating the above recommenders.

It is crucial to look for any duplicate entries for a movie in the database when making recommendations. Check and delete any duplicate entries in the id column since two movies may share the same name but have distinct ids.

Dropping the null values from the dataset is the next step. The "id" and "popularity" columns' data types must be "int" and "float" respectively. Let's use the "astype()" function to change the data type of these columns.

We must change the genre field into a list that simply contains the names of the genres that the movies come under in order to provide movie recommendations and pick the best films in each category. We accomplish this by appending the genre name to a list by retrieving it from the appropriate list of dictionaries. The genres section has been updated and now includes a list of the genres that each individual movie belongs to.

### 2.2 Exploding DataFrame

Let's get another selection of the dataset where the genres column would only have a single value rather than a list before creating some recommenders. For example (Fig. 1):

| | genres | id | popularity | title | vote_average | vote_count |
|---|---|---|---|---|---|---|
| 0 | [Animation, Comedy, Family] | 862 | 21.946943 | Toy Story | 7.7 | 5415.0 |
| 1 | [Adventure, Fantasy, Family] | 8844 | 17.015539 | Jumanji | 6.9 | 2413.0 |

**Fig. 1.** Subset Of DataFrame with List of Genres

We want the list to be expanded like (Fig. 2):

| | genres | id | popularity | title | vote_average | vote_count |
|---|---|---|---|---|---|---|
| 0 | Animation | 862 | 21.946943 | Toy Story | 7.7 | 5415.0 |
| 1 | Comedy | 862 | 21.946943 | Toy Story | 7.7 | 5415.0 |
| 2 | Family | 862 | 21.946943 | Toy Story | 7.7 | 5415.0 |
| 3 | Adventure | 8844 | 17.015539 | Jumanji | 6.9 | 2413.0 |
| 4 | Fantasy | 8844 | 17.015539 | Jumanji | 6.9 | 2413.0 |
| 5 | Family | 8844 | 17.015539 | Jumanji | 6.9 | 2413.0 |

**Fig. 2.** Subset Of DataFrame with one Genre per Movie in a row

Utilize the explode() method of the pandas DataFrame to expand the list of genres [10]. This method turns each item in a list-like structure into a row. DataFrame exploded listings to cells of the subset columns are obtained. This subset can be used for a variety of data visualisation tasks.

### 2.3 Recommendation Based on Popularity

Let's simply sort the subset DataFrame by the "popularity" column to present the top 20 movies to the new subscriber (Fig. 3).

Let's use the Seaborn module to create a barplot (Fig. 4) to further illustrate this DataFrame. Place the terms "popularity" and "title" on the x and y – axes respectively. Minions is the film that is the most well-liked across all genres. This list of movies can be suggested to a new subscriber and functions similarly to the YouTube trending page.

### 2.4 Genre Based Movie Distribution

Let's examine this DataFrame, where the genre field only contains one genre, in greater detail in order to offer recommendations that are relevant to that genre.

First, let's look at the amount of entries and null values in the genres_subset_df DataFrame. There are 2442 films for which no genre has been designated. Since these null records only make up roughly 2.6% of the dataset's total listed movies, let's remove

