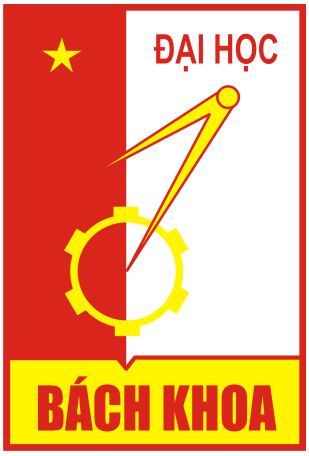
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



INTRODUCTION TO DATA SCIENCE - IT4142E

CAPSTONE PROJECT: Movie recommender system with IMDb movies dataset analysis

Instructor: Ph. D. Nguyen Thi Oanh
Assoc. Prof. Than Quang Khoat

Class: 151946

Students: Nguyen Chi Long - 20210553
Ngo Xuan Bach - 20215181
Le Xuan Hieu - 20215201
Dinh Viet Quang - 20215235
Nguyen Viet Thuan - 20210826

Ha Noi, May - 2024

Contents

Abstract

Movie is a popular form of entertainment and with countless options available on platforms like IMDb, helping users discover films that match their taste has become increasingly valuable. This project focuses on developing a personalized movie recommender system using the movies dataset collected from the IMDb platform. The dataset contains various features including genres, user ratings, cast, reviews and many more factors. Our analysis aims to identify patterns and relationships within the data to provide movie recommendations. To tackle this task, we tried to apply both content-based and collaborative filtering methods and demonstrate by training them on our collected dataset. Collaborative filtering using matrix factorization methods, combined with content-based filtering, yielded the best recommendation accuracy. Our project provides an efficient movie recommendation system while highlighting key factors that influence user preferences, helping film enthusiasts find movies more easily and improving their overall viewing experience through a website interface that provides the movies information and demonstrate our recommendation model. Additionally, we conducted thorough analyses of these models and explored various approaches to achieve better results. After trial and error, we finally got 0.67 and 0.48 for HR@5 score and NDCG@5 score, respectively.

1 Problem Introduction

1.1 Background

IMDb (short for Internet Movie Database) is an online platform that provides information about movies, TV shows, podcasts, home videos, video games, and streaming content [4]. It includes details like cast, crew, biographies, plot summaries, trivia, ratings, and reviews from fans and critics. IMDb started in 1990 as a fan-run movie database on the Usenet group "rec.arts.movies" and moved to the web in 1993. Amazon has owned IMDb through its subsidiary IMDb.com, Inc. since 1998. The site's message boards were removed in February 2017. As of March 2022, IMDb had data on about 10.1 million titles (including TV episodes), 11.5 million individuals, and 83 million registered users [1].

All users can view IMDb's title and talent pages, but only registered users who are logged in can contribute new content or suggest edits to existing entries. Most of the site's information is provided by these volunteers. Registered users with a history of reliable contributions can update cast lists, credits, and certain other details. However, changes to images, titles, cast and crew names, character names, and plot summaries require approval, which typically takes 24 to 72 hours.

1.2 Brief Description

With the growing collection of movies available on platforms like IMDb, finding movies that match individual preferences has become a significant challenge for users. A movie recommender system aims to address this issue by analyzing data and providing personalized suggestions based on user behavior, preferences, and content features. By leveraging the rich IMDb dataset, which includes details such as genres, ratings, cast, crew, and reviews, it becomes possible to uncover patterns and relationships that drive user preferences.

The data for this project was collected from January, 1960 to October, 2024, involving a total of 19250 movies and 376448 user ratings. Information regarding movie details was acquired from `https://www.imdb.com/search/title/?title_type=feature&user_rating=1,10&count=250`. The user rating data of each movie was obtained from `https://www.imdb.com/title/movie_id/reviews/?sort=num_votes%2Cdesc`, where `movie_id` is the ID of the corresponding movie. To ensure the ratings represent a wide consensus of audience feedback, they were sorted by the number of votes. This sorting prioritizes reviews with the most engagement, providing a more reliable and representative measure of public opinion while minimizing the impact of outliers or sparse ratings.

This project involves exploratory data analysis (EDA) to understand key features such as genre popularity, rating distributions, and user review patterns. These insights will guide the development of a robust recommendation engine. Various machine learning techniques, such as content-based filtering and collaborative filtering, will be implemented to provide personalized recommendations.

Challenges in this analysis include handling missing data, managing the diversity and scale of the dataset and addressing the cold-start problem for new users or movies. Additionally, the subjective nature of user preferences adds complexity, as it requires the model to balance between general trends and individual tastes.

The input data for our recommender system is either a movie title or a user ID. If the users do not have an IMDb account, they can enter a movie title, and our system will display a list of similar movies. Alternatively, if the users have IMDb account, they can enter their user ID, and our system will provide a list of movies based on their rating history.

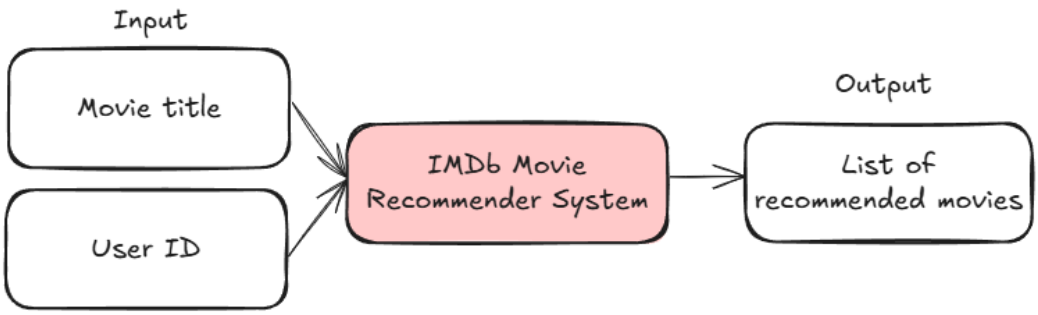


Figure 1: The input and output IMDb Recommender System

The study aims to build a high-performing movie recommender system that not only predicts user preferences accurately but also provides valuable insights into the features that most influence movie popularity and user engagement. Ultimately, the system will enhance the user experience by simplifying the process of discovering movies and contributing to a deeper understanding of audience behavior in the film industry.

1.3 Outline

We organize the report as follow:

Section 2: Describe how we collected data from the source and and how to clean them

Section 3: Summary how row data have been integrate for further analysis

Section 4: Demonstrate exploratory data analysis for getting the property and valuable insights helping build more effective model.

Section 5: Outline the methodology used for each experiment in the project, providing a step-by-step explanation of the modeling, parameter fixing, and prediction processes for each model used.

Section 6: Describe the main workflow of our work as well as the Web demonstration.

2 Data Collection & Cleaning

The desired information is collected from the official archive of IMDb page (`https://www.imdb.com`). We then transformed the collected data to a tabular form and performed preprocessing, making it consistent across multiple sources and easily usable for the following stages. Besides, missing records or inconsistent patterns are critical aspects and if necessary, we tried to re-collect them.

