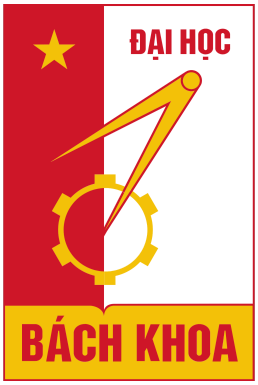
Hanoi University of Science and Technology
Semester: 2024.1
Course: Data Science
Supervisor: Assoc. Prof. Than Quang Khoat
Dr. Tran Viet Trung
Dr. Bui Thi Mai Anh
Dr. Nguyen Thi Oanh



MOVIE RECOMMENDATION SYSTEM

December, 2024

Group 10

Group 10:	VU TRUNG THANH	20220066
	LUONG HUU THANH	20225458
	DOAN ANH VU	20225465
	NGUYEN MAU TRUNG	20225534

Contents

1 Introduction

1.1 Motivation

In today's fast-paced digital age, the film industry faces a significant challenge: how to connect audiences with the right content in the middle of an ever-expanding library of options. With the overwhelming volume of movies available across theaters and online streaming platforms, viewers often find it difficult to choose what to watch. As a result, the demand for intelligent, personalized recommendations has grown exponentially. Addressing this need, a movie recommendation system has become an essential tool for helping audiences quickly discover relevant and high-quality films tailored to their preferences.

Additionally, as consumer viewing habits shift toward digital and streaming, recommendation systems play a crucial role in keeping viewers engaged, broadening their viewing options, and ultimately improving their satisfaction and retention on these platforms. In a highly competitive market, a strong recommendation system is essential for attracting and retaining audiences while enhancing the overall user experience.

We want to develop a recommendation model that provides personalized movie suggestions based on users' past preferences and behaviors. This model should accurately predict and recommend movies each user is likely to enjoy, helping them navigate vast content libraries effectively.

1.2 Problem formulation

The recommendation problem addressed in our project is based on predicting numerical ratings that users assign to movies, which can be formulated as a learning problem about user-item interactions:

Let M and N denote the number of users and items, respectively. The user–item interaction is represented by a numerical rating matrix $R \in \mathbb{R}^{M \times N}$, where each entry r_{ui} indicates the explicit rating provided by user u and item i:

$$r_{ui} = \begin{cases} r, & \text{if user } u \text{ has rated item } i \text{ with score } r; \\ 0, & \text{if no rating exists.} \end{cases}$$

Here, r represents a numerical score, typically on a predefined scale (e.g., 1–10), reflecting the user's preference for the item. A value of zero indicates the absence of a rating. The goal is to predict the missing entries in R — that is, to estimate $\hat{r}_{ui}$, the predicted rating for user u and item i, given the observed ratings. This is a supervised learning problem, where the observed ratings are treated

as the training data.

Formally, the task involves learning a function $\hat{r}_{ui} = f(u, i|\theta)$, where f is the function that maps user u and item i, along with model parameters θ to predicted rating $\hat{r}_{ui}$. Learning this function often involves minimizing the square loss between predicted value $\hat{r}_{ui}$ and target value r_{ui}.

2 Data Collection

For the process of gathering data for our project, we mainly use two powerful web scraping tools: **Selenium** and **BeautifulSoup**.

Selenium, a versatile tool for browser automation, was used for dynamically interacting with webpages, ensuring access to user rating lists that required scrolling or navigating through dynamic elements. On the other hand, **BeautifulSoup**, a Python library for parsing HTML and XML documents, enabled efficient extraction of structured data from static web pages. Together, these tools formed the backbone of our scraping pipeline, combining automation and precision for effective data collection.

For our data sources, there are many choices such as Rotten Tomatoes, Metacritic, Fandango, etc. We decided to choose **IMDb** for multiple reasons, especially considering our goal of building a robust movie recommendation system.

- IMDb has a large and comprehensive database of movies, TV shows, and related information, including metadata such as cast and crew, genres, plot summaries, release dates, production companies, and more.
- IMDb provides a vast amount of user-generated content, most notably user ratings and reviews. This abundance of user data enable us to identify users with similar tastes and recommend movies that similar users have enjoyed. The volume of ratings on IMDb allows for more statistically significant analysis and more accurate recommendations.
- The structure of IMDb's data is relatively consistent and well-organized. This consistency makes data extraction and processing more manageable, which is essential for a large-scale project like building a recommendation system. While some platforms rely heavily on APIs or have more complex data structures, IMDb's web pages offer a stable base for data collection using web scraping techniques.

The collection of user rating data involved a two-step process. First, a list of popular films was obtained from IMDb, and Selenium was used to gather a pool of users who had rated those films.

Then for each user, their rating history was scraped, with Selenium handling the dynamic nature of IMDb's user rating pages. Scrolling scripts were used to load all content dynamically, and exceptions were handled to prevent data loss.

Gathering film metadata followed a simpler static scraping approach. Film metadata was extracted directly from IMDb's movie pages. BeautifulSoup parsed the HTML content, extracting metadata from structured content in the page. Unlike user rating data, these static film pages allowed for straightforward HTTP requests, saving time on crawling.

Rate limiting was carefully implemented throughout the process to avoid overloading the servers and ensure compliance with ethical data scraping practices. By introducing delays between requests and handling server responses, the data collection process balanced efficiency with responsibility. These techniques ensured the acquisition of a rich, high-quality dataset, forming the foundation for building the recommendation system.

After collecting the data, it was stored in CSV files for efficient management and further analysis. The user-rating data, consisting of user IDs, film IDs, and their corresponding ratings, was saved in a structured format for better accessibility. Similarly, the metadata for films, including attributes such as the film title, description, genre, content rating, duration, and aggregated ratings, was stored in separate CSV files. Organizing the data in this way ensured compatibility with standard data science tools and libraries. This structured format helps subsequent data preparation and exploratory data analysis (EDA), enabling efficient loading, filtering, and analysis of data.

3 Data Preparation & Exploratory Data Analysis (EDA)

3.1 Preprocessing

The film dataset under observation consists of 13 columns, each providing essential information about films. The *fid* column serves as a unique identifier for each film. The *name* column contains the title of the film, while *description* provides a brief summary of its content. The *ratingCount* and *ratingValue* columns represent the number of ratings and the average rating score, respectively. The *contentRating* column specifies the age-appropriateness of the film. The *genre* and *keywords* columns categorize the film based on its theme and associated keywords. The *duration* column records the runtime of the film, and *datePublished* indicates its release date. The *actor* and *director* columns list the main cast and the director of the film. Finally, the *image* column provides a link to the film's poster or thumbnail. These attributes collectively offer a comprehensive overview of the dataset.

The user-rating data has three columns, which are *uid* for user id, *fid* for film id and *rating* show the rating that user have rated.

The raw scraped film data contained several inconsistencies, including missing values, redundant columns, and varying notations for the same meaning. To optimize and prepare the dataset for further analysis, we applied several preprocessing techniques to address these issues.

Firstly, for the *contentRating* column, we observed that films were tagged with different notations representing similar classifications. For example, films suitable for all audiences had tags such as “G" and “Approved". To standardize these notations, we grouped tags with similar meanings into a unified category. In this case, the aforementioned tags were grouped under the label *General*. This process was repeated for all classification types, resulting in eight distinct categories for content ratings that will be used in later stages.

```
'PG':['P','PG','TV-MA','TV-PG','M/PG'],
'PG-16':['C16','16+','T16', 'T'],
'PG-13':['C13','13+','PG-13','T13','TV-14'],
'General':['G','Approved','TV-G','Passed','GP'],
'Banned':['(Banned)'],
'Adults':['NC-17','T18','M','X','MA-17','R','C18'],
'Not Rated':['Not Rated', 'Unrated'],
'Children':['K','TV-Y7-FV','TV-Y7', 'TV-Y']
```

Figure 1: Group of content ratings.

