Preliminary Report

Project 19
Movie Recommendation System

By Jan Szachno and Aleksandra Głogowska

---

1. Task Description

The task is to develop a recommendation system for movies 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.

During work on the project, we are going to use content-based filtering - since our dataset contains a lot of information about each of the movies, but does not contain information about preferences of each particular user.

2. Dataset Description

The 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.

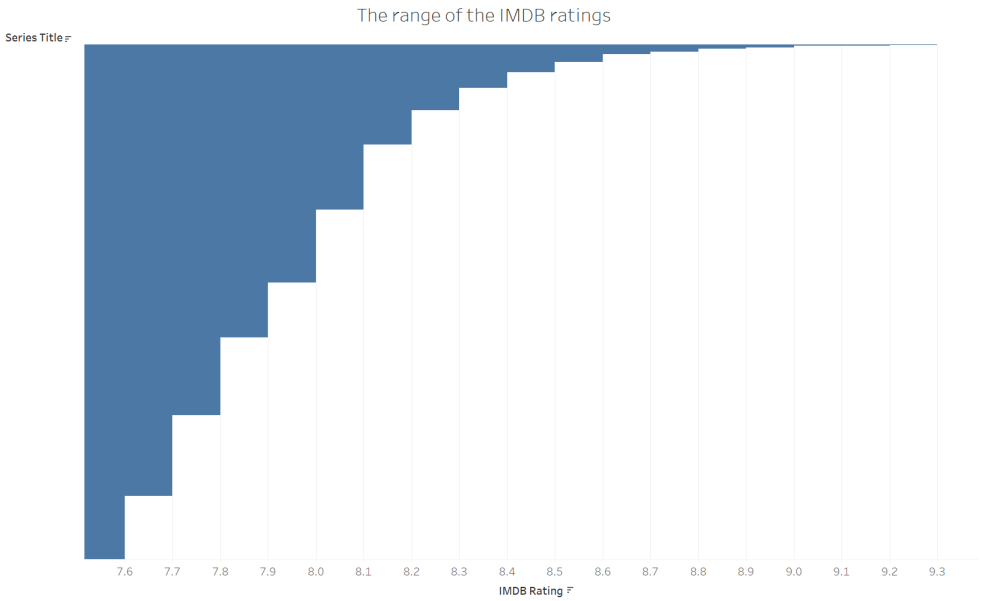

Figure 1.1. - Distribution of the IMDB ratings in the database.

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.