2.1 Data Collection

The dataset required for our objective consists of three components: movie URLs, detailed movie metadata and user rating data. These components were collected in stages and integrated to form the complete dataset necessary for building the movie recommendation system.

The first source, movie URLs, was initially crawled from IMDb's search results page from January, 1960 to October, 2024 at `https://www.imdb.com/search/title/?title_type=feature&user_rating=1,10&count=250`. This webpage provides an up-to-date and specific list of films rated by IMDb users. To collect all movie URLs, we implemented a web scraping pipeline using Selenium due to the dynamic structure of the IMDb website. The scraper navigated the search results, automatically expanding the page by clicking the "See More" expanding button to load additional results. Each movie's URL was extracted from the anchor tags on the page (**19250 movies in total**) and stored in a CSV file (`movies_url_data.csv`) for further processing.

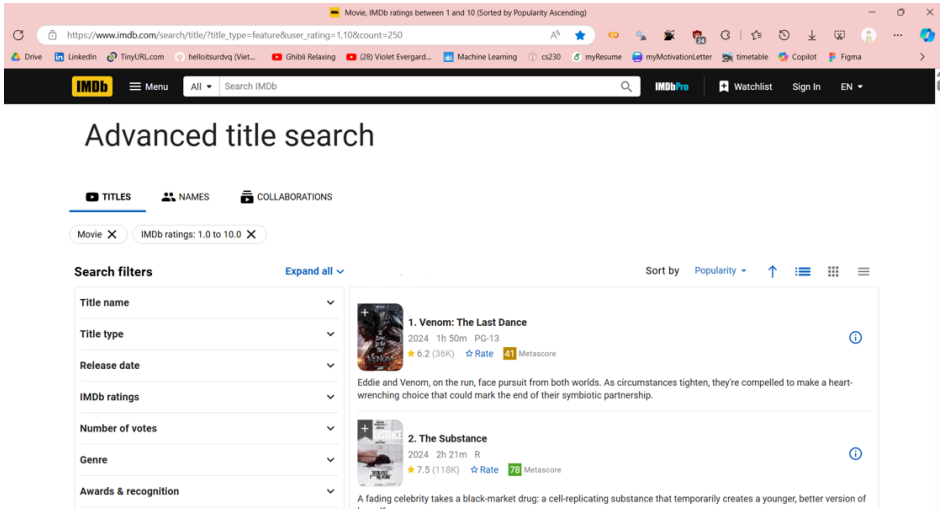


Figure 2: The first page of IMDb for movie URLs scraping

In the second stage, movie details, was crawled from individual IMDb movie pages using the Scrapy framework. Each movie's URL from the previous step in `movies_url_data.csv` file was visited to extract extensive information about the movie such as the movie's title, year of release, director, cast, duration, genres, plot summary, budget, gross revenue, country of origin, awards, and IMDb rating metrics. The description for all 19 fields in total is summarized in Table 1. Special attention was given to the genre field, filtering genres based on a predefined set of popular categories such as Action, Drama, and Comedy. Missing or incomplete data fields were handled by assigning default values, preserving dataset consistency. This stage produced a rich tabular dataset (`movies_details.csv`) containing the full metadata for all collected movies.

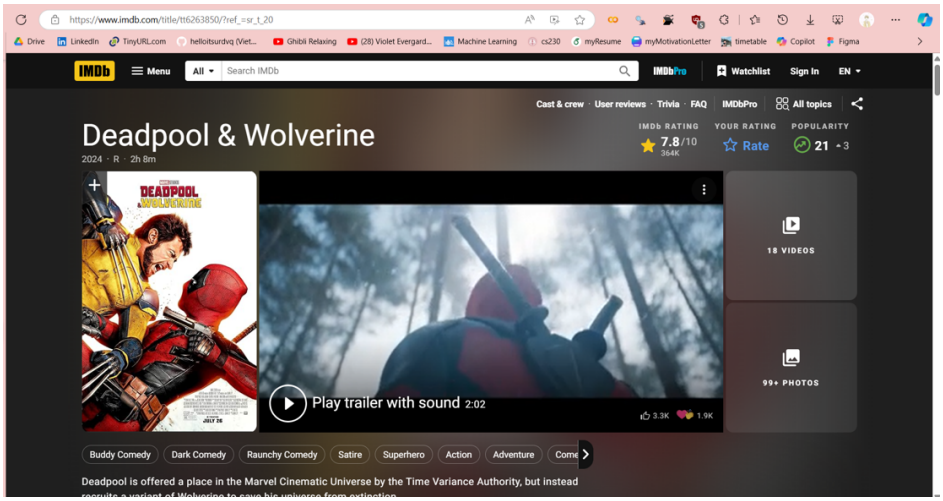


Figure 3: An example of movie's detail page

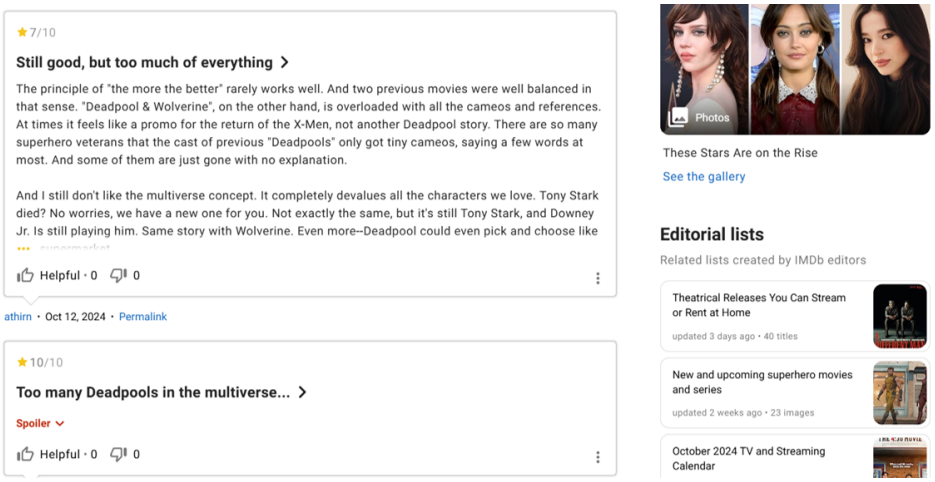


Figure 4: An example of an user rate for movie

Table 1: Movie's detail dataset

Field	Description	Example
title	Title of the movie	*Inception*
img_url	URL of the movie's poster image	`https://image-url.com/poster.jpg`
year	Release year of the movie	2010
director	Name of the movie's director(s)	*Christopher Nolan*
stars	List of main actors or stars in the movie	`["Leonardo DiCaprio", "Elliot Page"]`
duration	Runtime of the movie (in minutes)	148
genres	List of genres the movie belongs to	`["Action", "Sci-Fi", "Thriller"]`
overview	Short summary or plot description of the movie	*Lorem ipsum...*
rating	IMDb rating of the movie	8.8
num_rating	Number of user ratings	2,100,000
num_user_review	Number of user reviews	1500