For numerical columns, such as *duration* and *date published*, missing values were replaced with the mean of the respective columns. This approach ensures that null values do not hinder further analysis while preserving the integrity of the dataset.

Additionally, we noted that while every film in the dataset had at least one associated genre, the *keywords* column occasionally contained missing values. To address this, we randomly selected keywords from the corresponding film genres to fill in these missing entries.

Finally, for the *actor* and *director* columns, rows with missing values were imputed with the placeholder label “unknown". This ensures that no row is excluded from analysis due to incomplete information.

Through these preprocessing steps, the dataset was cleaned, standardized, and prepared for subsequent exploratory data analysis and model development.

3.2 Exploratory Data Analysis

3.2.1 Film data

In this subsection, we explore the key characteristics of the film dataset, including distributions, patterns, and potential anomalies. This analysis helps us understand the overall structure of the data, such as the *duration*, *contentRating*, and *genres*. The insights gained will guide the development of the recommendation system.

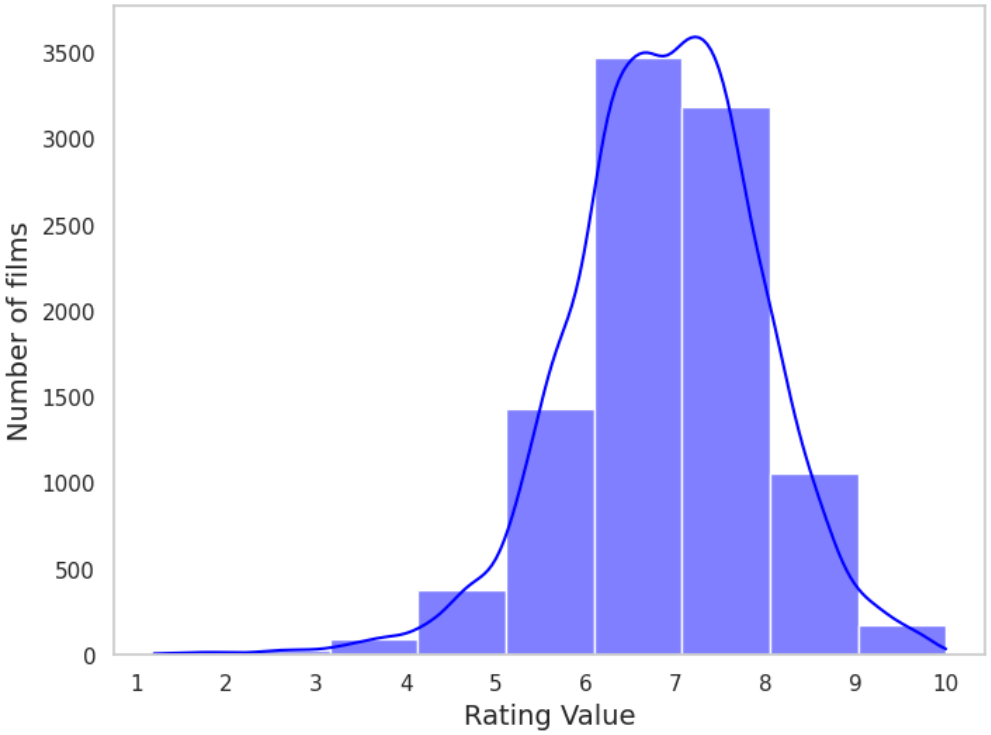


Figure 2: Rating Values Distribution.

The distribution of film ratings, shown in Figure 2, reveals that the majority of ratings are concentrated between 6 and 8, with a peak around 7. This indicates that most films receive moderate to above-average scores. The distribution exhibits a slight left skew, as ratings below 5 are less frequent, while those above 8 decline steadily. Exceptionally high ratings near 10 are rare, suggesting fewer films achieve this distinction. Overall, the distribution resembles a near-normal curve, though slightly asymmetrical, with a sharper drop-off on the higher end. This clustering around average values is a crucial observation for optimizing the recommendation system.

Figure 3 illustrates the distribution of the top 10 most frequent movie *genres* in the dataset. *Drama* emerges as the dominant genre, with over 5,000 films, far surpassing the other categories. *Comedy* follows with more than 3,000 entries, while *Action* ranks third with approximately 2,600 movies. The remaining genres, such as *Crime*, *Adventure*, and *Thriller*, show a gradual decline in rep-

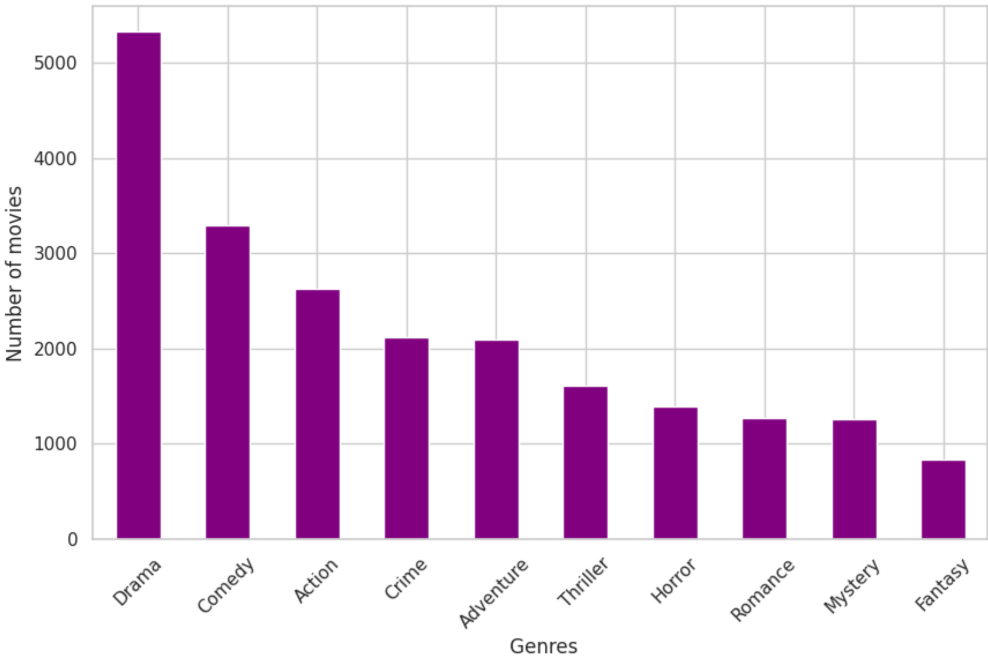


Figure 3: Top 10 common genres.

resentation. Notably, *Fantasy* has the fewest entries among the top 10. This distribution underscores the prominence of *Drama* and *Comedy*, reflecting the significant emphasis on these genres in film production.

By integrating insights from the *genres* column (Figure 3) and the most frequent *keywords* (Figure 4), we uncover valuable descriptive features for the recommendation system. While the *genres* column captures broader categories, the *keywords* provide specific thematic elements that enrich the dataset. Together, these two attributes enable the extraction of meaningful features, balancing categorical classifications with detailed descriptors of film content.

Figure 4: Most common keywords.

