Final Report

Project 19
Movie Recommendation System

By Jan Szachno and Aleksandra Głogowska

---

1. Task Description

The task is to develop a movie recommendation system using the IMDB dataset. It has to cluster similar movies based on their description and other metadata.
A recommendation system in AI is a type of technology that predicts and suggests items to users based on various forms of input data. These systems are widely used to enhance user experience and engagement by personalizing content and suggestions according to individual preferences and behaviors. The main goal of the task is to make a recommendation system based on two methods and compare them based on a metric.

2. Dataset Description

The first dataset that we decided to use consists of 1000 top rated movies and tv shows listed on the IMBD website. It comprises various features that can be leveraged for both exploratory analysis and feature engineering for the recommendation system:

- Poster Link - While visually appealing, this is typically not used in analysis but could be useful for user interface design, which in our case does not matter. Poster Link will be removed from our dataset due to its redundancy in the recommendation process.
- *Series Title* (also a movie title) - Is not really useful in the analysis, and will not provide any additional info about the trends in the dataset, but is crucial for the recommendation process. It will be used for identification and potentially for linking with other data sources.
- *Released Year* - It is useful for showing the trend of which movies are possibly more popular or better scored depending on the time of release. We can check if the older or newer movies, and series, are more often watched.
- *Certificate* - In a recommendation system refers to the movie's rating (e.g., G, PG, R), which is crucial for aligning content with audience age appropriateness and preferences. It allows the system to filter and recommend movies that suit the viewer's demographic, enhancing personalization and user satisfaction.
- *Runtime* - Could be correlated with genre or ratings, longer movies might perform differently in specific genres. It can be also a factor in the recommendation system, provided that we would have the personalised data of the user, in our case, runtime data will be probably of less consideration.
- *Genre* - This is the main grouping feature for the movie recommendation system. Usually, users pick movies based on their genres. Along with other important features, this will be the base for the recommendation algorithm.
- *IMDB_Rating* - Normally, this feature of the database is important in a recommendation system as it provides a quantifiable measure of a movie's overall viewer approval and popularity. Additionally, these ratings help in refining the accuracy of predictive models by serving as a benchmark for evaluating similarities between movies and user rating patterns.
- Overview - This text data allows for content-based filtering in movie recommendation systems, where movies can be suggested based on similarities in storylines, themes, or genres detailed in their overviews. WIth the use of Python’s CountVectorizer, we will be able to transform the text into numerical data and measure how closely movies align.
- Meta_score - It refers to a weighted average score out of 100, derived from critical reviews aggregated from various sources. It provides a consolidated indicator of the critical reception of movies, helping to gauge how well-received a film is among critics, which can contrast with audience ratings like those from IMDB. It can be helpful in the movie recommendation

system, since the recommendation algorithm should also balance viewer preferences with critical opinions, which will enhance the diversity and credibility of the recommendations.

- *Director* - Can be used for understanding impacts of directorial style on movie ratings and preferences. Directors often have a distinct style or thematic focus that influences their films. Analyzing the director's impact can reveal patterns in movie features such as genre, pacing, and cinematography. This consistency helps in clustering movies into meaningful groups that align with specific viewer preferences.
- *Star1, Star2, Star3, Star4* - Names of the lead stars, crucial for recognizing user's actor-based preferences and popularity. In database analysis, lead actors are significant because their popularity and acting credibility can heavily influence a movie's financial success and ratings, making actor-related data a valuable predictor in analytical models.
- *No of votes* - In this database the number of votes is crucial for assessing the reliability and stability of the IMDB ratings. A higher number of votes typically indicates a more universally accepted and robust rating, reducing the likelihood of skewed results from a smaller, less diverse voter base. In a movie recommendation system, this metric is important because it helps prioritize recommendations based on movies that have not only high ratings but also strong viewer engagement, suggesting broader appeal and validation.
- *Gross* - In the context of database analysis, it is a key indicator of its commercial success and can highlight trends in consumer preferences across different genres, regions, or time periods. Analyzing this data helps to understand which types of films are financially successful, providing insight into market dynamics and audience tastes. In a movie recommendation system, incorporating gross earnings can enhance the recommendation logic by prioritizing movies that are not only critically acclaimed but also popular and successful, potentially increasing user satisfaction by suggesting widely enjoyed films.

To observe interesting patterns and receive statistics, we prepared visualizations in Tableau, Excel, and VSCode in Python using libraries like Pandas, NumPy, MatPlotLib and Seaborn. First, we modified the database in Jupiter Notebook, using Pandas library. The database have been modified in the way that the poster_link feature and the 'min' unit from the runtime column have been removed. We decided that the poster_link is irrelevant in the case of the movie recommendation systems.

This IMDB database only consists of top 1000 movies and tv series, and the ranking is based on the ratings. Ratings are in range 7.6 to 9.3. We plotted the movie and tv series names with their IMDB ratings, to demonstrate the distribution of the ratings in this database. The plot is shown in Figure 1.1. With ratings confined to a narrow high range, the system may struggle to differentiate sufficiently between "good" and "great" films, potentially limiting the effectiveness of predictive analytics that rely on a wider variance in ratings to discern preferences more accurately.