Field	Description	Example
num_critic_review	Number of critic reviews	350
budget	Estimated budget of the movie (in USD or specified currency)	$160,000,000
gross	Total gross revenue of the movie (in USD or specified currency)	$836,800,000
country	Country where the movie was produced	*USA*
metascore	Metascore rating from Metacritic	74
oscar	Number of Oscars won by the movie	4
win_and_nomination	Total number of awards won and nominations received	152 wins & 204 nominations
url	IMDb URL link to the movie's detail page, containing the ID of the movie	`https://www.imdb.com/title/tt1375666/`

The third stage involved gathering user rating data from each movie's review page at URLs formatted as `https://www.imdb.com/title/movie_id/reviews/?sort=num_votes%2Cdesc`, where `movie_id` is the ID of the corresponding movie. Sorting the reviews by the number of votes ensured that only the most representative and reliable ratings were captured, minimizing the impact of sparse or outlier data. User ratings, along with corresponding user IDs and movie ID (described in Table 2), were extracted and stored, providing deeper insights into how users rated movies, capturing trends and patterns in audience sentiment. In general, we have successfully collected **376448 ratings** in our project for further research .

Table 2: User Rating dataset

Field	Description	Example
user_id	Unique identifier for each user	`"ur2125550"`
movie_id	Unique identifier for each movie, extracted from the 'url' in movie's detail dataset	`"tt0036172"`
rating	Rating given by the user to the movie	9

The data collection process faced several challenges. One of the main issues was the dynamic nature of modern IMDb's website, which required using Selenium to interact with content generated by JavaScript. Another challenge was dealing with inconsistencies in the metadata, such as missing director names or incomplete awards information. These gaps were handled by assigning default values to maintain consistency.

2.2 Data Cleaning

Data consistency is one of the ten key aspects of data quality. In our context, it refers to ensuring that data complies with required formats and values, maintaining uniformity between the original and updated versions of a record stored internally or across multiple storage locations. To achieve this, we evaluate the two datasets to fix structural errors and create a well-designed interface so that we are able to understand the information more easily and accurately. The two datasets we utilized will be discussed in detail in the following 2 subsections. We also present the techniques we take to process the data for the model training.

2.2.1 Data Validation

When scraping movie details with the provided code, handling missing or inconsistent data is crucial to ensure the dataset is clean and usable for downstream applications, such as analysis or machine learning. This section discusses our strategies for managing such issues effectively.

Identifying Missing or Inconsistent Data

- **Missing Data:** Fields such as director, budget, or gross revenue might be unavailable for certain movies. This can occur due to differences in webpage structures or gaps in IMDb's information.
- **Inconsistent Data:**
  - Formatting differences, such as varying units for duration (e.g., ‘‘`1h 30min`’’ vs. ‘‘`90 min`’’), can occur.
  - Variability in genres or stars due to scraper logic or unrecognized data.

Handling Missing Data Assign default values to ensure consistent schema:

- Strings: Use ‘‘`N/A`’’ for unavailable textual data (e.g., director, genres).
- Numeric: Use `0` for missing numeric fields (e.g., rating, number of reviews).
- Lists: Use an empty list for fields like stars and genres if no values are found.

Example:

```
director = soup.find('a', class_='ipc-metadata-list-item__list-
    content-item--link')
director = director.text if director else "N/A"
```

Structured Validation" Validate each scraped field such as

- Ensure numeric fields, such as `rating`, are properly parsed and within the expected range (e.g., 0–10).
- Validate lists, such as `genres` or `stars`, to include only recognized values.

Fallback Mechanisms: Use alternative webpage elements to fetch missing data. For example, if the primary tag for the release year fails, check nearby tags.

Handling Inconsistent Data

- Convert duration into a unified unit, such as minutes.
- Parse budgets and grosses to numeric values, handling currency symbols and commas.

Example:

```
duration = duration.replace("h", "").replace("min", "").strip()
```

Data Cleaning Rules Implement rules to clean text fields

- Trim extra whitespace or remove special characters.
- Map genre names to a predefined list for consistency.

Graceful Degradation Skip or log problematic links without halting the scraping process, ensuring partial results are usable.

Post-Scraping Data Validation - Perform checks to identify anomalies:

- Movies with ‘‘`N/A`’’ in critical fields (e.g., title).
- Outliers in numeric fields like ratings or budgets.

Imputation or Removal: Decide on imputation (e.g., filling missing numeric ratings with the average) or removal of entries with excessive missing data based on the use case.

2.2.2 Data Preprocessing

First of all, to **Handle missing data**, there are some approaches that we used:

- **Dropping rows with missing values:** Rows with missing values in essential columns such as `title`, `genres`, `rating`, `year`, `overview`, and `duration` were removed to maintain data quality
- **Filling missing budget units:** Missing values in the `budget_unit` column were assumed to be in USD and filled accordingly:

```
df_movie["budget_unit"].fillna("USD", inplace=True)
```

Data Conversion and Transformation

- **Conversion of rating count to integers and currency conversion for budget and gross** values in the `num_rating` column (e.g., `1.5K`, `2M`) were converted to integers:

```
def convert_to_int(value):
    if "K" in value:
        return int(float(value.replace("K", "")) * 1000)
    elif "M" in value:
        return int(float(value.replace("M", "")) * 1000000)
    else:
        return int(value)
df_movie["num_rating"] = df_movie["num_rating"].apply(
    convert_to_int)
```

The `budget` and `gross` columns were normalized to USD using predefined exchange rates:

```
def convert_budget_to_usd(row):
    if row["budget_unit"] in currency_conversion:
        return float(row["estimated_budget"]) *
            currency_conversion[row["budget_unit"]]
    return float(row["estimated_budget"])

df_movie["budget_usd"] = df_movie.apply(convert_budget_to_usd
    , axis=1)
```

- **Extraction of Numerical Values:** The `gross` column, containing both numerical and unit values, was split into `estimated_gross` and `gross_unit`:

```
df_movie["estimated_gross"] = df_movie["gross"].str.extract("
    (\d+)")
df_movie["gross_unit"] = df_movie["gross"].str.extract("(\D+)
    ")
```

- **Extraction of Awards and Nominations:** Information about wins and nominations was extracted from the `win_and_nomination` column into separate columns:

```
wins = df_movie["win_and_nomination"].str.extract(r"(\d+)
    wins?")
nominations = df_movie["win_and_nomination"].str.extract(r"(\
    d+) nominations?")
wins.fillna(0, inplace=True)
nominations.fillna(0, inplace=True)
df_movie["win"] = wins
df_movie["nomination"] = nominations
```

- **Duplicate removal**: Duplicate entries based on the `movie_id` column were removed to ensure data integrity:

```
df_movie = df_movie.drop_duplicates(subset=["movie_id"])
```

- **Column cleanup**: After processing, unnecessary columns such as `budget_unit`, `gross_unit`, and `win_and_nomination` were dropped to reduce the complex of the dataset:

```
df_movie.drop(["url", "budget", "gross", "estimated_budget",
    "estimated_gross"], axis=1, inplace=True)
```

We have implemented these preprocessing steps to ensure that the dataset is ready for exploratory data analysis and modeling by addressing issues like missing values, inconsistent data types and feature extraction.