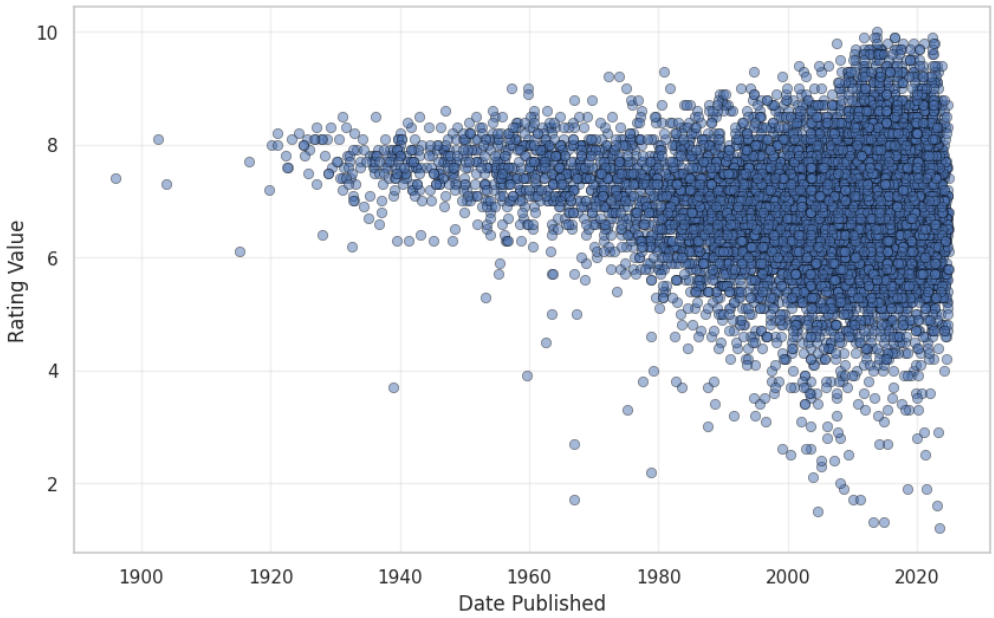


Figure 5: Scatter Plot of Rating Value vs Date Published.

Scatter plot depicting the relationship between *Rating Value* and *Date Published* are presented in Figure 5. A clear trend is observed, where older films (pre-1960) tend to have higher ratings, clustering around values of 7 to 8. In more recent decades, the spread of ratings increases considerably, with both extremely high and low scores becoming more common. This wider variation suggests changes in audience preferences and rating behaviors, likely influenced by the substantial growth in film production. Additionally, the rightward skew in the *Date Published* axis indicates that the dataset contains a higher volume of films from the late 20th century onward. This temporal trend offers critical insights into the evolution of film ratings and helps contextualize historical and contemporary patterns for the recommendation system.

Figure 6 displays the correlation matrix for the numerical columns: *ratingValue*, *ratingCount*, *duration*, and *yearPublished*. The diagonal values are all 1.00, representing the perfect correlation of each column with itself. The following insights can be derived from the matrix:

- ***ratingValue* and *ratingCount*:** There is a moderate positive correlation of 0.24, suggesting that films with higher ratings tend to attract more votes. However, this correlation is not particularly strong.

- ***duration* and *ratingCount*:** A weak positive correlation of 0.25 is observed, indicating that longer films are slightly more likely to receive more votes, though the effect is minimal.

- ***duration* and *ratingValue*:** The correlation between *duration* and *ratingValue* is 0.02, showing a negligible relationship, meaning that film length does not significantly affect the average

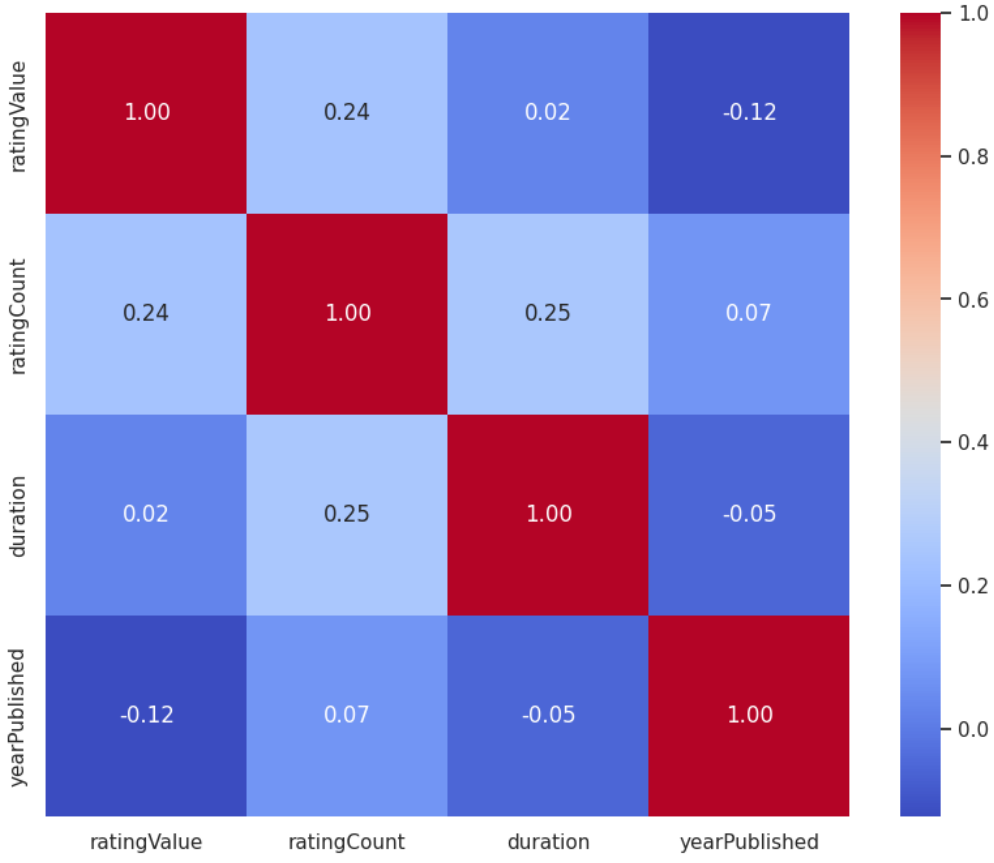


Figure 6: Correlation matrix of numerical columns.

rating.

- ***yearPublished* and *ratingValue*:** A weak negative correlation of -0.12 suggests that more recent films tend to have slightly lower ratings. This is consistent with the trend observed in Figure 5, where ratings for newer films show greater variability.
- ***yearPublished* and *ratingCount*:** The correlation of 0.07 is weak but positive, implying that newer films tend to attract slightly more votes, likely due to their availability and contemporary relevance.

We can see the correlation matrix indicates that while there are some relationships between numerical features, such as between *ratingCount* and *duration*, the correlations are generally weak. This suggests that no single numerical attribute overwhelmingly influences *ratingValue*, and a more nuanced combination of factors should be considered when building the recommendation system.

3.2.2 Users-rating data

Here, we analyze the user dataset to gain insights into user behaviors, preferences, and interactions with films. Understanding user data, such as ratings and watch history, is crucial for building an effective recommendation system tailored to user interests.

As illustrated in Figure 7, the distribution shows the number of users across different ranges of ratings submitted. To improve the quality and reliability of the recommendation system, users who have contributed fewer than or equal to 10 ratings were excluded from the analysis. This removal step ensures that the system primarily focuses on active users who provide more meaningful data for modeling.

Figure 8 presents the distribution of users according to their average ratings. It is evident that most users tend to rate movies between 6 and 8 on average, suggesting a general tendency toward positive ratings. There are significantly fewer users with very low or very high average ratings (close to 1 or 10), indicating that extreme ratings are less common. This distribution highlights a skew toward moderate to high average ratings, which can be an important consideration when designing and evaluating the models.