| | genres | id | popularity | title | vote_average | vote_count |
|---|---|---|---|---|---|---|
| 30700 | [Family, Animation, Adventure, Comedy] | 211672 | 547.488298 | Minions | 6.4 | 4729.0 |
| 33356 | [Action, Adventure, Fantasy] | 297762 | 294.337037 | Wonder Woman | 7.2 | 5025.0 |
| 42222 | [Family, Fantasy, Romance] | 321612 | 287.253654 | Beauty and the Beast | 6.8 | 5530.0 |
| 43644 | [Action, Crime] | 339403 | 228.032744 | Baby Driver | 7.2 | 2083.0 |
| 24455 | [Adventure, Family, Animation, Action, Comedy] | 177572 | 213.849907 | Big Hero 6 | 7.8 | 6289.0 |
| 26564 | [Action, Adventure, Comedy] | 293660 | 187.860492 | Deadpool | 7.4 | 11444.0 |
| 26566 | [Action, Adventure, Comedy, Science Fiction] | 283995 | 185.330992 | Guardians of the Galaxy Vol. 2 | 7.6 | 4858.0 |
| 14551 | [Action, Adventure, Fantasy, Science Fiction] | 19995 | 185.070892 | Avatar | 7.2 | 12114.0 |
| 24351 | [Action, Thriller] | 245891 | 183.870374 | John Wick | 7.0 | 5499.0 |
| 23675 | [Mystery, Thriller, Drama] | 210577 | 154.801009 | Gone Girl | 7.9 | 6023.0 |
| 24873 | [Science Fiction, Adventure, Thriller] | 131631 | 147.098006 | The Hunger Games: Mockingjay - Part 1 | 6.6 | 5767.0 |
| 44274 | [Drama, Science Fiction, War] | 281338 | 146.161786 | War for the Planet of the Apes | 6.7 | 1675.0 |
| 26567 | [Adventure, Action, Science Fiction] | 271110 | 145.882135 | Captain America: Civil War | 7.1 | 7462.0 |
| 292 | [Thriller, Crime] | 680 | 140.950236 | Pulp Fiction | 8.3 | 8670.0 |
| 26560 | [Adventure, Action, Fantasy, Comedy] | 166426 | 133.827820 | Pirates of the Caribbean: Dead Men Tell No Tales | 6.6 | 2814.0 |
| 12481 | [Drama, Action, Crime, Thriller] | 155 | 123.167259 | The Dark Knight | 8.3 | 12269.0 |
| 536 | [Science Fiction, Drama, Thriller] | 78 | 96.272374 | Blade Runner | 7.9 | 3833.0 |
| 17818 | [Science Fiction, Action, Adventure] | 24428 | 89.887648 | The Avengers | 7.4 | 12000.0 |
| 43286 | [Action, Animation, Comedy, Family] | 268531 | 88.561239 | Captain Underpants: The First Epic Movie | 6.5 | 159.0 |
| 33361 | [Drama, Thriller, Science Fiction] | 339988 | 88.439243 | The Circle | 5.4 | 1015.0 |

**Fig. 3.** Top 20 Movies based on Popularity in a Tabular Form

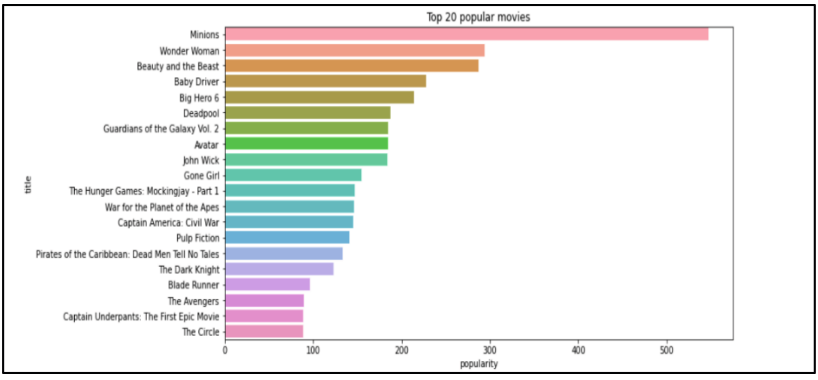


**Fig. 4.** Bar Graph showing the top 20 movies and their popularity

them from the DataFrame. Next, use the value_counts() function to determine how many movies there are in each genre. The result is a Table which displays the data. By utilising the seaborn module, we can also create a barplot to visualise this genre distribution (Fig. 5).

Here, we can see that the genre of drama has the most films, followed by comedy and so forth.