3 Exploratory Data Analysis

3.1 Movies data

Looking at the word cloud in Figure 5, we notice interesting insights into the most common keywords found in movie overviews that point out recurring themes, relationships, and story elements. The most prominent word, "life", suggests that movies often revolve around exploring the meaning, struggles, or experiences of life. Similarly, we notice words like "find", "family", "love", and "friend", emphasizing themes of connection, relationships, and personal growth, which are central to many film narratives. We also see the presence of words like "woman", "man", "young", and "father", indicating that familial roles and characters across different age groups play a significant part in movie plots. Moreover, verbs like "discover", "become", "live", and "take" maintain the idea of personal change, self-discovery, and adventure, which is basically the key driver in character development and story arcs. Other relevant words, such as "murder", "world", "story", and "death", are representative of popular genres, including mystery, thriller, and drama. In general, one would say that movies tend to address the very basic human feelings and experiences, such as love, relationships, growth, and adversities, but also include the components of suspense, action, and discovery that intrigue viewers. These insights will help in identifying patterns and preferences in building a recommendation system because these are in line with audiences' common interests and viewing behaviors.

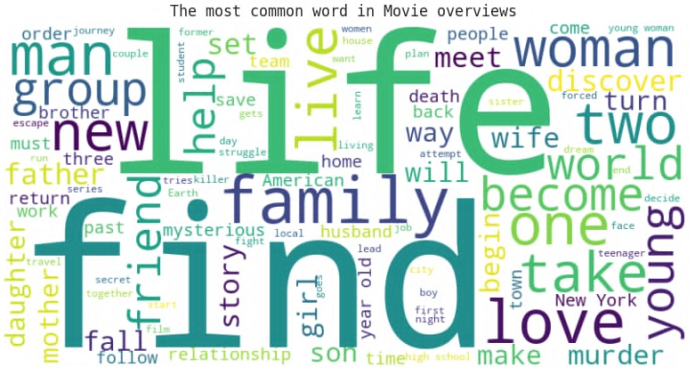


Figure 5: The most common word in Movie overviews

The violin plot below illustrates the distribution of ratings between Oscar-winning movies, labeled "1", and non-Oscar-winning movies, labeled "0". From the Figure 6, we see a good separation in rating distributions among the two classes. The ratings for the non-Oscar movies (0) are more spread out and centered between 5 to 7. There is also the presence of low outliers; this suggests that some

movies not considered as Oscar movies could even obtain very low ratings close to 0. Moreover, in comparison with Oscar-winning movies, the median rating of a movie for non-Oscar movies seems to be somewhat lower. In contrast, the Oscar-winning movies (1) show a much tighter and higher rating distribution. The ratings for Oscar movies are concentrated around 7 to 8, with very few ratings falling below 6. This suggests that Oscar-winning movies tend to consistently receive higher ratings, with very little variation compared to their non-Oscar counterparts. From this, we may infer that Oscar-winning movies tend to have higher ratings among audiences, perhaps because they are perceived as more qualitatively better, well-told, or critically acclaimed. Non-Oscar movies are more spread out, which could be due to the broader range of quality and reception. This can also be helpful in understanding the preferences of audiences and factors that affect movie ratings, which is helpful for a recommendation system.

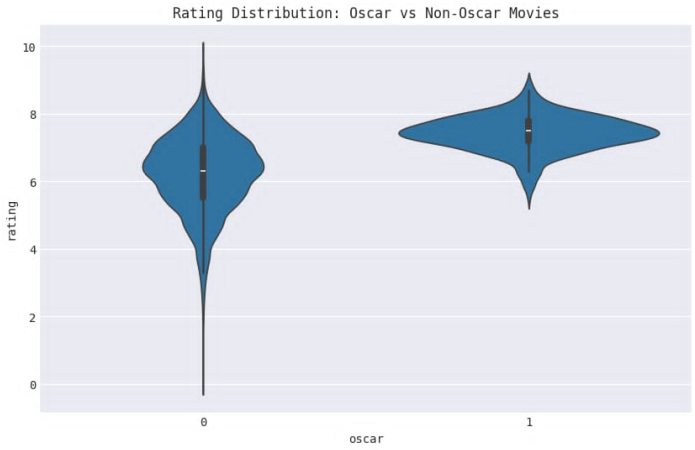


Figure 6: Rating Distribution: Oscar vs Non-Oscar Movies

The figure 7 reveals a significant rise in movie production, particularly after 2000, with an exponential spike post-2010. Among the five genres analyzed, *Drama* consistently dominates, surpassing 120 movies by 2020, highlighting its enduring appeal and adaptability. Other genres, such as *Drama - Romance* and *Comedy - Drama*, show moderate growth with notable fluctuations, while *Comedy* exhibits sustained popularity but remains below Drama. Prior to 1950, production across all genres was minimal, with steady growth observed from the 1950s to the 1990s, followed by a sharp inflection point in the early 2000s, coinciding with technological advancements, the rise of digital platforms, and democratized access to filmmaking. This rapid growth underscores evolving audience preferences, the prominence of hybrid storytelling, and increased production diversity. For movie recommender systems, these trends emphasize the need to account for temporal biases, genre dominance, and the sheer volume of content in recent decades to ensure balanced and accurate recommendations. Understanding these patterns is crucial for aligning modern recommender models with shifting audience behaviors and production dynamics.

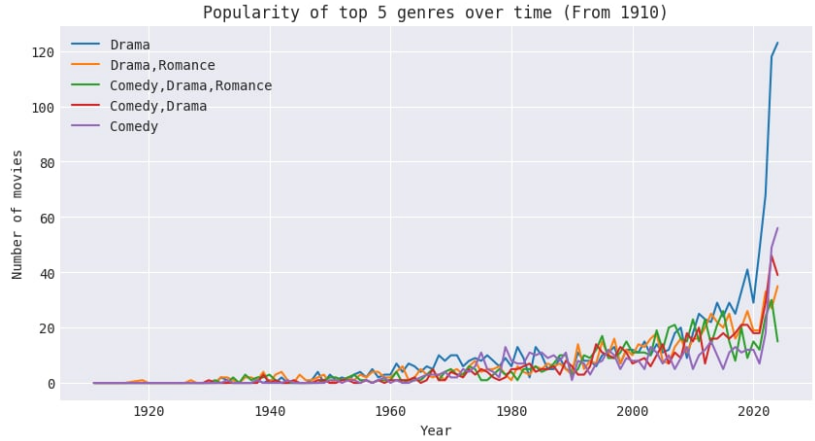


Figure 7: Popularity of Top 5 Genres Over Time

The Figure 8 shows that Drama is the most dominant genre, with over 10,000 movies, far exceeding the counts of other top genres. Comedy and Thriller follow with roughly 6,000 and 5,000 movies, respectively, while Romance and Action account for smaller totals of approximately 3,500 and 3,000 movies. This significant disparity highlights Drama's strong presence in the movie industry, likely due to its versatility in storytelling and broad appeal across diverse audiences.