The Figure 9 appears to be right-skewed, meaning there are more users who tend to give movies higher ratings than lower ratings. The most popular movies are usually highly rated.

The histogram from Figure 10 shows that the majority of movies receive only a small number of ratings, while movies with a very high number of ratings are much less frequent. This pattern is similar to the user distribution, where a small subset is highly active. To ensure the recommendation system focuses on movies with sufficient data for meaningful analysis, movies with an extremely

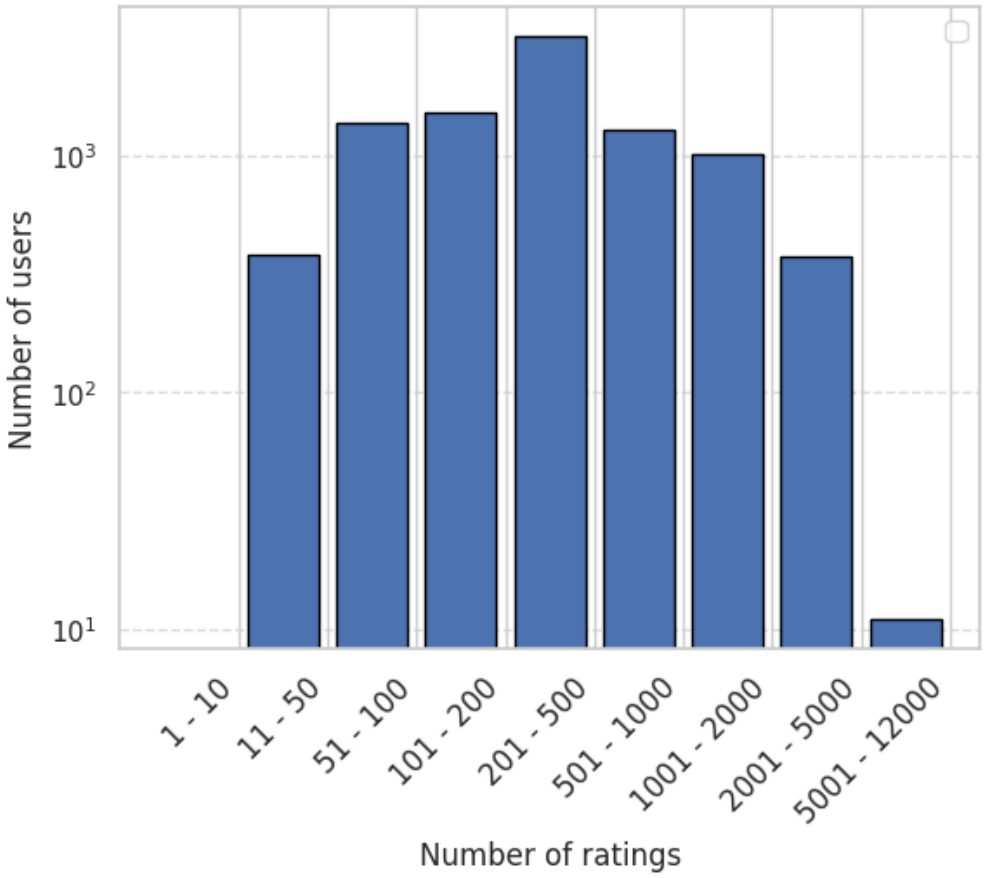


Figure 7: Distribution of users based on the number of ratings provided.

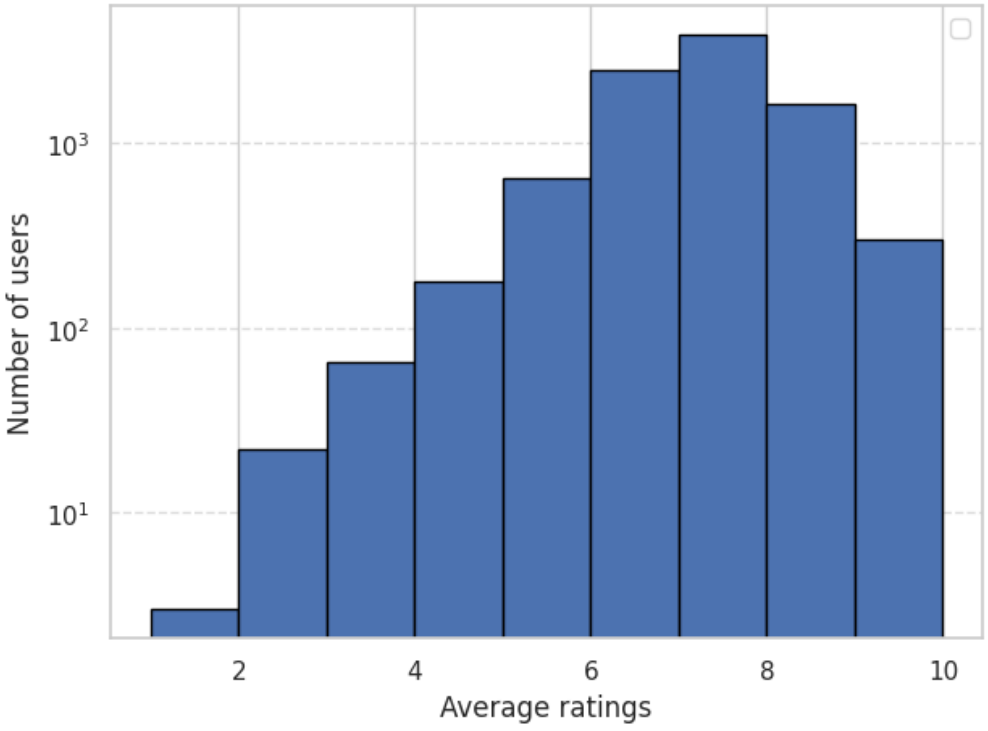


Figure 8: Distribution of users based on average ratings.

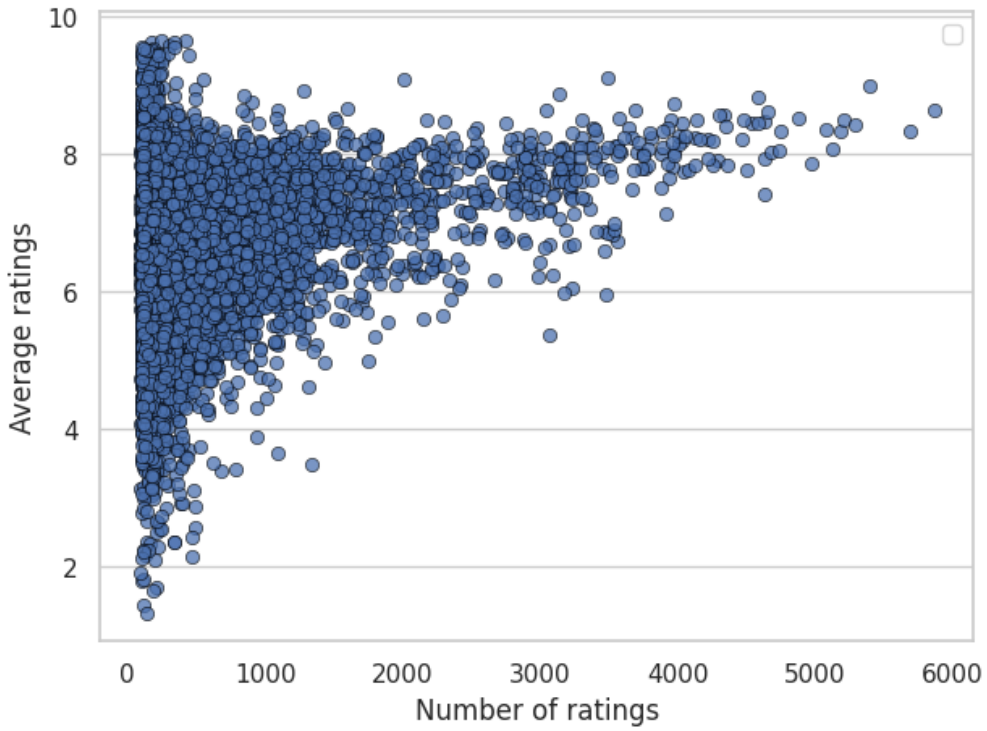


Figure 9: Scatter plot between average ratings and number of ratings.