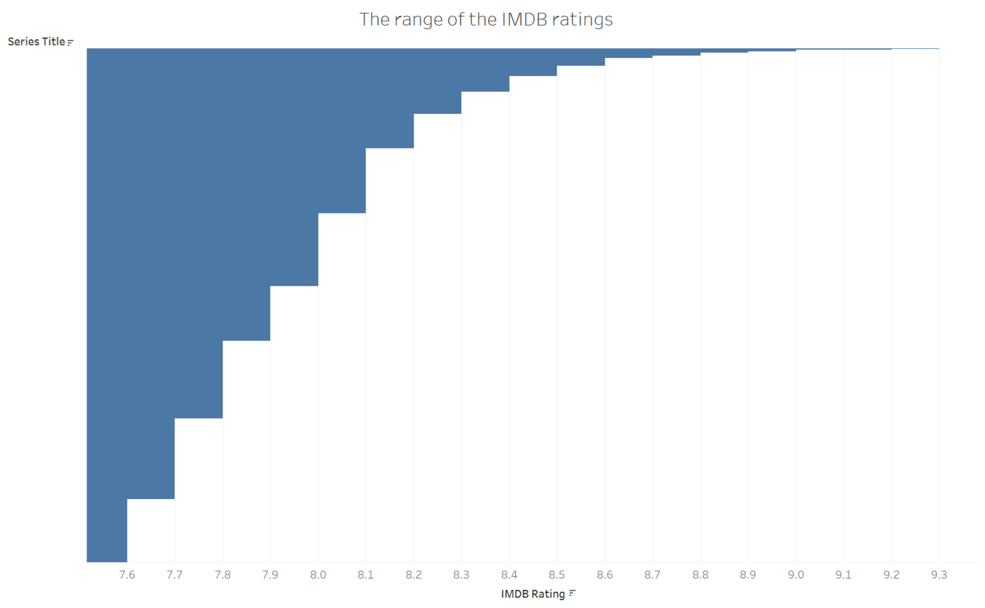


Figure 1.1. - Distribution of the IMDB ratings in the database.

High-rated movies tend to cluster in certain popular or critically acclaimed genres, potentially leaving out less common genres. This could skew the recommendation system towards more mainstream or universally acclaimed genres, reducing diversity in recommendations. Let’s plot a graph showing the distribution of the genres in our database to see if the assumption above is true. The visualization is shown in Figure 1.2. Genres have not been separated since some movies naturally fit into multiple genres, reflecting a blend of themes and styles. Separating these genres might oversimplify or distort the understanding of such movies, as the combination itself can provide unique insights into the content and appeal of the film.

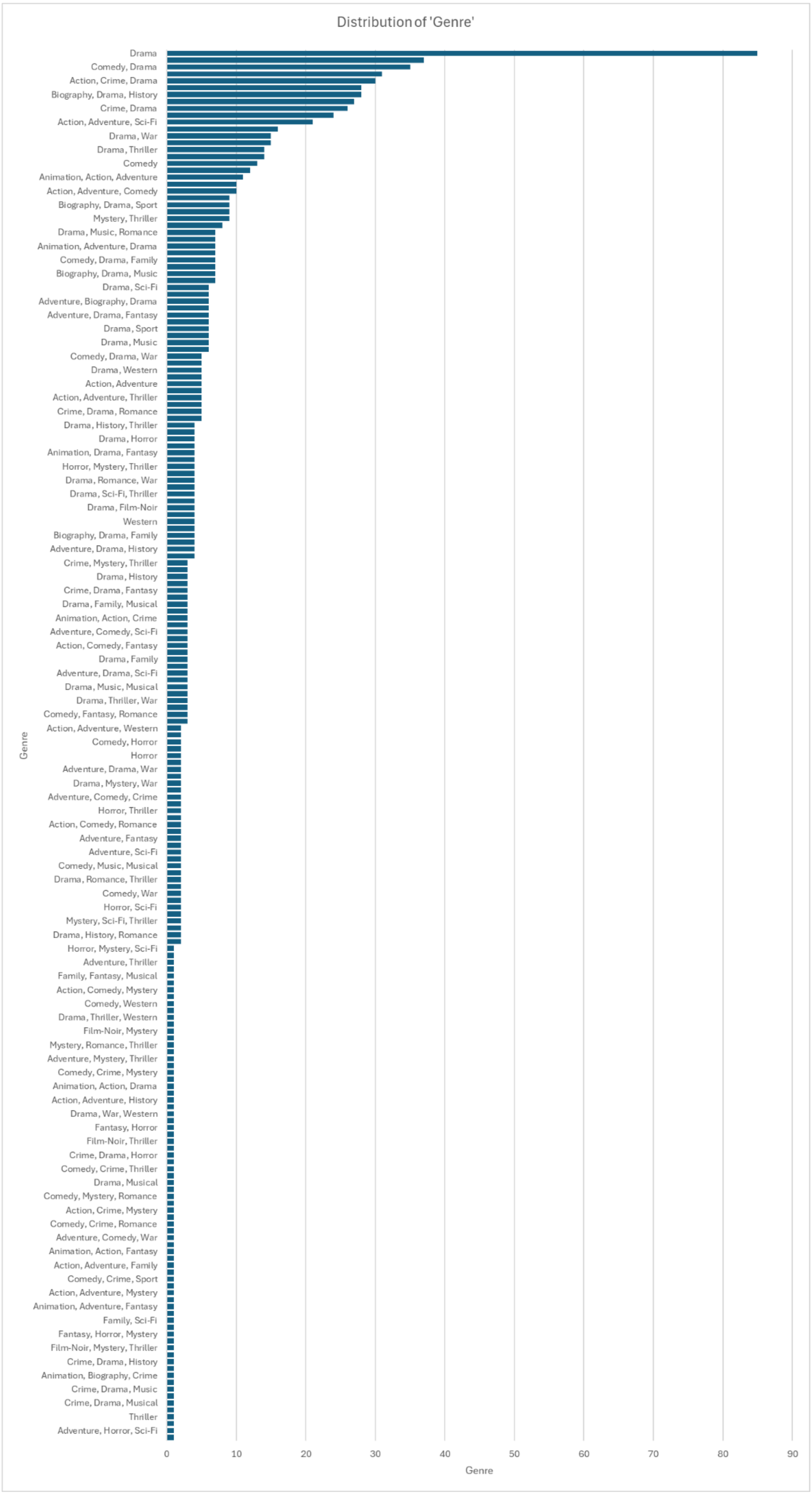


Figure 1.2. - The distribution of the movie genres in the database.