The pie chart on the right, titled "Proportion of Movie Genres," provides further context, showing that Drama constitutes 19.5% of all movies, while Comedy and Thriller each hold approximately 11% of the total. Romance and Action represent smaller shares at 7.4% and 7.0%, respectively. Interestingly, the "Other" category makes up 43.6% of the total, emphasizing the wide range of less dominant genres that collectively contribute to the overall movie landscape.

These observations underline two key insights. First, the prevalence of Drama, Comedy, and Thriller indicates their sustained popularity and production focus, which may introduce biases in movie recommender systems unless adjusted for diversity. Second, the substantial share of "Other" genres underscores the importance of niche categories, which cater to varied audience preferences and should not be overlooked. Balancing popular and niche genres in recommendation algorithms is essential for achieving comprehensive and user-centric recommendations.

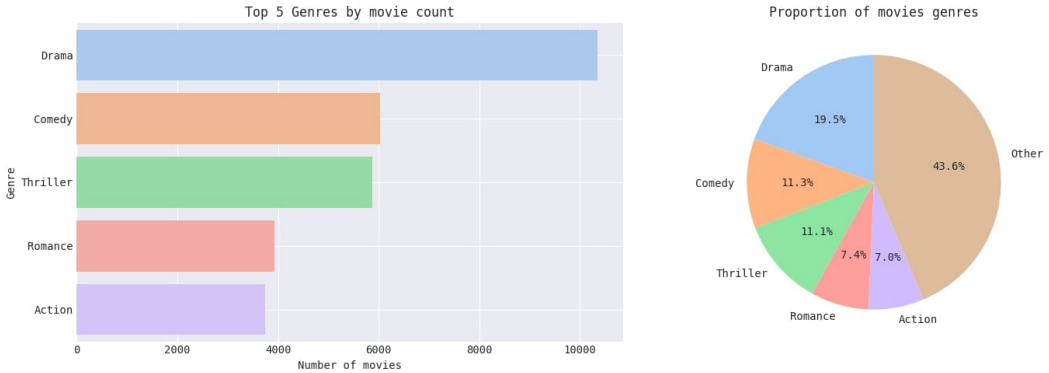


Figure 8: Top 5 genres by movie count

The Figure 9 highlights the most prolific actors based on the number of movies they have appeared in. Nicolas Cage leads the list with close to 90 appearances, significantly outpacing the other stars. Following him are Robert De Niro and Bruce Willis, with approximately 83 and 80 appearances, respectively, reflecting their long-standing careers and high involvement in the movie industry.

The next group includes Samuel L. Jackson with 62 appearances, followed by actors such as Morgan Freeman and Tom Hanks, both contributing around 57 to 58 movies. At the lower end of

the top 10 list are Dennis Quaid, Anthony Hopkins, Nicole Kidman, and Michael Caine, all with appearances hovering around 55. While the difference between these actors is not large, their presence reflects consistent and sustained popularity across decades of filmmaking.

This analysis underscores the importance of these actors in driving movie viewership, as their frequent appearances suggest strong audience demand for their performances. For movie recommender systems, this trend is relevant because starring actors often serve as a significant factor in influencing user preferences. Additionally, the disparity in appearances—ranging from 90 at the top to around 55 at the lower end—highlights the varying levels of output even among highly recognizable stars. Incorporating actor-based metrics can therefore enhance the personalization of movie recommendations, particularly for users who follow specific stars' filmographies.

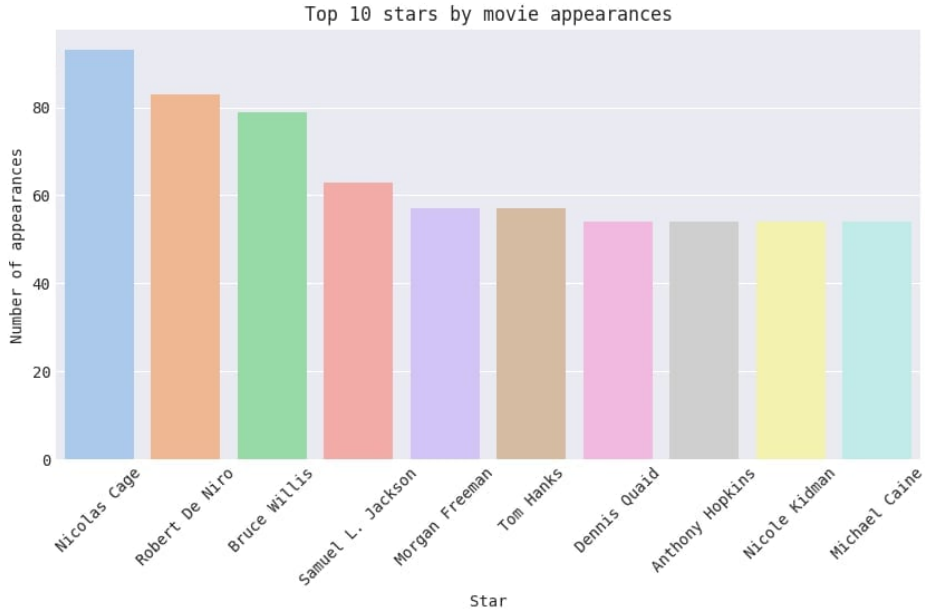


Figure 9: Top 10 stars by movie appearances

The Figure 10 provides an overview of the runtime of movies in the dataset. The distribution shows a clear right-skewed pattern, where the majority of movies fall within the range of 80 to 120 minutes. The peak occurs at approximately 100 minutes, suggesting this duration as the standard runtime for most feature-length films. This aligns with industry conventions, where movies of around 90 to 120 minutes are generally well-received by audiences.

Shorter movies, below 60 minutes, are considerably less common, likely representing documentaries, short films, or experimental works. In contrast, longer movies, extending beyond 150 minutes, occur infrequently, with their counts diminishing rapidly as duration increases. Movies exceeding 200 minutes are rare, reflecting niche productions such as epics, extended director's cuts, or special-purpose films aimed at highly dedicated audiences.

This analysis highlights the importance of movie duration as a key feature in movie recommendation systems. Users may prefer movies within a certain duration range based on their time constraints or attention spans. The heavy concentration of movies around the 90 to 120-minute range also indicates the need for recommender systems to account for runtime preferences while balancing recommendations that include both mainstream feature-length films and niche longer or shorter productions. Understanding the distribution of durations allows for better user experience by aligning suggested movies with audience viewing patterns.