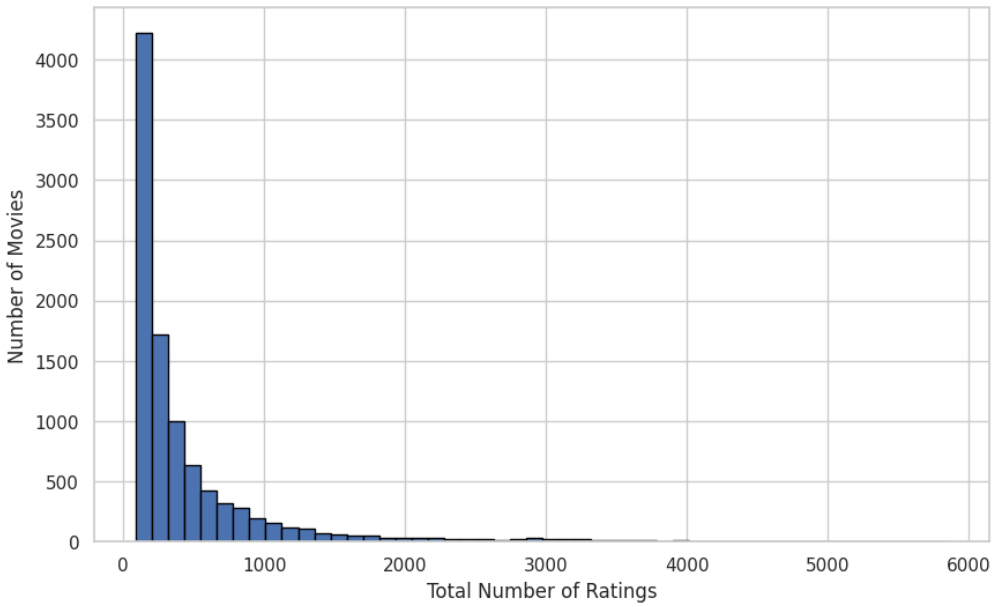


Figure 10: Histogram of Total Ratings per Movie.

low number of ratings have been removed. This step reduces noise in the dataset and enhances the quality of the model by concentrating on movies with substantial user engagement.

4 Models & Evaluation

4.1 Content-Based Approach

The content-based recommendation system in this project focuses on utilizing the features of movies to compute similarities and make predictions. Each movie is represented as a feature vector derived from its attributes (e.g., genres, directors, etc.). The approach is outlined as follows:

- **Compute Movie Similarities**: The feature vectors of all movies are used to calculate pairwise cosine similarities. This creates a similarity matrix that represents how closely related each movie is to others based on their features, based on [2]
- **Fit the Nearest Neighbors Model**: A Nearest Neighbors model is fitted using the computed similarity matrix, with cosine similarity as the distance metric.
- **Generate Candidate Movies**: Given a user ID as input, the movies the user has rated are identified. The Nearest Neighbors model is then used to find a set of candidate movies, denoted as F, that are most similar to the movies the user has rated.
- **Predict Ratings for Candidate Movies**: For each movie f in F, the cosine distance between f and the movies rated by the user is calculated. The k-nearest neighbors of f are identified, and the predicted rating for f is computed as the mean of the ratings the user has given to these k-nearest neighbors.
- **Generate Recommendations**: The output of the system is a list of recommended movies in the form of $(\text{film\_id}, \text{predicted\_rating})$, sorted by predicted rating in descending order.

The feature vector for each film is constructed by processing and combining multiple columns from the dataset. The columns used are: *fid*, *contentRating*, *genre*, *keywords*, *duration*, *actor*, *director*. Each column is transformed as follows:

- **Content Rating:** The *contentRating* column contains categorical labels representing the rating classification of the film. Tags with similar meanings (e.g., "PG-13", "PG") are grouped into common categories to simplify and standardize the values. The standardized content ratings are then transformed using a **OneHotEncoder**, which converts the categorical labels into binary vectors, where each unique content rating is represented as a separate dimension.

- **Duration:** The ‘duration‘ column is a numerical attribute representing the length of the film. To ensure all numerical values are on the same scale, this column is normalized using a **MinMaxScaler**. The scaler maps the duration values to a range between 0 and 1.
- **Genre, Keywords, Actor, and Director:** These columns contain textual information about the film, such as its genre, associated keywords, cast, and director. Each of these columns is processed using **TF-IDF (Term Frequency-Inverse Document Frequency)** to convert the text into numerical feature vectors. TF-IDF emphasizes terms that are unique to a specific film while down-weighting terms that are common across many films, capturing the distinctiveness of the film’s attributes.

After processing all columns, the resulting feature vectors from each transformation step are concatenated to form a single, unified feature vector for each film. This feature vector combines information about the film’s content rating, duration, genre, keywords, cast, and director, enabling a comprehensive representation of the film for similarity computation.

4.2 Collaborative Filtering

Collaborative filtering [5] is a popular technique in recommendation systems that predicts a user’s preferences by analyzing the preferences of similar users. It operates on the idea that users who have similar behavior in the past are likely to have similar preferences in the future. Collaborative filtering can be divided into two main branches: **memory-based** and **model-based**.

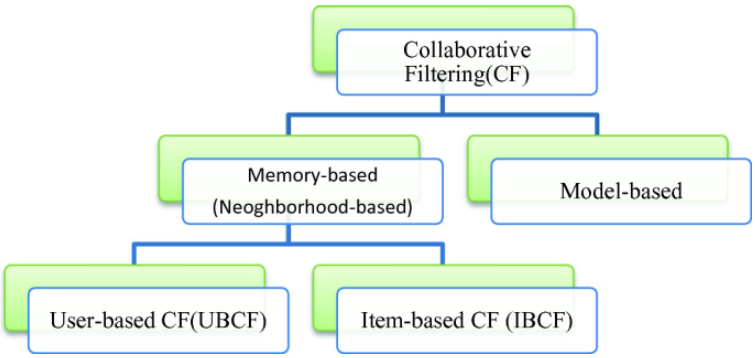


Figure 11: A general classification of collaborative filtering-based recommendations. [4]

4.2.1 Memory-Based

Memory-based collaborative filtering [5], also called Neighborhood-based collaborative filtering is one of the most common recommendation system techniques, which recommends items by similarities between users or items. The basic idea is to recommend items to a user based on the

preferences of other users who are similar to them. This technique can be divided to two approaches: **user-based** and **item-based**. In our project, we focus on **user-based approach**.