The assumption above has been discovered to be true. The visual suggests that the most frequent genre combinations likely correspond to more mainstream or popular choices, which could potentially skew a recommendation system towards these combinations. Consequently, less common genres or unique genre combinations may be underrepresented. This distribution could indeed reduce diversity in recommendations, as the system might favor these more common genre combinations when generating recommendations, potentially neglecting niche genres or unique cinematic experiences that could appeal to specific audience segments.

The data imbalance is also visible in other features of our database. The dataset is biased towards movies from specific eras that generally receive higher ratings. For instance, classic, newer Hollywood films or contemporary blockbusters dominate, overlooking the older movies. Here, we can see a major imbalance between movies from 1993 to 2018, over the older movies from 1920 to 1992. The visualization of this imbalance can be seen on Figure 1.3.

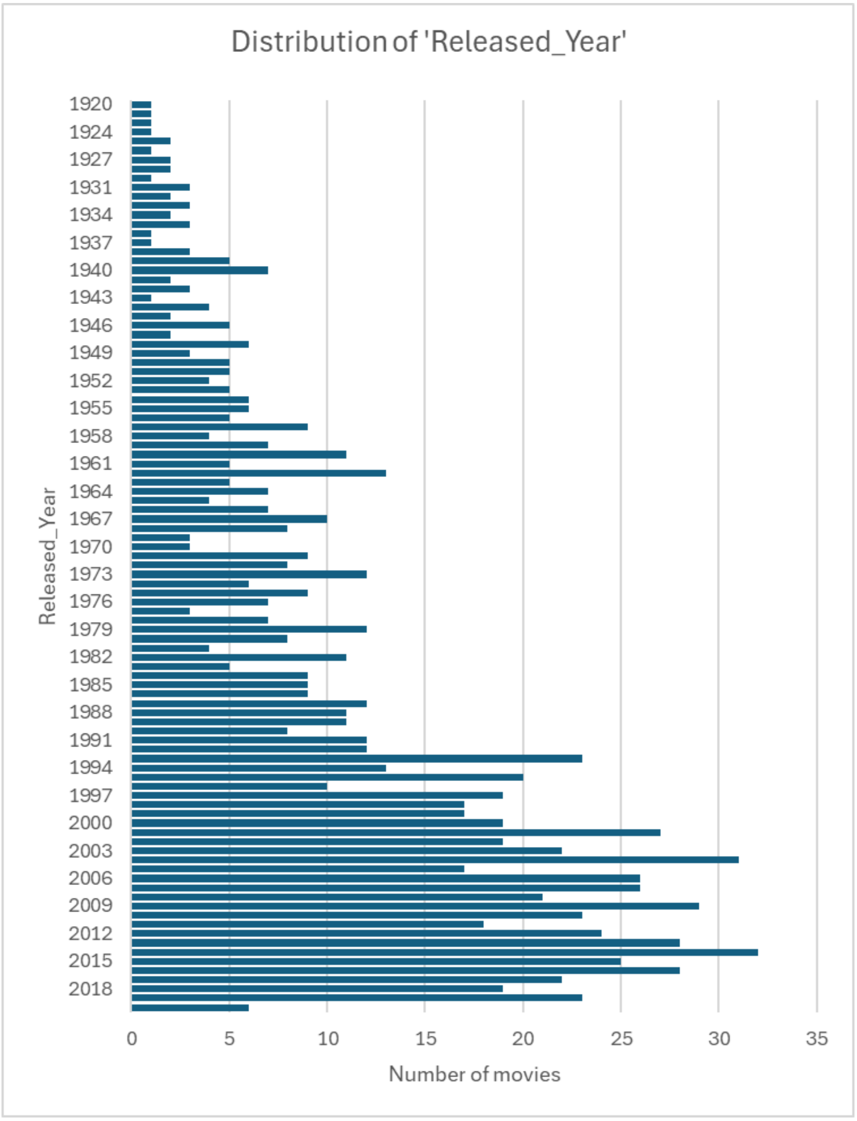


Figure 1.3 - The distribution of the release years in the database.

This data imbalance in the distribution of the movies directed by different people is also visible here. We decided to provide a table (Table 1.3) to show the number of directors associated with the number of movies they directed.

Movies Directed	***Number of Directors***
1	352
2	99
3	43
4	19
5	13
6	7
7	3
8	4
9	3
10	2
11	1
13	1
14	1

Table 1.1. - The distribution of the directors in the database.

A significant number of directors have directed only one film. This represents a considerable proportion of the directors in the dataset, indicating a broad base of less prolific directors. As the number of movies directed increases, the number of directors decreases sharply, showcasing the typical long-tail distribution. A very small group of directors have directed more than 10 films, highlighting that only a few directors are highly prolific.

Additionally, we checked for the number of null fields in the dataset. We discovered, that we have 101 Certificate, 157 Meta_score and 167 Gross data missing. Considering that there is 1000 movies and tv series in the database, these numbers are not that significant, and the features are crucial, so we could not remove them.

The correlation between the IMDB_Rating, Meta_score, No_of_Votes, and Gross has been also studied and pictured in Table 1.2. These coefficients measure the strength and direction of the linear relationship between pairs of variables.

	IMDB_Rating	Meta_score	No_of_Votes	Gross
IMDB_Rating	1.000000	0.268531	0.494979	0.082381
Meta_score	0.268531	1.000000	-0.018507	-0.053659
No_of_Votes	0.494979	-0.018507	1.000000	0.602128
Gross	0.082381	-0.053659	0.602128	1.000000

Table 1.2. - IMDB_Rating, Meta_score, No_of_Votes, and Gross correlation table.