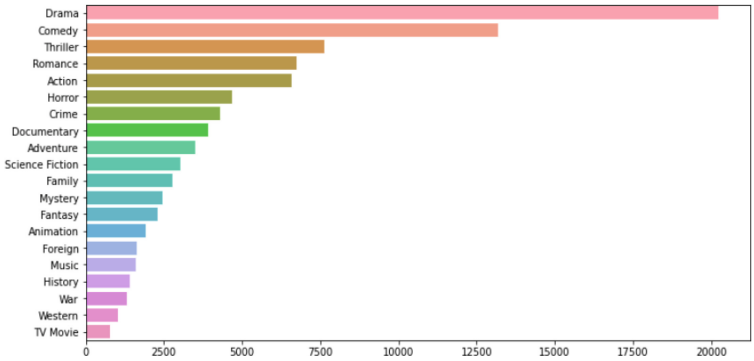


**Fig. 5.** Bar Plot showing the number of movies in each genre

### 2.5 Analyzing Genre Based Popularity

By examining the following dataset characteristics, one can determine the viewpoint of the audience being popularity, vote_average, and vote_count.

Let's create a pivot table to get the average values for popularity, vote average, and vote count across all genres. Make a new DataFrame with just the columns like genres, popularity, vote_average and vote_count from the subset of genres DataFrame. Utilize the pandas module's pivot table() function and pass the aforementioned freshly formed DataFrame and index = ['genres'] as inputs. This will show the average values for the vote_average, vote_count, and popularity columns for each genre.

Reset the index of the generated DataFrame using the reset index() function, then display the resulting DataFrame (Fig. 6).

| | genres | popularity | vote_average | vote_count |
|---|---|---|---|---|
| 0 | Action | 4.773008 | 5.584932 | 259.232777 |
| 1 | Adventure | 6.001218 | 5.687593 | 410.936390 |
| 2 | Animation | 4.709924 | 6.275544 | 234.534715 |
| 3 | Comedy | 3.230777 | 5.715642 | 109.106937 |
| 4 | Crime | 4.142982 | 5.877649 | 176.028346 |
| 5 | Documentary | 0.948341 | 5.823028 | 12.836641 |
| 6 | Drama | 3.010777 | 5.905888 | 96.570074 |
| 7 | Family | 4.728138 | 5.753632 | 242.281171 |
| 8 | Fantasy | 5.366410 | 5.790775 | 333.171503 |
| 9 | Foreign | 0.761960 | 5.735639 | 7.984558 |
| 10 | History | 3.466370 | 6.154220 | 107.364092 |

**Fig. 6.** Table displaying the highest mean popularity score of different genres

The top 5 most renowned genres, as determined by mean popularity score, are Adventure, Fantasy, Science Fiction, Action and Family. Therefore, since the viewer is most

likely to enjoy these recommendations, we can recommend films from these genres to them. Similarly, we can obtain result of which genre received the most votes. The top 5 genres with the most votes come out to be as follows:

1. Adventure
2. Fantasy
3. Science fiction
4. Action
5. Family

When looking at the data for genres which have the highest vote average, the top 5 entries are:

1. Animation
2. History
3. War
4. Drama
5. Music

These figures and lists can be utilized to suggest movies from suitable genres to new users.

### 2.6 Recommendation Based on Ratings

Now, sort the subset movie DataFrame by the "vote average" column to create a movie recommender based on the average rating of the film. Many of the highly rated films (vote average = 10.0) have vote count = 1.0, as we can see. This indicates that only 1 subscriber gave these movies a rating of 10, which is significant.

As a result, we are unable to declare these films to be the highest rated films because so few users gave their ratings. As a result, using the vote average column alone will not allow us to suggest movies to a new subscriber. In other words, the following drawbacks apply when employing a basic average rating as a metric:

It does not take into consideration the popularity of a movie. A movie with a rating of 9 from 20 voters will be considered better than a movie with a rating of 8 from 10,000 voters. On a related note, this metric will favour movies with a smaller number of voters with skewed or extremely high ratings.