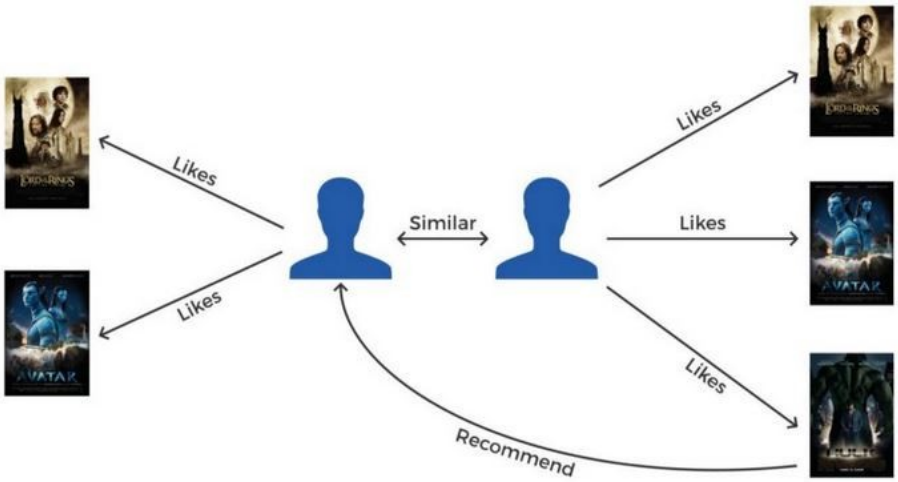


Figure 12: User-based collaborative filtering

The user-based approach begins with the construction of a **user-item rating matrix**. In this matrix, each row represents a user, each column represents an item (in our project, movie), and each cell holds the rating given by a user to a specific item.

Figure 13: User-item rating matrix

Before calculating similarities between users, it's important to normalize the ratings to adjust for individual differences in rating scales and biases. Normalization usually involves subtracting the user's average rating from each of their non-zero ratings, which results in mean-centered ratings. This step ensures that variations in rating tendencies (for example, one user consistently rates movies higher than another) don't distort the similarity measurements.

The normalization formulae for a user's rating r_{ui} of user u for item i is:

$$r'_{ui} = r_{ui} - \bar{r}_u$$

where:

- r'_{ui} is the normalized rating
- r_{ui} is the original rating of user u for item i
- $\bar{r}_u$ is the average of non-zero ratings of user u

The next step involves **computing similarity between users** based on their normalized ratings. To identify users with similar preferences, various similarity metrics can be employed, in this project, I utilized **Cosine Similarity**.

Cosine Similarity quantifies how closely aligned two users' normalized rating vectors are, producing a value between -1 and 1. The formula for Cosine Similarity between users A and B is:

$$\text{cosine\_similarity}(A, B) = \frac{\sum_{i=1}^{n}(A'_i \times B'_i)}{\sqrt{\sum_{i=1}^{n}(A'_i)^2} \times \sqrt{\sum_{i=1}^{n}(B'_i)^2}}$$

where:

- A'_i and B'_i are the normalized ratings given by user A and user B for the same item.
- n is the number of items.

After obtaining the similarity scores, we will identify the target **user's nearest neighbors**. These neighbors are users whose similarity scores with the target user are particularly high, suggesting that they share similar preferences and rating patterns. For instance, if we are generating recommendations for User 1, our analysis may reveal that User 2 and User 3 have the strongest similarity measures, making them the best candidates to serve as reference points for User 1's recommendations.

Next, we **predict the ratings** for items that the target user has not yet evaluated. This estimation is performed by calculating a weighted average of the nearest neighbors' normalized ratings, where each neighbor's influence is determined by its similarity score with the target user. The formula for this rating prediction is as follows:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N(u)} \text{sim}(u, v) \times r'_{ui}}{\sum_{v \in N(u)} |\text{sim}(u, v)|}$$

where:

- $\hat{r}_{ui}$ is the predicted rating of user u for item i
- $\bar{r}_u$ is the average non-zero rating of user u
- $\text{sim}(u, v)$ is the similarity between user u and user v

- r'_{ui} is the normalized rating
- $N(u)$ is the set of nearest neighbors of user u

This formula effectively adjusts the target user's average rating by considering how neighbors' deviations from their own averages influence the predicted rating.

Finally, based on these predicted ratings, the system **recommends items** to the user. The items with the highest predicted ratings, which the user has not yet rated, are suggested as personalized recommendations. This ensures that the recommendations align closely with the user's inferred preferences, enhancing the overall effectiveness of the recommender system.

4.2.2 Matrix Factorization

Matrix Factorization collaborative filtering [3] is a model-based approach widely used in recommendation systems to address the scalability and sparsity limitations of memory-based methods. It decomposes the user-item rating matrix into two lower-dimensional matrices: a **user latent matrix** and an **item latent matrix**. Each user and item are represented as latent feature vectors, capturing hidden relationships such as user preferences and item characteristics.

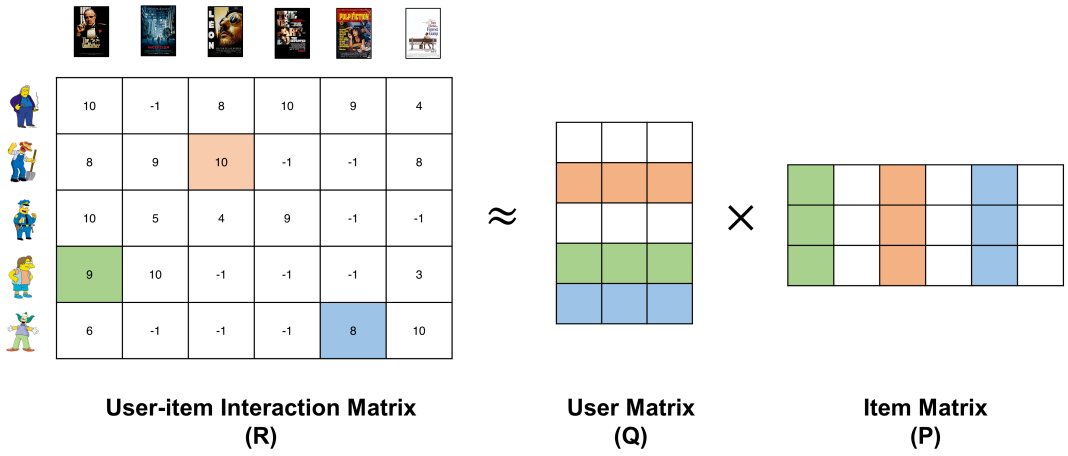


Figure 14: User-item rating matrix is decomposed into user latent matrix and item latent matrix

This method not only captures latent factors that influence user preferences and item characteristics but also incorporates bias terms to improve prediction accuracy. Specifically, the model considers three components: the global average rating, user bias (how much a user tends to rate above or below the average), and item bias (how an item's ratings typically deviate from the global average).

The predicted rating is computed as:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T \cdot q_i$$

where:

- μ is the global average non-zero rating
- b_u is the bias for user u
- b_i is the bias for item i
- p_u is the latent vector of user u
- q_i is the latent vector of item i

By combining these bias terms with the latent factors, the model can better account for user and item tendencies, leading to more accurate and personalized recommendations. This method is highly effective in handling sparse data, which is common in real-world recommender systems like movie recommendation platforms.

4.2.3 Neural Collaborative Filtering

He et al. [1] argue that, Matrix Factorization (MF) can be considered as a linear model of latent factors. The dot product in matrix factorization (MF) is unable to capture the complicated nonlinear relationships, and this can limit the expressiveness of MF. As such, neural modelling approach for collaborative filtering are often used.

In the project, a simple neural collaborative filtering framework based on multilayer perceptron (MLP), similar to original one in [1] is used, modified to compensate for explicit feedback (movie ratings).

We can then formulate the predictive model as

$$\hat{r}_{ui} = f(P^T v_u^U, Q^T v_i^I | P, Q, \theta)$$

where $P \in \mathbb{R}^{M \times K}$ and $Q \in \mathbb{R}^{N \times K}$ are latent factor matrices for users and items, M, N, K are number of users, items and dimension of the latent space; v_u^U and v_i^I are feature vectors describe user u and item i - in our case is just a vector with one-hot encoding. The interaction function f here is a MLP network.