IMDB_Rating and Meta_score (0.268531):
This positive correlation suggests a moderate relationship where higher Meta scores are somewhat associated with higher IMDB ratings. This indicates that generally, movies that are critically well-received tend to also be rated favorably by viewers, but the correlation is not very strong.

IMDB_Rating and No_of_Votes (0.494979):
A stronger positive correlation of roughly 0.49 implies that movies with higher IMDB ratings tend to also have a higher number of votes. This might suggest that more popular or highly-rated movies attract more viewers and, consequently, more ratings.

IMDB_Rating and Gross (0.082381):
A very low positive correlation indicates that there is little to no linear relationship between the gross earnings of a movie and its IMDB rating. This means higher ratings do not necessarily predict higher earnings.

Meta_score and No_of_Votes (-0.018507):
A negligible negative correlation suggests that there is no meaningful linear relationship between the Meta score and the number of votes a movie receives. Critical acclaim (as measured by Meta score) does not seem to significantly influence the number of people rating a movie.

Meta_score and Gross (-0.053659):
A slight negative correlation indicates that higher Meta scores are not associated with higher gross earnings and might even slightly predict lower earnings, but this relationship is very weak and likely not significant.

No_of_Votes and Gross (0.602128):
A substantial positive correlation suggests a strong relationship where movies with higher gross earnings tend to receive more votes. This relationship implies that more commercially successful movies, likely due to larger audiences, also receive more ratings.

Finally, we checked which stars are having the most appearances in the database. The results are being shown in Figure 1.4.

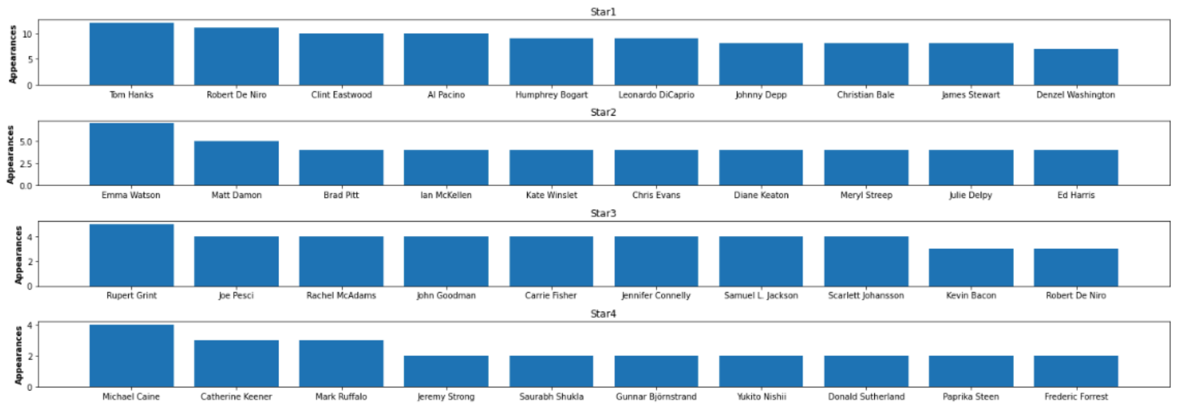


Figure 1.4 - A chart of actors and actresses appearing the most frequently.

The second database that we decided to use collects the ratings of the movies from users. The full original database consists of 26 million ratings from 270,000 users on 45,000 movies. We only used the subset of 100,000 ratings from 700 users on 9,000 movies for our task. These ratings were used for the collaborative filtering parts of our code.

3. Algorithms Description

The algorithm is based on two methods of information filtering systems. We prepared three files in total. To be able to accurately compare these two methods, we created the *content_based.py* and *collaborative.py* files. As the name suggests, the first file includes the recommendation system using the content-based filtering method. It takes the movie's name and suggests a couple of similar ones. The second file is based on the collaborative filtering method. For a particular user, the program suggests a title based on user's ratings of different movies. After that, we decided to create a third file that combined these two methods, and we called it *combined.py*. The last file, combines two methods by taking top 5 movies recommended from each and preparing a list, containing 10 movies.

In the implementation, in every one of the three files, we are using pandas and numpy for the data manipulation, as well as, sklearn, for the text processing and calculating similarities.

3.1. Content_based.py file code and algorithm description

Content-based filtering is a recommendation system approach that leverages the features of items to recommend similar items based on a user's past preferences, rather than relying on the opinions of other users, which is typical of collaborative filtering. This method operates on the principle that if a user liked a particular item in the past, they are likely to appreciate similar items in the future.

The system creates a feature matrix for each movie by aggregating features such as genre, overview, and actors using TF-IDF vectorization. To recommend new items, it calculates a similarity score for each item based on cosine similarity between the combined feature matrices of the items. These scores help rank the items, and the top-ranked ones are then recommended to the user. The similarity between items, which is central to scoring them, is measured using cosine similarity.

First, we are creating an instance of a TfidfVectorizer class. The TfidfVectorizer is a tool used to convert a collection of raw documents into a matrix of TF-IDF features. TF-IDF stands for Term Frequency-Inverse Document Frequency, which is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents. In our case, the words are scored against other words appearing in other records in the database. The TfidVectorizer will ignore common English stop words in the text. In our code, after creating the instance of the class, we are transforming the overview column in the database.

The TF-IDF vectorization in the code inherently handles the normalization of text features. It scales the term frequencies and adjusts them based on their frequency in the document set, resulting in normalized values within the range [0, 1]. This ensures that no single term disproportionately influences the similarity calculations.

In the code, TF-IDF vectorization is applied to the 'Genre', 'Overview', and 'Actors' columns, converting them into matrices of numerical values representing the importance of words in these contexts. The 'Actors' column is created by concatenating the names of the top four stars for each movie. These TF-IDF matrices capture textual and categorical information about the movies, which are then combined to compute similarities between them. This process allows the model to understand and compare movies based on their descriptions and metadata.