Because of this, it might be challenging to determine a movie's quality when there are so few voters. We must take into account both the average rating and the total number of voters in order to address these issues. We shall employ IMDB's weighted rating system for this.

**Weighted Average Rating:**

Let us use the weighted average rating formula to recommend top rated movies to the user [3]. This technique is employed by IMDB to determine its top movies chart. The weighted rating approach ensures that a film with only a few ratings, all at 10, would not rank above "the Godfather", for example, with a 9.2 average from over 500,000 ratings. It is mathematically represented as:

$$W = Rv + Cmv + m \tag{1}$$

where,

W is the Weighted Rating.

R is the average rating of the movie.

v is the number of votes for the movie.

m is the minimum votes required to be listed in the top movies list.

C denotes the mean vote across the whole dataset.

After calculating the mean vote across the whole dataset, we observe that the average rating of a movie is around 5.6 on a scale of 10.

Lastly, we need to calculate m i.e. the minimum votes required to be listed in the top movies list. For our recommender, we will use cutoff m as the 90th percentile. A 90% quantile means that in order to assign a weighted rating to a movie, it must have more than 90% votes.

The minimum number of votes, m comes out to be 160 which means any movie which has received less than 160 votes will not be considered for weighted rating and hence will not be recommended at all. Since now we have the m, let us simply filter the movies whose vote_count is greater than or equal to 160 using the steps given below:

- Use DataFrame.copy() function to create a copy of the original movie subset DataFrame followed by DataFrame.loc[] function to filter movies.
- Specify the following condition inside the loc[] function to obtain only those movies whose vote_count is greater than or equal to m: mov_subset_df['vote_count'] > = m

Thus, the qualified_movies_df DataFrame consists of all the movies which have received more than the minimum number of votes. Now, final step is to calculate the weighted rating for each qualified movie. To do this, we will:

- Define a function weighted_rating and pass the corresponding DataFrame as input. Inside this function:
- Determine v using the vote_count column of the DataFrame.
- Determine R using the vote_average column of the DataFrame.
- Calculate the weighted average W using the formula and return the calculated weighted rating.

The above function will calculate the weighted rating for each qualified movie. Let us add this weighted rating as a separate column to the qualified movies DataFrame using the steps given below:

- Use DataFrame.apply() function [12] and pass the above weighted_rating function as input to the apply() function. This will invoke the weighted_rating() function for each qualified movie.
- Set axis = 1 as we want to apply the weighted_rating function to each row of the DataFrame.
- Finally, display the first 5 values of the resulting DataFrame (Fig. 7).

Now, to display the top 25 highly rated movies:

- Sort the qualified movies DataFrame based on weighted_rating column in de-scending order.

| | genres | id | popularity | title | vote_average | vote_count | weighted_rating |
|---|---|---|---|---|---|---|---|
| 0 | [Animation, Comedy, Family] | 862 | 21.946943 | Toy Story | 7.7 | 5415.0 | 7.640257 |
| 1 | [Adventure, Fantasy, Family] | 8844 | 17.015539 | Jumanji | 6.9 | 2413.0 | 6.820300 |
| 4 | [Comedy] | 11862 | 8.387519 | Father of the Bride Part II | 5.7 | 173.0 | 5.660759 |
| 5 | [Action, Crime, Drama, Thriller] | 949 | 17.924927 | Heat | 7.7 | 1886.0 | 7.537211 |
| 8 | [Action, Adventure, Thriller] | 9091 | 5.231580 | Sudden Death | 5.5 | 174.0 | 5.556685 |

**Fig. 7.** The weighted rating of the movies in the DataFrame.

- Display the first 25 entries of the sorted DataFrame (Fig. 8).