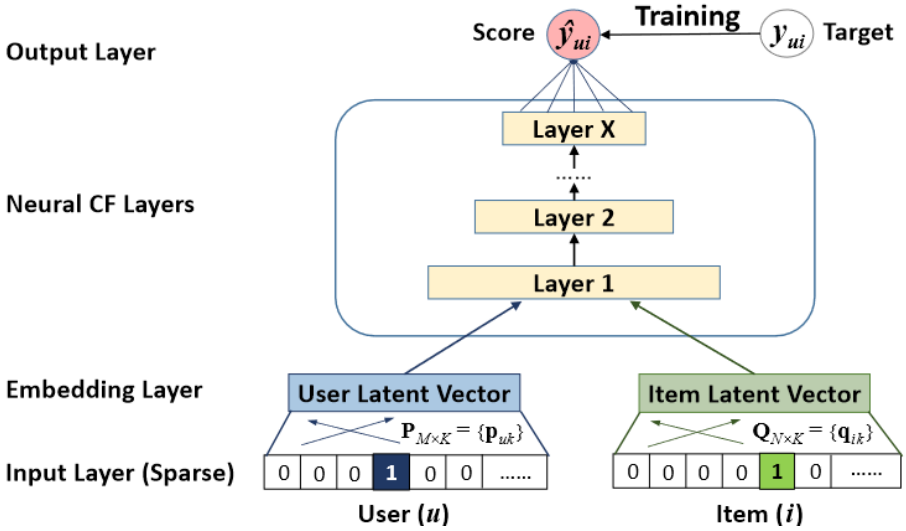


Figure 15: Neural collaborative filtering framework, as described in [1]

By replacing the inner product in MF with a MLP network, the model gains the ability to learn non-linear interactions between users and items, and can capture complex relationships and dependencies that a simple linear dot product cannot, leading to more nuanced and accurate recommendations.

4.3 Evaluation

Assessing the performance of a recommender system is essential to ensure its ability to deliver accurate and relevant suggestions to users. In this project, we utilized four key evaluation metrics: **Root Mean Square Error (RMSE)**, **Recall at K**, **Precision at K**, and **F1-Score**. Each metric provides unique insights into different aspects of the system's performance, allowing for a comprehensive assessment.

Root Mean Square Error (RMSE) measures the accuracy of predicted ratings compared to the actual ratings. It evaluates how well the system predicts numerical ratings for items. RMSE is calculated as:

$$\text{RMSE} = \sqrt{\frac{\sum_{(u,i)\in D}(\hat{r}_{ui} - r_{ui})^2}{|D|}}$$

where:

- $\hat{r}_{ui}$ is the predicted rating for user u and item i
- r_{ui} is the actual rating
- D is the set of all user-item pairs in the test set

A lower RMSE value indicates better accuracy, as it reflects smaller differences between predicted and actual ratings. This metric is particularly useful for evaluating the system's ability to predict explicit feedback, such as star ratings for movies.

Recall at K evaluates the system's ability to retrieve relevant items for a user out of the total number of relevant items available. It highlights the coverage of the recommendations and ensures that relevant items are not missed. A higher recall indicates that the system retrieves most of the items a user might be interested in.

$$\text{Recall@K} = \frac{\text{Number of relevant items retrieved in top K}}{\text{Total number of relevant items}}$$

Precision at K, on the other hand, measures the proportion of relevant items retrieved among the top K recommendations provided by the system. It emphasizes the relevance of the recommended items. A higher precision means that the system provides recommendations that are more aligned with the user's interests

$$\text{Precision@K} = \frac{\text{Number of relevant items retrieved in top K}}{K}$$

F1-Score combines both precision and recall to provide a single metric that balances these two aspects. It is particularly useful when there is a trade-off between precision and recall, as it ensures that both metrics are given equal importance in the evaluation. The F1-Score is calculated as the harmonic mean of precision and recall:

$$\text{F1-Score} = 2 \times \frac{\text{Precision@K} \times \text{Recall@K}}{\text{Precision@K} + \text{Recall@K}}$$

These metrics allowed us to assess the recommender system's capability to generate accurate predictions, retrieve relevant items, and maintain a balance between precision and recall. This comprehensive evaluation offers valuable insights into the system's overall performance and its ability to fulfill user expectations effectively.

Using these metrics, we evaluated the performance of our models. For Recall@K, Precision@K, and F1-Score, we selected $K = 25$ and set the relevance threshold $t = 6$. Due to the nature of the content-based approach used, it was not possible to compute RMSE for this model. The evaluation results are summarized in Table 1.

Model	RMSE	Recall@K	Precision@K	F1-Score
Content-based	—	0.4338	0.9514	0.5959
Neighborhood-CF	1.6703	0.6937	0.9069	0.7862
MF-CF	1.4517	0.6982	0.9202	0.7940
Neural-CF	1.5770	0.7280	0.9035	0.8063

Table 1: Evaluation results for different models

The results highlight the strengths and weaknesses of each approach. The content-based model shows a high Precision@K (0.9514), indicating that it excels at recommending relevant movies among the top-25 suggestions. However, its Recall@K (0.4338) is significantly lower, meaning it struggles to cover all relevant items for users. This trade-off suggests that the content-based model prioritizes precision over recall, which could limit its effectiveness for users with diverse preferences.

The collaborative filtering models show more balanced performance, indicating a good trade-off between precision and recall. The Neural-CF model achieves the highest F1-Score (0.8063) and a strong Recall@K (0.7280), demonstrating its ability to capture complex patterns in user-item interactions. However, its RMSE (1.5770) is higher than MF-CF, indicating that while it ranks recommendations effectively, its ability to predict exact ratings could be improved.

The consistently higher Precision@K compared to Recall@K likely reflects data sparsity in the dataset. With limited user interactions, the models effectively rank a small subset of relevant movies at the top, boosting precision but failing to retrieve all relevant movies, resulting in lower recall. This imbalance may also arise from skewed user preferences, where frequent interactions dominate. Addressing this issue could involve enriching the dataset with more interactions or using data augmentation to mitigate data sparsity.

5 Application

This outlines the development of our server-side flask web, a web application for movie exploration and rating. It empowers users to search for movies, view detailed information, provide ratings, and dynamically update movie lists. This dynamic update is achieved through **AJAX**, enhancing the user experience by minimizing page reloads.

The website is comprised of four key pages :

- Login Page: uses for handle user authentication, allows existed user_id or guest to access the application.

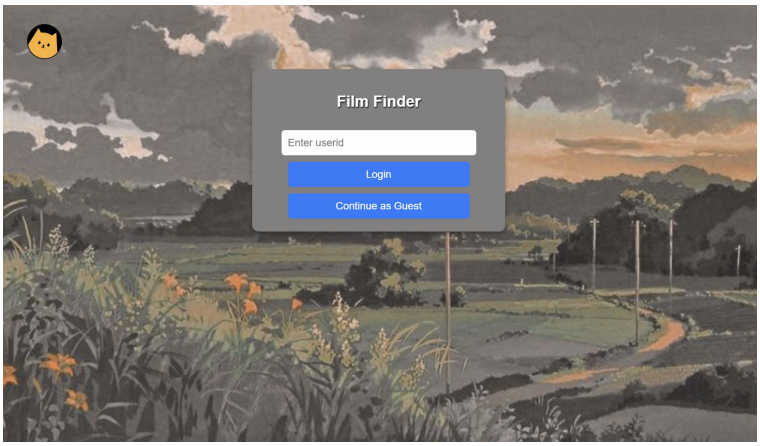


Figure 16: Login Page.