After that, we check for the NaN values in the combined matrix. “Not a Number” are special floating-point values used in data processing to represent missing or undefined numerical data. If no NaN values are found, the cosine similarity matrix is computed, which measures the similarity between each pair of movies based on the combined features.

The next part creates a series that maps movie titles to their respective indices in the dataframe. This mapping allows for a quick lookup of movie indices when generating recommendations.

At the end of the implementation, there are functions responsible for generating the recommendation, based on the cosine similarity. The functions first check if the movie exists in the IMDB database. Then they retrieve the index of the given movie title. They compute the similarity scores between the given movie and all other movies using the similarity_matrix. The similarity scores are sorted in descending order to identify the most similar movies. Then, the input movie is excluded from the recommendation list by filtering out its index. The top 10 similar movie titles are returned and printed out as an output.

3.2. Collaborative.py file code and algorithm description

Collaborative filtering recommends items by analyzing user behavior, such as ratings or interactions. It identifies patterns among users and items, suggesting items that similar users have liked or rated highly. There are two main types: user-based, which looks at similar users, and item-based, which looks at similar items.

In this case, we are using an additional database, that were already described in the first section, to be able to base the recommendation on the user's rankings. The code is designed to find users who have rated the input movie highly and then recommend other movies that these users have also liked.

First, we prepare the dataframe by importing both databases, identifing movies that exist in both datasets and filter the dataframes accordingly, and merging the filtered movie with rating data.

We create a structure called the Pivot Table. In the implementation, the method reshapes the dataframe into a matrix where the rows will be indexed by *user_id*, so each row represents a different user, columns will be indexed by *Series_Title*, so each column represents a different movie, and the values in the matrix will be the *ratings* given by users to movies. After that, this matrix is used in the cosine similarity calculation.

Cosine similarity function, in the implementation, computes the cosine similarity between each pair of users. It measures the cosine of the angle between two non-zero vectors in a multi-dimensional space, essentially measuring how similar two user's rating patterns are. The resulting similarity matrix is a square matrix where each entry [*i, j*] represents the cosine similarity score between user *i* and user *j*. The similarity matrix is converted into a pandas dataframe (*user_similarity_df*), with both rows and columns indexed by *user_id*. This makes it easier to access and interpret similarity scores.

After that, the recommendation functions are implemented. They take the movie title, and check if it exists in the main database. If so, the recommendation function finds all users who have rated the given movie title greater than 0. Then, we calculate the average similarity score for each user based on the selected users who have rated the given movie. It first selects the rows in user similarity dataframe corresponding to these movie users, which represents their similarity scores with all other users. Then, it computes the mean similarity score across these rows, resulting in an average similarity score for each user, indicating how similar each user is to the group of users who liked the movie. The scores are sorted.

The program selects 10 similar users. It then, generates movie recommendations by first selecting the rows of the user-item matrix that correspond to the top similar users. After that, it calculates the average rating each movie received from these users, producing a Series where the index is the movie titles and the values are the average ratings. Finally, the movies are sorted in descending order of their average ratings, so the movies with the highest average ratings from similar users are listed first, indicating the most recommended movies.

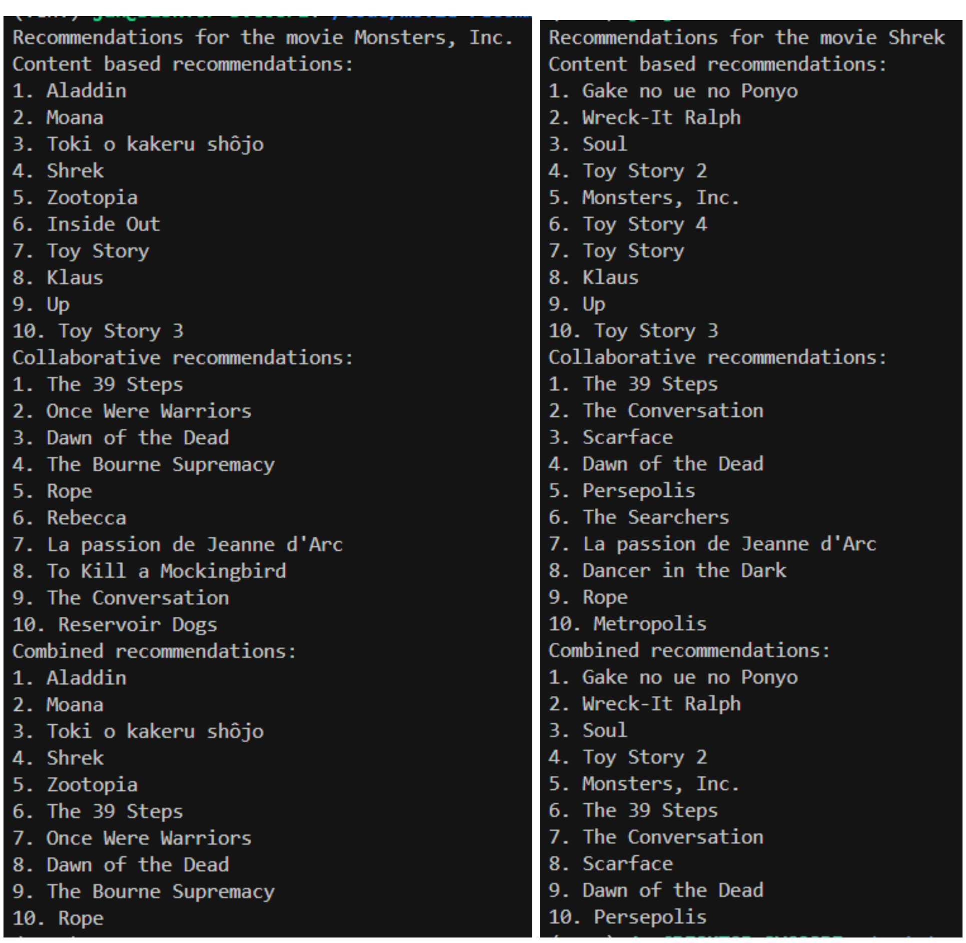
4. Experiments and Conclusions

For the experiments and conclusions, we decided to provide the known movie titles and evaluate the accuracy of the movie recommendation system selection based on our subjective assessment. The outputs of the experiments are presented in Figures 4.1.-4.3. The console outputs are the results of the combined.py program execution.