| | genres | id | popularity | title | vote_average | vote_count | weighted_rating |
|---|---|---|---|---|---|---|---|
| 314 | [Drama, Crime] | 278 | 51.645403 | The Shawshank Redemption | 8.5 | 8358.0 | 8.445871 |
| 834 | [Drama, Crime] | 238 | 41.109264 | The Godfather | 8.5 | 6024.0 | 8.425442 |
| 10309 | [Comedy, Drama, Romance] | 19404 | 34.457024 | Dilwale Dulhania Le Jayenge | 9.1 | 661.0 | 8.421477 |
| 12481 | [Drama, Action, Crime, Thriller] | 155 | 123.167259 | The Dark Knight | 8.3 | 12269.0 | 8.265479 |
| 2843 | [Drama] | 550 | 63.869599 | Fight Club | 8.3 | 9678.0 | 8.256387 |
| 292 | [Thriller, Crime] | 680 | 140.950236 | Pulp Fiction | 8.3 | 8670.0 | 8.251408 |
| 522 | [Drama, History, War] | 424 | 41.725123 | Schindler's List | 8.3 | 4436.0 | 8.206643 |
| 23673 | [Drama] | 244786 | 64.299990 | Whiplash | 8.3 | 4376.0 | 8.205408 |
| 5481 | [Fantasy, Adventure, Animation, Family] | 129 | 41.048867 | Spirited Away | 8.3 | 3968.0 | 8.196059 |
| 2211 | [Comedy, Drama] | 637 | 39.394970 | Life Is Beautiful | 8.3 | 3643.0 | 8.187177 |
| 1178 | [Drama, Crime] | 240 | 36.629307 | The Godfather: Part II | 8.3 | 3418.0 | 8.180082 |
| 1152 | [Drama] | 510 | 35.529554 | One Flew Over the Cuckoo's Nest | 8.3 | 3001.0 | 8.164262 |
| 351 | [Comedy, Drama, Romance] | 13 | 48.307194 | Forrest Gump | 8.2 | 8147.0 | 8.150275 |
| 1154 | [Adventure, Action, Science Fiction] | 1891 | 19.470959 | The Empire Strikes Back | 8.2 | 5998.0 | 8.132922 |
| 1176 | [Drama, Horror, Thriller] | 539 | 36.826309 | Psycho | 8.3 | 2405.0 | 8.132722 |
| 18465 | [Drama, Comedy] | 77338 | 16.086919 | The Intouchables | 8.2 | 5410.0 | 8.125841 |
| 40251 | [Romance, Animation, Drama] | 372058 | 34.461252 | Your Name. | 8.5 | 1030.0 | 8.112548 |
| 289 | [Thriller, Crime, Drama] | 101 | 20.477329 | Leon: The Professional | 8.2 | 4293.0 | 8.107238 |
| 3030 | [Fantasy, Drama, Crime] | 497 | 19.966780 | The Green Mile | 8.2 | 4166.0 | 8.104515 |
| 1170 | [Drama, Crime] | 769 | 15.424092 | GoodFellas | 8.2 | 3211.0 | 8.077464 |
| 2216 | [Drama] | 73 | 18.157166 | American History X | 8.2 | 3120.0 | 8.074065 |
| 15480 | [Action, Thriller, Science Fiction, Mystery, A... | 27205 | 29.108149 | Inception | 8.1 | 14075.0 | 8.072106 |
| 22879 | [Adventure, Drama, Science Fiction] | 157336 | 32.213481 | Interstellar | 8.1 | 11187.0 | 8.065007 |
| 7000 | [Adventure, Fantasy, Action] | 122 | 29.324358 | The Lord of the Rings: The Return of the King | 8.1 | 8226.0 | 8.052651 |
| 256 | [Adventure, Action, Science Fiction] | 11 | 42.149697 | Star Wars | 8.1 | 6778.0 | 8.042769 |

**Fig. 8.** The first 25 movies ranked based on the weighted rating assigned to them.

## 2.7 Genre-specific Recommendation

Since everyone has a preferred genre, the user may occasionally want to navigate to a specific genre and receive recommendations based on the genre they have chosen. So, let us modify the recommendation engine to recommend movies based on the genre you have chosen. Firstly, we expand the genres column of qualified movies. Victimization the explode() function, We will get one genre instead of an inventory of genres from an information Frame. To begin, expand the genres column of qualified movies. Using the

explode() function, you can get a single genre rather than a list of genres from a Data Frame (Fig. 9).