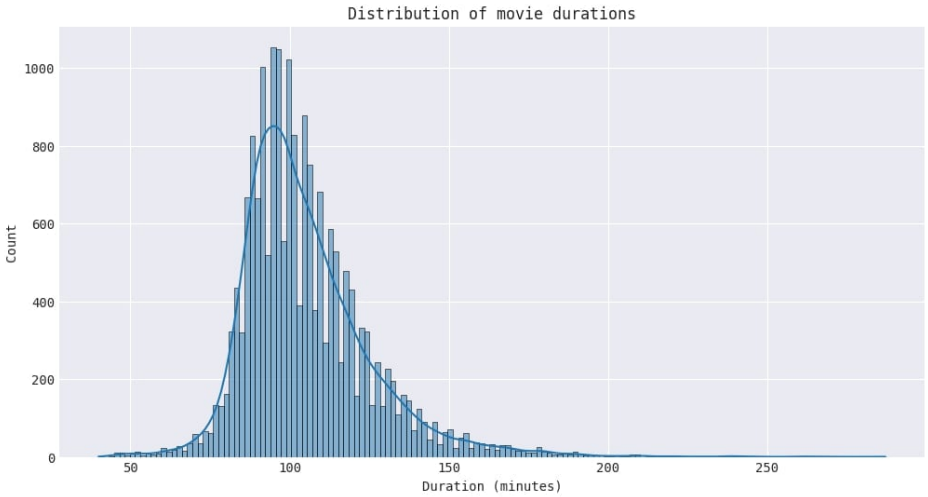


Figure 10: Distribution of movie duration

The figure 11 provides insights into the distribution of ratings across the genres Drama, Comedy, Thriller, Romance, and Action. The density plots reveal that the majority of movies across all genres cluster within the average rating range of 5 to 7, with peaks between 6.0 and 7.0. Notably, *Drama* (blue) and *Romance* (red) exhibit sharper peaks around 6.5, suggesting a concentration of ratings slightly higher than the other genres. *Thriller* (green) and *Action* (purple) display a broader spread, with ratings extending more noticeably towards the lower end, indicating greater variability in audience reception. In contrast, *Comedy* (orange) shows a slightly narrower distribution, reflecting consistent yet modest ratings clustered closer to the 6.0 mark. The density patterns highlight that while Drama and Romance maintain a relatively stable and higher rating trend, genres like Action and Thriller are subject to greater variance, which may arise due to differing expectations or production quality within these genres. These findings are critical for recommender systems, as understanding the variability in genre ratings allows for more precise recommendations, accounting for user preferences and rating tendencies. Moreover, the concentration of ratings within a limited range underscores the importance of leveraging other features, such as user demographics and temporal trends, to differentiate movie recommendations more effectively.

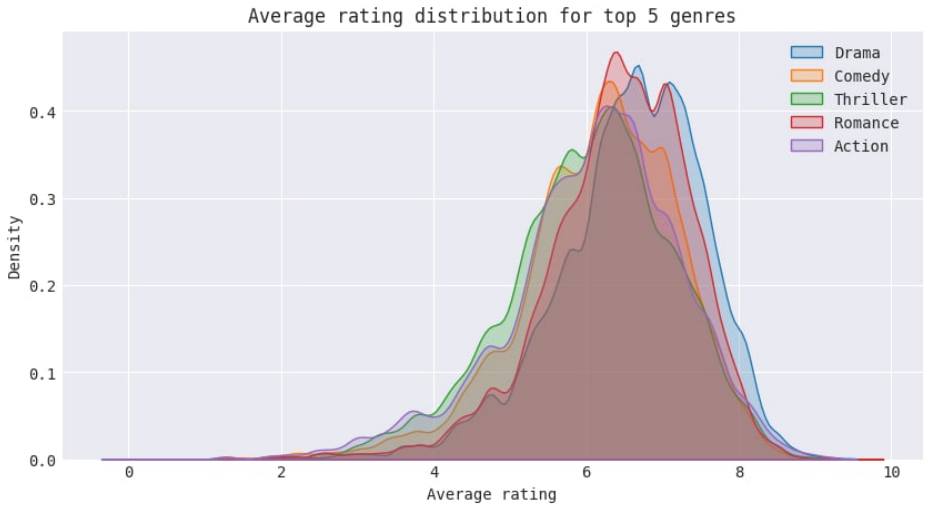


Figure 11: Average Rating Distribution for Top 5 Genre

3.2 Movies with user rating data

Figure 12 indicates that the distribution of user ratings is highly skewed, with ratings of 1 and 10 dominating the chart. Rating 10 has the highest frequency, exceeding 60,000 counts, followed closely by rating 1. This pattern suggests that users tend to provide extreme ratings, likely reflecting strong emotional reactions or polarized opinions about the movies they watch. It also gives interesting information on the middle-grade ratings, which are pretty seldom in appearance: the counts between 2 and 6 keep floating between 25,000 and 30,000, meaning users prefer to not give a movie an average grade. However, we notice a definite upward trend beyond rating 6, with a significant increase at 7 and 8, reaching approximately 40,000 and 50,000 counts, respectively, before peaking at 10. This could suggest a general tendency toward positive evaluations; thus, users are more likely to express satisfaction than dissatisfaction. In this process, we have tried to gain insight into the user rating behavior in order to come across biases in rating distribution. These extreme tendencies to give extreme ratings, along with lower mid-level scores, highlight one more challenge a recommender model may face: the polarized nature of users. Further, in dealing with these biases, we try to be more accurate and fair in recommending movies.