4.1. Experiments

CASE 1 - Animated Movies or Movies for Children.

First, we decided to test the recommendation system, inputting “Monsters Inc.” as well as the “Shrek” movie.



```
Recommendations for the movie Monsters, Inc.
Content based recommendations:
1. Aladdin
2. Moana
3. Toki o kakeru shôjo
4. Shrek
5. Zootopia
6. Inside Out
7. Toy Story
8. Klaus
9. Up
10. Toy Story 3
Collaborative recommendations:
1. The 39 Steps
2. Once Were Warriors
3. Dawn of the Dead
4. The Bourne Supremacy
5. Rope
6. Rebecca
7. La passion de Jeanne d'Arc
8. To Kill a Mockingbird
9. The Conversation
10. Reservoir Dogs
Combined recommendations:
1. Aladdin
2. Moana
3. Toki o kakeru shôjo
4. Shrek
5. Zootopia
6. The 39 Steps
7. Once Were Warriors
8. Dawn of the Dead
9. The Bourne Supremacy
10. Rope
```

```
Recommendations for the movie Shrek
Content based recommendations:
1. Gake no ue no Ponyo
2. Wreck-It Ralph
3. Soul
4. Toy Story 2
5. Monsters, Inc.
6. Toy Story 4
7. Toy Story
8. Klaus
9. Up
10. Toy Story 3
Collaborative recommendations:
1. The 39 Steps
2. The Conversation
3. Scarface
4. Dawn of the Dead
5. Persepolis
6. The Searchers
7. La passion de Jeanne d'Arc
8. Dancer in the Dark
9. Rope
10. Metropolis
Combined recommendations:
1. Gake no ue no Ponyo
2. Wreck-It Ralph
3. Soul
4. Toy Story 2
5. Monsters, Inc.
6. The 39 Steps
7. The Conversation
8. Scarface
9. Dawn of the Dead
10. Persepolis
```

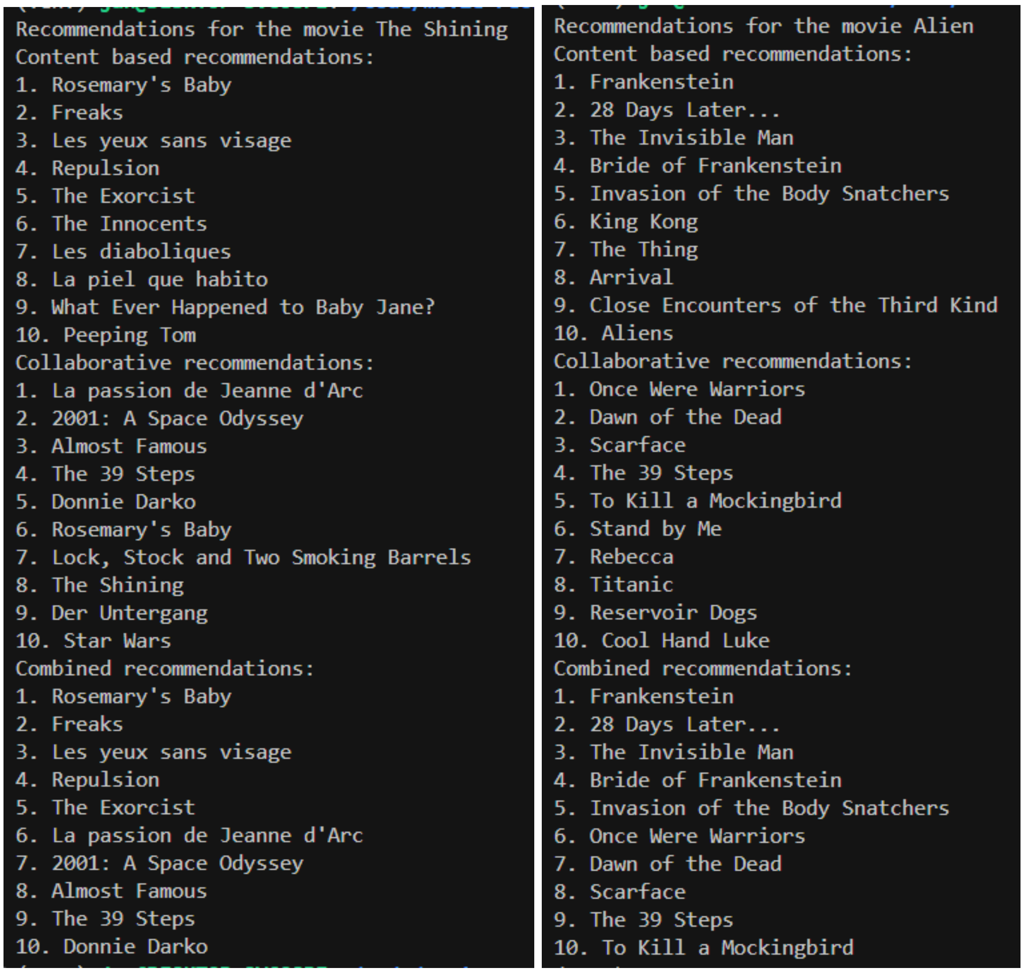
Figure 4.1. - Recommendations for the movies “Monsters, Inc.” and “Shrek”.