| | genres | id | popularity | title | vote_average | vote_count | weighted_rating |
|---|---|---|---|---|---|---|---|
| 0 | Drama | 278 | 51.645403 | The Shawshank Redemption | 8.5 | 8358.0 | 8.445871 |
| 1 | Crime | 278 | 51.645403 | The Shawshank Redemption | 8.5 | 8358.0 | 8.445871 |
| 2 | Drama | 238 | 41.109264 | The Godfather | 8.5 | 6024.0 | 8.425442 |
| 3 | Crime | 238 | 41.109264 | The Godfather | 8.5 | 6024.0 | 8.425442 |
| 4 | Comedy | 19404 | 34.457024 | Dilwale Dulhania Le Jayenge | 9.1 | 661.0 | 8.421477 |

**Fig. 9.** Screenshot showing DataFrame which consists of a single genre of the movies along with the obtained weighted rating.

Let us build a custom function that takes a genre as input and returns a DataFrame of the top 25 most heavily weighted movies belonging to that genre to recommend movies based on the genre the user chooses or prefers.

To recommend movies by genre:

Declare a genre recommender() function that takes a fav_genre as input. In this function:

Create the DataFrame Recommended_df returned by this function.

- In a Genre-based DataFrame, pass the following condition to filter movies based on the selected Genre.
- genre-based_df[‘gen’] = = preferred genre
- Returns the first 25 records in a Recommended_df data frame.

## Thriller Movies Recommendation

- Let us test the feature by getting recommendations for thriller movies.
- Get a thrmovie recommendation and test its features.
- To do this, call genre_recommender() and pass “Thriller” as input to this function to get a thriller movie recommendation (Fig. 10).

| | genres | id | popularity | title | vote_average | vote_count | weighted_rating |
|---|---|---|---|---|---|---|---|
| 10 | Thriller | 155 | 123.167259 | The Dark Knight | 8.3 | 12269.0 | 8.265479 |
| 12 | Thriller | 680 | 140.950236 | Pulp Fiction | 8.3 | 8670.0 | 8.251408 |
| 35 | Thriller | 539 | 36.826309 | Psycho | 8.3 | 2405.0 | 8.132722 |
| 41 | Thriller | 101 | 20.477329 | Leon: The Professional | 8.2 | 4293.0 | 8.107238 |
| 51 | Thriller | 27205 | 29.108149 | Inception | 8.1 | 14075.0 | 8.072106 |
| 66 | Thriller | 807 | 18.457430 | Se7en | 8.1 | 5915.0 | 8.034639 |
| 70 | Thriller | 274 | 4.307222 | The Silence of the Lambs | 8.1 | 4549.0 | 8.015679 |
| 78 | Thriller | 77 | 15.450789 | Memento | 8.1 | 4168.0 | 8.008256 |
| 80 | Thriller | 694 | 19.611589 | The Shining | 8.1 | 3890.0 | 8.001959 |
| 82 | Thriller | 500 | 12.220340 | Reservoir Dogs | 8.1 | 3821.0 | 8.000259 |
| 87 | Thriller | 629 | 16.302466 | The Usual Suspects | 8.1 | 3334.0 | 7.986357 |
| 89 | Thriller | 264644 | 12.443291 | Room | 8.1 | 2838.0 | 7.967556 |
| 100 | Thriller | 567 | 17.911314 | Rear Window | 8.2 | 1531.0 | 7.955726 |
| 111 | Thriller | 205596 | 31.595940 | The Imitation Game | 8.0 | 5895.0 | 7.937066 |
| 120 | Thriller | 1124 | 16.945560 | The Prestige | 8.0 | 4510.0 | 7.918401 |
| 128 | Thriller | 111 | 11.299673 | Scarface | 8.0 | 3017.0 | 7.880054 |
| 143 | Thriller | 16869 | 16.895640 | Inglourious Basterds | 7.9 | 6598.0 | 7.845980 |
| 149 | Thriller | 210577 | 154.801009 | Gone Girl | 7.9 | 6023.0 | 7.840956 |
| 170 | Thriller | 670 | 10.616859 | Oldboy | 8.0 | 2000.0 | 7.823580 |
| 175 | Thriller | 348 | 23.377420 | Alien | 7.9 | 4564.0 | 7.822721 |
| 178 | Thriller | 1422 | 18.515448 | The Departed | 7.9 | 4455.0 | 7.820895 |
| 185 | Thriller | 78 | 96.272374 | Blade Runner | 7.9 | 3833.0 | 7.808573 |
| 200 | Thriller | 146233 | 11.962620 | Prisoners | 7.9 | 3183.0 | 7.790796 |
| 224 | Thriller | 11324 | 15.813629 | Shutter Island | 7.8 | 6559.0 | 7.748048 |
| 246 | Thriller | 426 | 18.208220 | Vertigo | 8.0 | 1162.0 | 7.711749 |