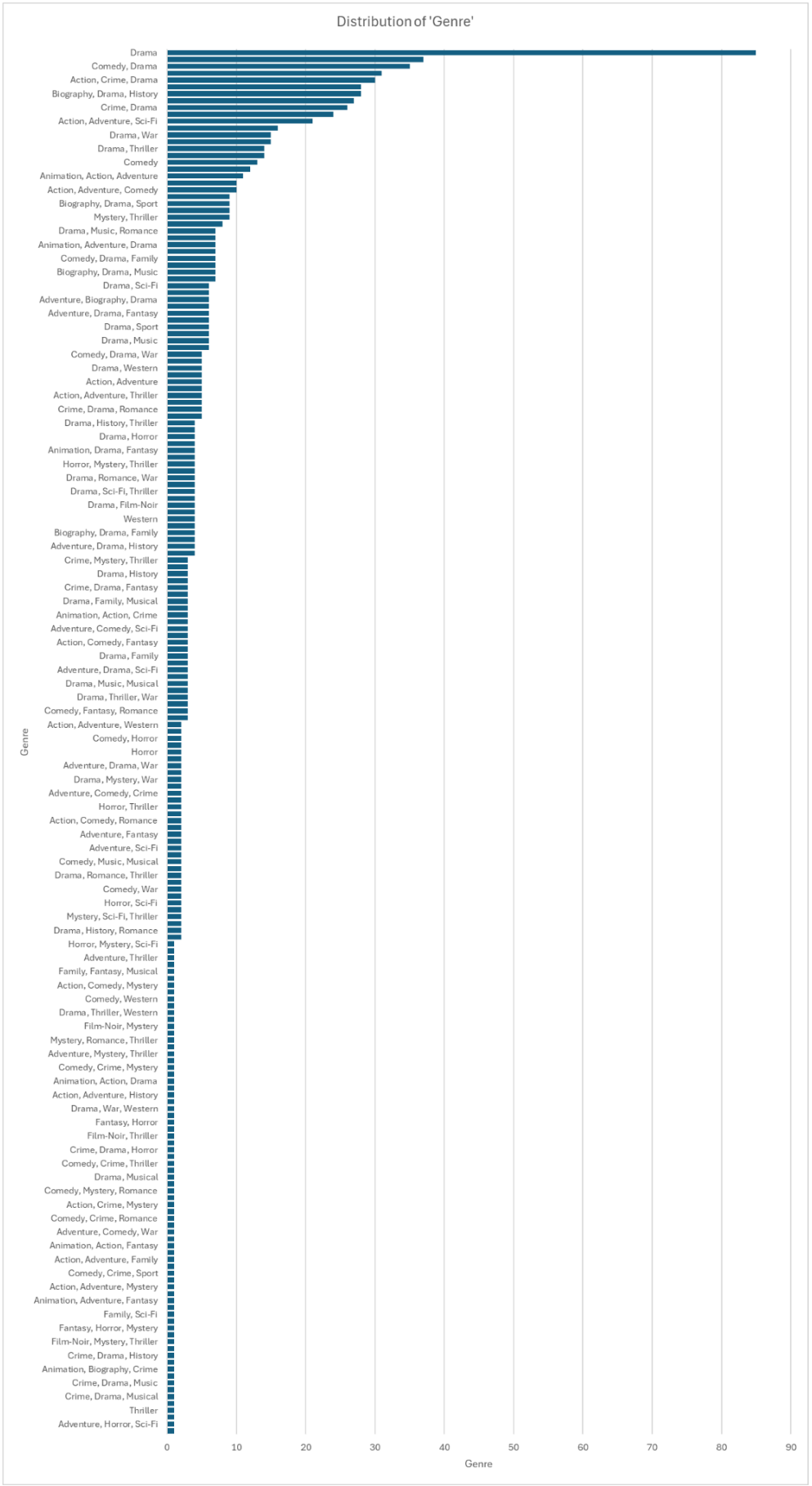

Figure 1.2. - The distribution of the movie genres in the database.

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.

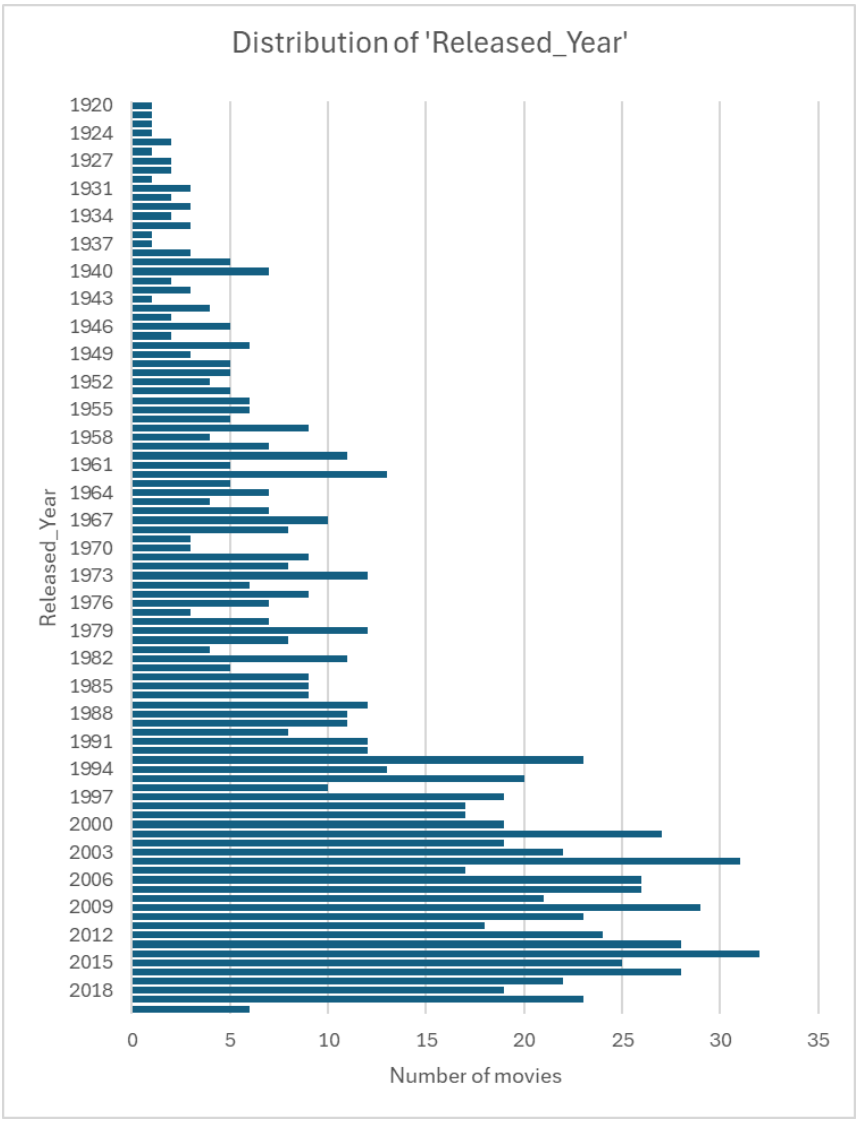

Figure 1.3 - The distribution of the release years in the database.

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.