The content-based recommendations for "Monsters, Inc." focus on animated and family-friendly movies, often from Disney or Pixar, such as "Aladdin," "Moana," "Shrek," "Toy Story," and "Up." This suggests that the TF-IDF vectorization effectively captures genres and themes similar to "Monsters, Inc." Recommendations like "Gake no ue no Ponyo," "Wreck-It Ralph," "Toy Story," and "Klaus" indicate a strong focus on animated and family genres, aligning with "Shrek's" comedic and fantasy elements.

The collaborative filtering recommendations include a mix of classic and diverse genres, such as "The 39 Steps," "Once Were Warriors," and "Dawn of the Dead." This variety suggests that users who liked "Monsters, Inc." also have diverse tastes, including suspense and drama. For “Shrek”, these recommendations again show a broader range, including classic and critically acclaimed films such as "La passion de Jeanne d’Arc," "The Conversation," and "Scarface." This suggests that fans of "Shrek" may also appreciate a variety of film genres, based on the ratings of different users that also liked “Shrek”.

CASE 2 - Horror and Thriller Movies.

The next inputted movies are “The Shining” and “Alien”.



```
Recommendations for the movie The Shining
Content based recommendations:
1. Rosemary's Baby
2. Freaks
3. Les yeux sans visage
4. Repulsion
5. The Exorcist
6. The Innocents
7. Les diaboliques
8. La piel que habito
9. What Ever Happened to Baby Jane?
10. Peeping Tom
Collaborative recommendations:
1. La passion de Jeanne d'Arc
2. 2001: A Space Odyssey
3. Almost Famous
4. The 39 Steps
5. Donnie Darko
6. Rosemary's Baby
7. Lock, Stock and Two Smoking Barrels
8. The Shining
9. Der Untergang
10. Star Wars
Combined recommendations:
1. Rosemary's Baby
2. Freaks
3. Les yeux sans visage
4. Repulsion
5. The Exorcist
6. La passion de Jeanne d'Arc
7. 2001: A Space Odyssey
8. Almost Famous
9. The 39 Steps
10. Donnie Darko
```

```
Recommendations for the movie Alien
Content based recommendations:
1. Frankenstein
2. 28 Days Later...
3. The Invisible Man
4. Bride of Frankenstein
5. Invasion of the Body Snatchers
6. King Kong
7. The Thing
8. Arrival
9. Close Encounters of the Third Kind
10. Aliens
Collaborative recommendations:
1. Once Were Warriors
2. Dawn of the Dead
3. Scarface
4. The 39 Steps
5. To Kill a Mockingbird
6. Stand by Me
7. Rebecca
8. Titanic
9. Reservoir Dogs
10. Cool Hand Luke
Combined recommendations:
1. Frankenstein
2. 28 Days Later...
3. The Invisible Man
4. Bride of Frankenstein
5. Invasion of the Body Snatchers
6. Once Were Warriors
7. Dawn of the Dead
8. Scarface
9. The 39 Steps
10. To Kill a Mockingbird
```

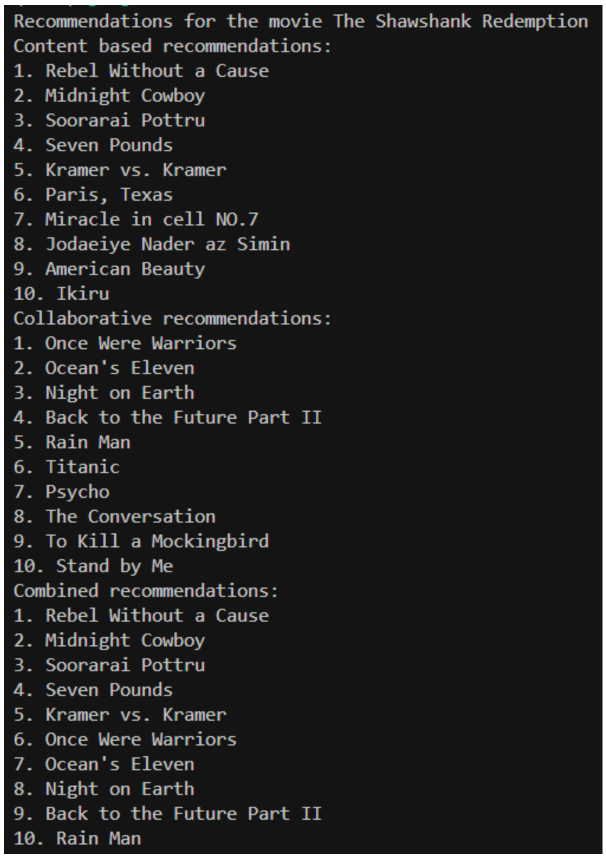
Figure 4.2. - Recommendations for the movies “The Shining” and “Alien”.

Movies recommended based on content similarity for “The Shining”, include psychological horror and thriller classics like "Rosemary's Baby," "Repulsion," "The Exorcist," and "What Ever Happened to Baby Jane?". This suggests that the content-based filtering is effectively identifying films with similar themes and atmospheric qualities. The other recommendations focus on classic horror and sci-fi films such as "Frankenstein," "28 Days Later...," "The Invisible Man," and "Invasion of the Body Snatchers." This indicates a strong alignment with the horror and science fiction elements present in "Alien."