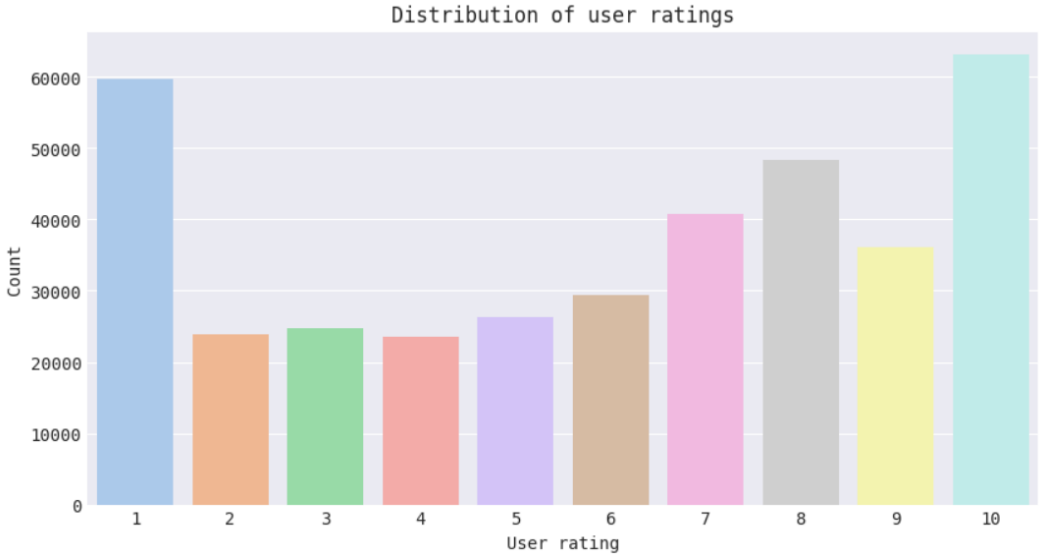


Figure 12: Distribution of user ratings

In Figure 13, we notice a distinct downtrend in the average movie ratings across the years. From the early 1900s until the 1950s, average ratings were high, bouncing between 7.0 and 8.0, with significant highs in the 1920s where the average was over 8.0. However, starting in the 1960s, we notice a shift, as the average ratings are consistently below 7.0. This trend continued for the next several decades, before leveling off in the range of about 6.0 from the 1980s through to the 2020s. We are interested in trying to identify the reasons for this long-term decline in ratings. A first hypothesis might be that the tastes and preferences of moviegoers have shifted over time, becoming either more critical or more diverse. Also, the enormous rise in films produced over the decades would bring in much greater variation in quality, thereby pulling down the average quality. Another aspect could be the democratization of ratings, meaning a greater and more diverse number of users in the modern digital era would end in far more consistent, but a little lower, ratings. This, in turn, underlines that, while exploring historical ratings for movie recommendations, one should not ignore the biases arising due to time. The tendencies of rating movies have changed over time, and these temporal changes may affect the recommendation models. These insights provide a basis for understanding generational changes in rating behavior and, therefore, enable us to make more accurate observations while assessing movies that span many years.

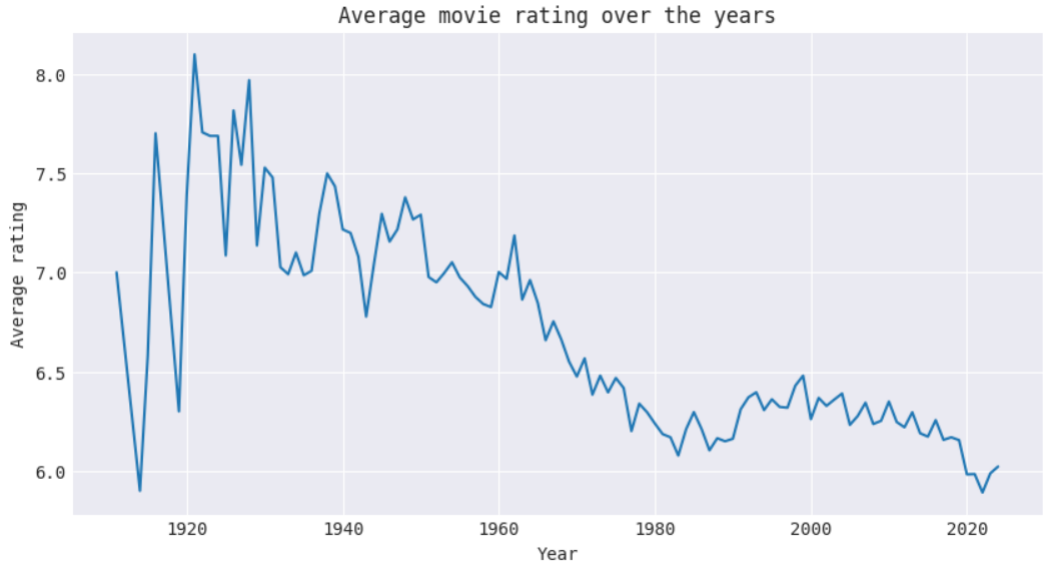


Figure 13: Average movie rating over the years

Figure 14 shows the top-rated movies based on users' ratings. It is a bar chart showing on the x-axis the number of ratings and on the y-axis, the movie titles. It is observed that Shawshank Redemption is top, which has received maximum ratings from the users among all the movies. In second, there are two movies, namely Dark Knight and Inception. The next tier of movies consists of Fight Club, Pulp Fiction, and Forrest Gump, but these are considerably further behind the top three in terms of ratings. The last four movies to complete the top 10 were Interstellar, The Lord of the Rings: The Fellowship of the Ring, The Matrix, and The Godfather. These prove that classic movies do hold a special place in users' hearts and, in fact, keep their relevance and audience appeal over time. Based on the chart above, one could immediately see that users have strong preferences for movie genres in the following categories: drama, action, and science fiction. It would also tend to imply that users would go towards critically acclaimed and evergreen classics, as many films evoke emotion or visual stimulation. With this in mind, an analysis will be conducted of user preference to further define their genre and theme tastes. These insights are invaluable in refining movie recommendation systems, helping to better tailor suggestions to user tastes.

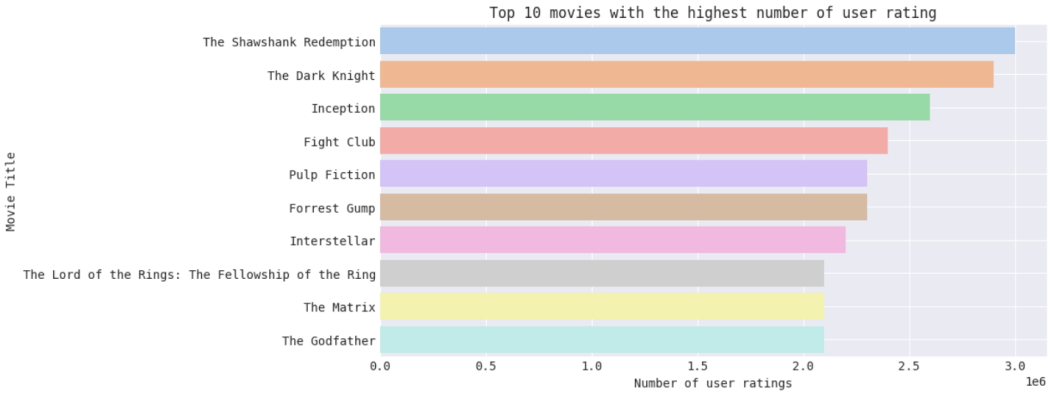


Figure 14: Top 10 movies with the highest number of user rating

Looking at Figure 15, we observe that the News genre leads in the highest average rating, closely followed by Documentary and Film Noir, which means that information-heavy and classic genres tend to be rated more highly by audiences. These genres, often associated with depth, authenticity, and artistic quality, resonate more deeply with viewers, which explains their higher ratings. Next come Biography, History, War, and Animation-all with an average rating of more than 6.5. These genres

are bound to succeed since they tell great stories-either visually or factually-which plays to their positive reception. The middle tier contains genres such as Drama, Western, Romance, and Comedy, which have average ratings of about 6.0. These genres tend to be more entertainment and emotionally invested, which often results in a wide variety of viewer opinions and, as a result, varied ratings. On the low end of the spectrum, we have Thriller, Sci-Fi, Horror, and Action, with Horror receiving the lowest ratings of all. While these genres are extremely popular and often draw huge audiences, their comparatively low ratings likely reflect one or more of polarized opinions, formulaic storytelling, or an emphasis on entertainment over substance. Based on our analysis, we suggest that genres with a hard-hitting narrative, historical relevance, or artistic expression are more likely to be rated highly, while action, suspense, and horror genres receive more mixed reviews. These preferences and biases will be important to understand in order to improve genre-based recommendations on movie platforms and ensure that viewers are more likely to be offered content aligned with their tastes.