- Main Page: presents different sections including movie suggestions sections to only registered users "You May Like" and the "Highest Rated Films" section for all users. This page also incorporates a search bar with search section for filtering movies when searched.

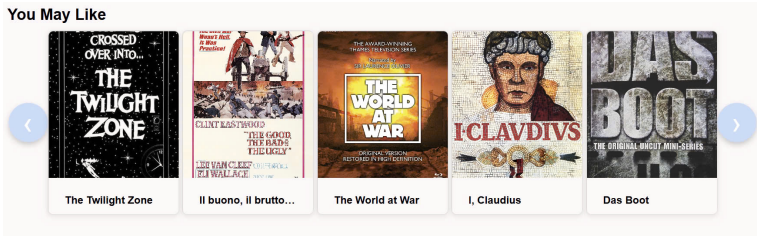


Figure 17: "You May Like" section.

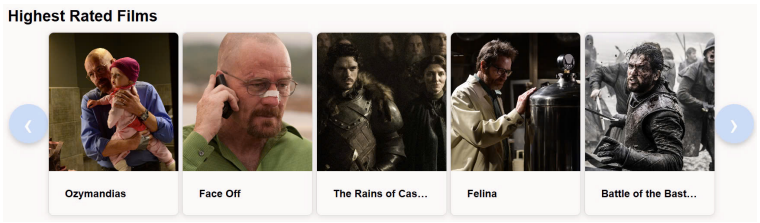


Figure 18: "Highest Rated Films" section.

- Content Page: shows detailed information for a chosen movie such as *name*, *ratingValue*, *ratingCount*, *duration*,*actor*, *director*. Only users (excluding guest) can rate movies on this page, with their ratings stored or modified in the database.
- Rated movies Page: displays all list of movies rated by the logged-in user.

For the back-end, the website is powered by **Flask** and **Flask-SQLAlchemy**. The former one manages routes, user sessions and variables passing, while the latter one make the interactions with the **SQLite** database more controllable, which creates two key tables - movies (for movie information) and ratings (for storing user ratings). This utilize the strength of database for faster query.

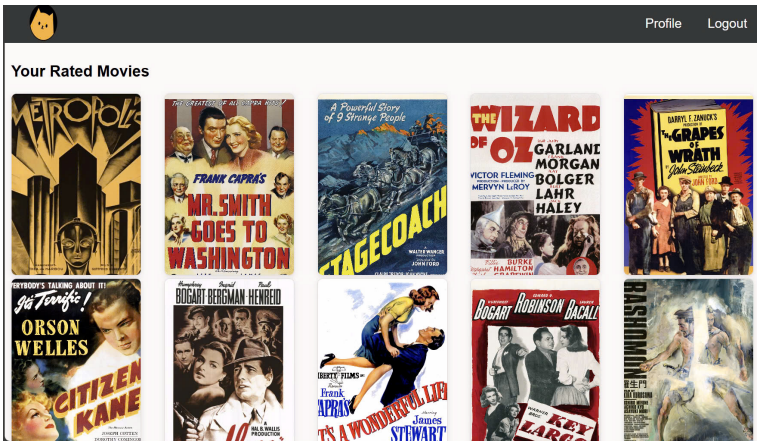


Figure 19: Your Rated Movie Page.

About the front-end, **HTML**, **CSS**, and **JavaScript** are used to create a responsive design that adapts to various devices. **AJAX** plays a crucial role in enabling dynamic updates for "You May Like" and "Highest Rated Films" sections. This approach enhances performance and user interaction by fetching information without reloading the entire page.

Film Finder successfully merges a robust back-end with a dynamic front-end. It offers user authentication, search capabilities, movie rating, AJAX-powered updates, and efficient database handling. The application is easily scalable, allowing future features such as user registration, movie reviews, and personalized recommendations. Overall, Film Finder demonstrates the effectiveness of integrating AJAX for a seamless user experience.

6 Discussion

6.1 Findings

The evaluation of our movie recommendation models highlights distinct strengths and limitations across different approaches. Content-Based Filtering excelled in precision, making it effective for recommending highly relevant items but struggled to cover a broader range of relevant films. Collaborative Filtering methods demonstrated better recall and balance, particularly Matrix Factorization and Neural collaborative filtering (NCF), which captured latent patterns in user-item interactions. NCF stood out as the most robust model overall, effectively leveraging complex interaction dynamics. Overall, the results demonstrate that while advanced models like NCF show promise, simpler models can excel in specific scenarios. The choice of the best model depends on the application that is needed.

6.2 Limitations

Despite the promising results, several limitations were observed during the building and experimenting of our system:

- **Data size and scalability**: The dataset's large size presented computational challenges. Training complex models like NCF required significant resources, limiting the ability to tune hyperparameters or experiment with larger-scale datasets.
- **Data limitations**: Our crawled data exhibited popularity bias, where a small number of popular movies dominated interactions - which lower the models' ability to recommend less popular but relevant films. Also, the data do not have demographic or contextual information about users, such as when or why a user might prefer specific films, or a user's location, age, gender, etc., which could be benefical to our models.
- **Cold start problem**: When new users join the system, their lack of interaction history makes it difficult to understand their preferences and generate accurate recommendations. Similarly, newly added items, such as movies, lack sufficient user engagement data to position them effectively within the recommendation framework. This limitation hinders personalization and content discovery, as the system struggles to connect new users with relevant content or expose new items to appropriate audiences.
- **Evaluation metrics**: Metrics like RMSE, Recall@K, and Precision@K provide valuable insights but do not fully capture user satisfaction or engagement. Metrics such as click-through rate or retention could provide richer insights but were not explored in our project.

Future work could focus on addressing the cold start problem, through hybrid models that incorporate data like user demographics or contextual information. Additionally, exploring more advanced deep learning techniques, such as graph neural networks or attention-based models, could enhance recommendation accuracy. Expanding the dataset and integrating real-time feedback could further improve personalization and adaptability of the system.

7 Conclusion

In conclusion, this project successfully developed and evaluated various movie recommendation models, including content-based, neighborhood collaborative filtering, matrix factorization, and neural collaborative filtering approaches. By leveraging data scraped from IMDb, we constructed a

robust dataset containing user ratings and film metadata, enabling the implementation of these models. Our evaluation metrics, including RMSE, Precision@K, Recall@K, and F1-Score, highlighted the strengths and limitations of each approach, with neural collaborative filtering emerging as a promising technique for balancing accuracy and relevance. Despite achieving notable results, challenges were identified as areas for improvement. Future directions include integrating auxiliary data, exploring advanced machine learning architectures, and expanding the dataset to enhance the system's adaptability and performance. This project underscores the importance of personalized recommendations in improving user experience and contributes valuable insights to the field of recommendation systems.

References

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering, 2017. URL https://arxiv.org/abs/1708.05031.

[2] A. Jiang. Improving diversity through recommendation systems in machine learning and ai, 06 2021. URL https://www.topbots.com/diversity-through-recommendation-systems/.

[3] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

[4] V. Verma and R. Aggarwal. *Neighborhood-based Collaborative Recommendations: An Introduction*. 02 2020. ISBN 978-981-15-3356-3. doi: 10.1007/978-981-15-3357-0.

[5] R. Zhang, Q.-d. Liu, Chun-Gui, J.-X. Wei, and Huiyi-Ma. Collaborative filtering for recommender systems. In *2014 Second International Conference on Advanced Cloud and Big Data*, pages 301–308, 2014. doi: 10.1109/CBD.2014.47.