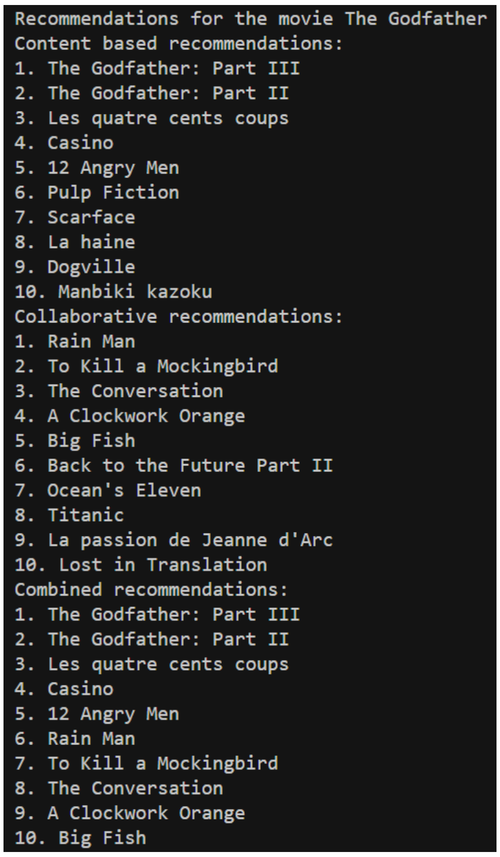
The collaborative filtering recommendations include a mix of psychological thrillers and other genres like "2001: A Space Odyssey," "Almost Famous," and "Lock, Stock and Two Smoking Barrels." This variety indicates that users who liked "The Shining" also have diverse tastes, appreciating both psychological depth and broader cinematic experiences. Collaborative filtering recommendations for "Alien" are diverse, including titles like "Once Were Warriors," "Scarface," "To Kill a Mockingbird," and "The 39 Steps." This suggests that users who enjoyed "Alien" have wide-ranging tastes spanning genres beyond horror and sci-fi.

CASE 3 - Movies Rated the Highest.

The inputted movies are “The Shawshank Redemption” and “The Godfather,” which are also rated the highest in the database.



```
Recommendations for the movie The Shawshank Redemption
Content based recommendations:
1. Rebel Without a Cause
2. Midnight Cowboy
3. Soorarai Pottru
4. Seven Pounds
5. Kramer vs. Kramer
6. Paris, Texas
7. Miracle in cell NO.7
8. Jodaeiye Nader az Simin
9. American Beauty
10. Ikiru
Collaborative recommendations:
1. Once Were Warriors
2. Ocean's Eleven
3. Night on Earth
4. Back to the Future Part II
5. Rain Man
6. Titanic
7. Psycho
8. The Conversation
9. To Kill a Mockingbird
10. Stand by Me
Combined recommendations:
1. Rebel Without a Cause
2. Midnight Cowboy
3. Soorarai Pottru
4. Seven Pounds
5. Kramer vs. Kramer
6. Once Were Warriors
7. Ocean's Eleven
8. Night on Earth
9. Back to the Future Part II
10. Rain Man
```



```
Recommendations for the movie The Godfather
Content based recommendations:
1. The Godfather: Part III
2. The Godfather: Part II
3. Les quatre cents coups
4. Casino
5. 12 Angry Men
6. Pulp Fiction
7. Scarface
8. La haine
9. Dogville
10. Manbiki kazoku
Collaborative recommendations:
1. Rain Man
2. To Kill a Mockingbird
3. The Conversation
4. A Clockwork Orange
5. Big Fish
6. Back to the Future Part II
7. Ocean's Eleven
8. Titanic
9. La passion de Jeanne d'Arc
10. Lost in Translation
Combined recommendations:
1. The Godfather: Part III
2. The Godfather: Part II
3. Les quatre cents coups
4. Casino
5. 12 Angry Men
6. Rain Man
7. To Kill a Mockingbird
8. The Conversation
9. A Clockwork Orange
10. Big Fish
```

Figure 4.3. - Recommendations for the movies “The Shawshank Redemption” and “The Godfather”

The content-based recommendations for “The Shawshank Redemption”, these recommendations include a mix of classic dramas and films with deep emotional and moral themes such as "Rebel Without a Cause," "Midnight Cowboy," and "Kramer vs. Kramer." This indicates that content-based filtering effectively identifies movies with similar thematic elements and narrative styles. In the case of “The Godfather”, content-based recommendations focus on classic and critically acclaimed films, particularly those with themes of crime, justice, and moral complexity, such as "The Godfather: Part II," "12 Angry Men," and "Scarface." This highlights the effectiveness of content-based filtering in identifying films with similar narrative and thematic depth. We can see that the recommendation system also suggested the user watch sequels of the inputted movie.

Collaborative filtering recommendations for “The Shawshank Redemption” include a variety of genres, including drama, action, and classics such as "Once Were Warriors," "Ocean's Eleven," and "Psycho." This suggests that users who liked "The Shawshank Redemption" also appreciate many films. The collaborative filtering recommendations for “The Godfather” include a mix of classic and highly rated films across various genres, including "Rain Man," "To Kill a Mockingbird," and "A Clockwork Orange." This suggests that fans of "The Godfather" have diverse tastes and appreciate a range of genres beyond crime dramas.

4.2. Conclusions

Combining content-based and collaborative filtering methods results in a balanced and comprehensive recommendation system. Content-based filtering ensures relevance and thematic similarity by suggesting movies with similar themes, genres, and styles, making it effective for new items and solving the cold start problem. However, it can lead to over-specialization. Collaborative filtering, on the other hand, captures diverse user preferences and provides personalized recommendations based on similar users' tastes, but it struggles with new items due to the cold start problem and depends heavily on user interaction data. The hybrid approach leverages the strengths of both methods, offering a richer and more satisfying recommendation experience by providing both content-relevant suggestions and user-driven diversity. Since we decided to do it in a fifty-fifty way by splitting the recommendations in half, we got both negative and positive aspects of these two methods reflected in the outputs of the programs.

For example, combined recommendations for "Monsters, Inc." and "Shrek" included both family-friendly animated films and a variety of other genres, while "The Shining" and "Alien" recommendations merged horror and thriller films with diverse user preferences, ensuring a comprehensive list.