**Fig. 10.** Screenshot showing Top 25 thriller movies having highest weighted rating.

## 3 Conclusion

Systems for recommending movies are incredibly helpful in our daily lives since they make evaluating movies faster and easier.

A filtration program called a recommendation system's major objective is to forecast a user's "rating" or "desire" for a particular domain-specific item or item. Since the domain-specific object in our case is a movie, the primary goal of our recommendation system is to identify and suggest just those films that a user would find appealing based on information about the user. Computer programs known as recommendation systems make suggestions to users based on a range of factors. These systems predict the goods that consumers are most likely to purchase and be interested in.

We Have Used Different Filtration Strategies [5].

- Content-based Filtering:- This filtration strategy is based on the item data provided. The algorithm suggests products that are like those that a user has previously liked. This similarity (generally cosine similarity) is calculated using information about the items as well as the user's previous preferences.

- Collaborative Filtering:-This filtration strategy is based on combining the user's behaviour and comparing it with other users' behaviour in the database [14]. This algorithm heavily relies on the history of all users. There are numerous approaches to implementing collaborative filtering, but the main concept to grasp is that in collaborative filtering, the data of multiple users influences the outcome of the recommendation. And does not rely on the data of a single user for modelling.

The movie recommender system provides users with a smart and personalised experience, assisting streaming media service providers in increasing user engagement. It also provides excellent movie recommendations and allows users to endlessly scroll through and watch movies one after the other. We observe that our system has correctly recommended top movies in the adventure genre. Hence, we note that the system accurately identified the best adventure films.

As a result, we were able to successfully construct the foundational recommendation engine for our new consumers. The accuracy of the movie description in the dataset is completely necessary for this kind of recommendation engine to function properly. Even if a film's genre classification is wrong, the system will still suggest it based on viewer votes. While this type of system is useful for new users, it is not a decent suggestion engine for those who have a large enough viewing history. Since better movies can be suggested depending on a user's viewing history. Most popular movie rating websites adapted the weighted rating method in some way to rate the films.

To coordinate the progress of the many approaches now being examined, it is crucial to identify appropriate methods for assessing and contrasting the outcomes obtained in this area. [13].

After completing informal evaluations, a small number of users responded favourably. For our system to be able to deliver more meaningful findings, we would like to have a larger data collection.