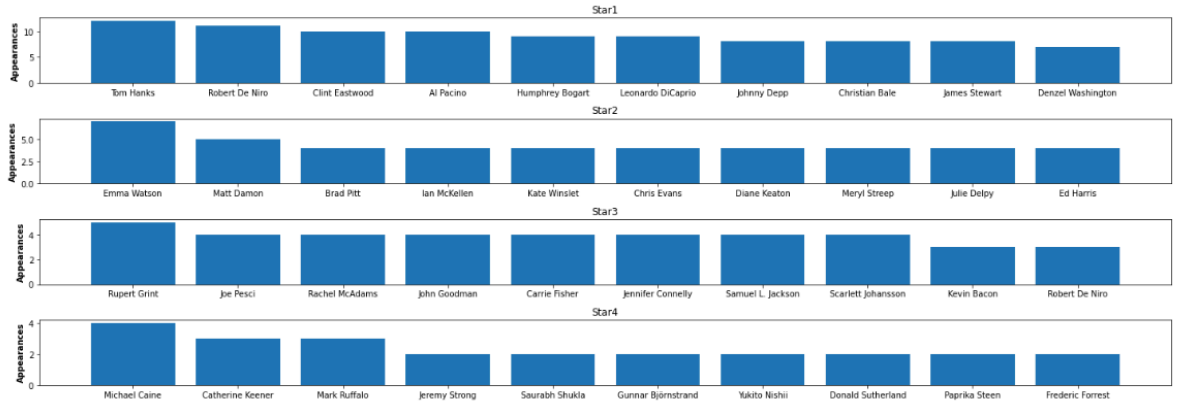

Figure 1.4 - A chart of actors and actresses appearing the most frequently.

3. Algorithms Description

To manage the data imbalance, we plan to employ a technique known as class weighting. **Class weighting** is a technique used in machine learning to adjust the importance of each class in a dataset during training, which is particularly useful in scenarios involving imbalanced datasets where some classes are underrepresented. The main goal of class weighting is to encourage the model to pay more attention to samples from these underrepresented classes, potentially improving model performance and achieving a more balanced and generalizable model. Class weights can be configured during the model's training setup. Rarer classes are assigned higher weights, thereby increasing the penalty for misclassifying them during training. Adjusting class weights affects the training by modifying the loss function. A higher weight on a class means that errors on samples from that class have a more significant impact on the total loss, compelling the model to optimize predictions for that class more effectively. However, finding the optimal class weights can be challenging. Overweighting a minority class might lead the model to overfit these classes at the cost of overall accuracy. Determining the most effective weights requires careful experimentation and validation to find the right balance.

When it comes to splitting the dataset into training and validation sets, we will need to experiment with different methods since the dataset is not balanced. When using random split it may happen that a certain genre is not even present in the training dataset. We plan to experiment with a stratified split provided by the sklearn library, to maintain the distribution of important categorical variables in both the training and validation sets. We consider to use the following fields in the process:

- Genre - to ensure variety of genres in both sets
- Released year - to ensure variety of periods

For performance evaluation, we are going to use cosine similarity of movies. Also, to take the imbalance and relatively small size of the dataset into account during the process of assessing the performance, we will use k-fold cross validation. **K-fold cross-validation** is a statistical method used to evaluate the performance of predictive models, ensuring they generalize well to an independent dataset. It is particularly useful in machine learning to determine the robustness of a model across unseen data. The entire dataset is divided into k distinct subsets, or "folds", of approximately equal size. The model undergoes training and testing k times, cycling through each fold such that every fold serves as the test set exactly

once, and the remaining k−1 folds form the training set. This cycle allows the model to train on various portions of the dataset and validate against the remaining parts, thoroughly examining its performance. During each iteration, the model is trained on the combined k−1 folds and tested on the one not used for training. Performance metrics such as accuracy, precision, recall, and F1-score for classification tasks, or mean squared error and mean absolute error for regression tasks, are calculated. After cycling through all kk folds, the individual performance metrics are averaged to yield a single performance estimation.

This technique uses all data points for training and validation, leading to efficient data utilization and a more accurate estimate of the model's performance. Despite its benefits, k-fold cross-validation can be sensitive to how the data is divided; if the dataset is imbalanced or the splitting isn't properly randomized, the results might not accurately reflect the model's ability to generalize.

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 process begins with item representation, where each item in the dataset is characterized by a set of features. For instance, in a movie recommendation system, these features might include the genres, director, scriptwriter, and major actors of a film.

Following this, the system creates a user profile for each individual by aggregating the features of items they have shown a preference for, either through high ratings or frequent interactions. To recommend new items, the system calculates a score for each item based on its similarity to the user's profile. 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 often measured using cosine similarity, Euclidean distance, or Pearson correlation, depending on the type of feature data.

Furthermore, the system uses feedback from the user's interactions with the recommended items to update and refine their profiles, which enhances the personalization of subsequent recommendations.

While content-based filtering can be highly effective, especially in scenarios like the cold start problem or when new items are frequently added, it can also lead to over-specialization. This happens when recommendations become too similar, failing to introduce users to new categories or genres, thus limiting the discovery of diverse content.

4. Tools, Framework and Libraries Used

In the project to develop a movie recommendation system based on the IMDB dataset there are used, several Python libraries including Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn - utilized for data manipulation, chart creation, analysis, and machine learning tasks. Visualization tools like Tableau and Excel are employed to explore the data and understand distributions.