## References

1. https://www.geeksforgeeks.org/python-implementation-of-movie-recommender-system/
2. http://trailerpark.weebly.com/imdb-rating.html?source=post_page
3. Simpson, D.G.: Weighted Averages. Department of Physical Sciences and Engineering, Prince George's Community (2010)
4. Goyani, M., Chaurasiya, N.: A review of movie recommendation system: Limitations, Survey and Challenges. ELCVIA: Electr. Lett. Comput. Vis. Image Anal. **19**(3), 18–37 (2020)
5. Iacovou, N., Bergstrom, P., Suchak, M., Resnick, P., Riedl, J.: An open architecture for cooperative netnews filtering is called Grouplens. In: Proceedings of the 1994 ACM conference on Computer Supported Cooperative Work, pp. 175–186. ACM (1994)
6. Pradeep, N., et al.: Content based movie recommendation system. Int. J. Res. Ind. Eng. **9**(4), 337–348 (2020)
7. https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset
8. https://movielens.org/
9. https://www.themoviedb.org/
10. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.explode.html
11. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.apply.html

12. Varma, A.N., Petluri, K.: Movie Recommender System using critic consensus. In: 2021 International Conference on Advances in Computing, Communication, and Control (ICAC3), pp. 1–4 (2021). https://doi.org/10.1109/ICAC353642.2021.9697196
13. Asnicar, F.A., Tasso, C.: "ifWeb: a prototype of user model-based intelligent agent for document filtering and navigation in the world wide web. In: Sixth International Conference on User Modeling (1997)
14. Pandharbale, P. B., Mohanty, S. N., Jagadev, A.K.: QoS-Aware Web Services Recommendations Using Dynamic Clustering Algorithms. Int. J. Inf. Syst. Model. Design **13**(6), 1–16 (2022). https://doi.org/10.4018/IJISMD.301274. ISSN: 1947–8186
15. Sathe, P., Mohanty, S.N., JagdevNovel, A.K.: Clustering-based recommendation framework. Int. J. Syst. Dyn. Appl. **11**(5), oa1 (2021). ISSN: 2160–9772. https://doi.org/10.4018/IJSDA.20220901.oa1
16. Ganayak, M., Mohanty, S.N., Jagadev, A.K.: Agricultural recommendation system for crops using different machine learning regression methods. Int. J. Agricult. Environ. Inf. Syst. **12**(1), 1–20 (2021). ISSN: 1947–3192. https://doi.org/10.4018/IJAEIS
17. Choudhury, S.S., Mohanty, S.N., Jagadev, A.K.: Multimodal trust-based recommender system with machine learning approaches for movie recommendation. Int. J. Inf. Technol. **13**(2), 475–482 (2021). https://doi.org/10.1007/s41870-020-00553-2. ISSN: 0984–8332
18. Garanayak, M., Sahoo, S., Mohanty, S.N., Jagadev, A.K.: An automated recommender system for educational institute in India. EAI Endorsed Trans. Scalable Inf. Syst. **24**(2), 1–13 (2020). https://doi.org/10.4108/eai.13-7-2018.163155 (2020). ISSN: 2032–9407
19. Mohanty, S.N., ParvinVinoth Kumar, K.C.R., Rani, R.S.S., Lakshmanaprabu, S.K.: Optimal rough Fuzzy clustering for User Profile Ontology based Web Page Recommendation Analysis. J. Intell. Fuzzy Syst. **37**(1), 205–216 (2019). https://doi.org/10.3233/JIFS-179078 . ISSN:1875–8967
20. Garanayak, M., Mohanty, S.N., Jagadev, A.K., Sahoo, S.: Recommender system using item based collaborative filtering (cf) and k-means. Int. J. Knowl.-Based Intell. Eng. Syst. 23(2), 93–101 (2019). ISSN:1327–2314. https://doi.org/10.3233/KES-190402
21. Kuanr, M., Mohanty, S.N.: Location based personalized recommendation systems for the tourists in India. Int. J. Business Intell. Data Mining **17**(3), 377 – 392 (2020). https://doi.org/10.1504/IJBIDM.2020.109294
22. Kuanr, M., Mohanty, S.N., Sahoo, S., Rath, B.K.: Crop Recommender System for the Farmers using Mamdani fuzzy Inference Model. Int. J. Eng. Technol. **7**(4.15), 277–280 (2018)