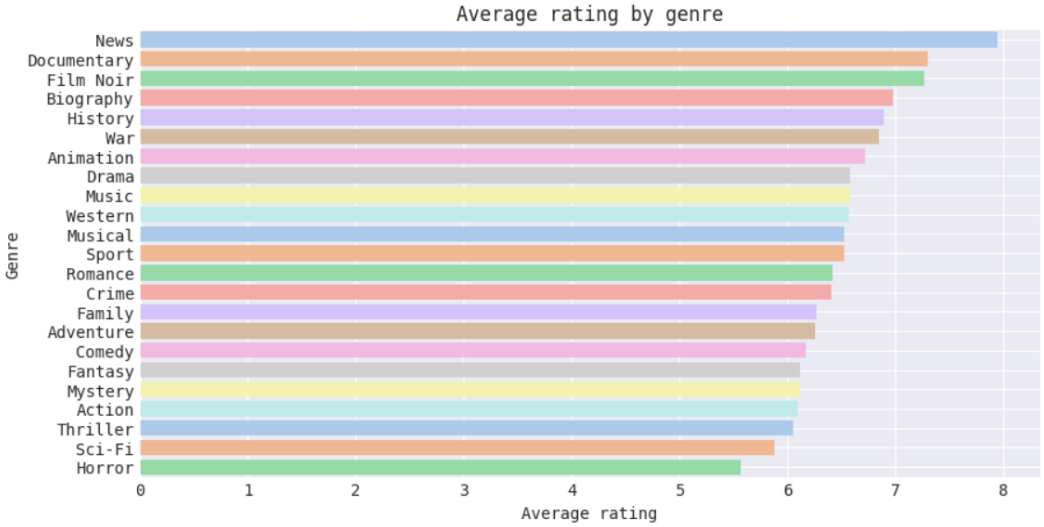


Figure 15: Average rating by genre

Figure 16, a scatter plot of Budget(USD) vs Gross Earnings(USD) that are both log scaled: From our analysis, we can say that there is a good positive correlation between budget and gross earnings-that is the higher the budget, the gross revenues will go up. At the same time, considering the scatter of points from the trend line, the returns are really all over the place at the lower and higher extremities of the budget spectrum. We observe that low-budget projects cluster around the low gross earnings, which supports the notion that small investments promise small returns. On the other hand, high-budget projects dominate the gross earnings, though they are more dispersed, which suggests that not every high-budget movie yields a proportionate level of success. One interesting pattern that we pick up is data congestion around the middle to the high budgets, probably reflecting both industry standard production costs and the typical return in the form of investment in the industry. In other words, this has much relevance for the film industry, with the preconceived relationship between budgets and financial return. Moreover, we identify several outliers where low-budget films have achieved outstanding gross earnings. These cases highlight the possibility that while a larger budget does increase the likelihood of higher revenue, success is still possible with smaller investments, though rarely. Concluding, the scatter plot is meant to show how central a driver of gross performance budget is, but it underlines that financial success in the entertainment industry is due to multiple factors, other than just the size of the budget.

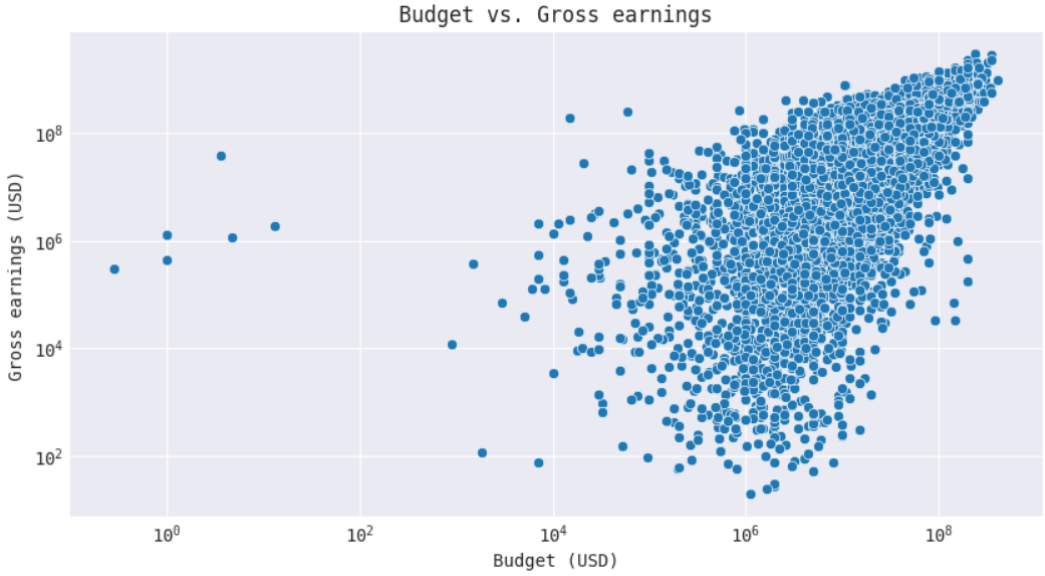


Figure 16: Budget vs. Gross earning

Figure 17 shows the trend of movie releases over the years, indicating clear growth patterns in the film industry. While examining the data, it can be seen that from the early 1900s to approximately 1960, movie releases grew at a sluggish pace, with some periods of decline but generally moving upward. Beginning in the 1960s, we see a marked increase in releases, reflecting a shift in both production capabilities and worldwide demand for cinema. This acceleration becomes even more pronounced from the 1980s onward, reflecting an increase in investment and a global expansion of the entertainment industry. But the most striking growth is post-2000, where we see a very strong boom in movie production that we can attribute to factors like technological advancement, globalization of film markets, and the availability of digital production tools, all of which have made filmmaking more accessible and cost-effective. The sharpest increase happens after 2015, when the releases of movies reached previously unparalleled levels, with more than 1,400 films in one year. This increase corresponds with the rise of digital streaming platforms, which have made entry easier for filmmakers, thus greatly increasing the volume of content being produced. We also note the sharp dip around 2020, which we can reasonably attribute to external factors like the global pandemic that temporarily halted production worldwide. In sum, the broader trend we see is exponential growth in movie releases, especially in the 21st century, driven by technological innovations and shifting viewer consumption patterns. This pattern points out the rapid changes in the landscape of the film industry, where new opportunities and challenges are influenced by modern distribution platforms and global demand.

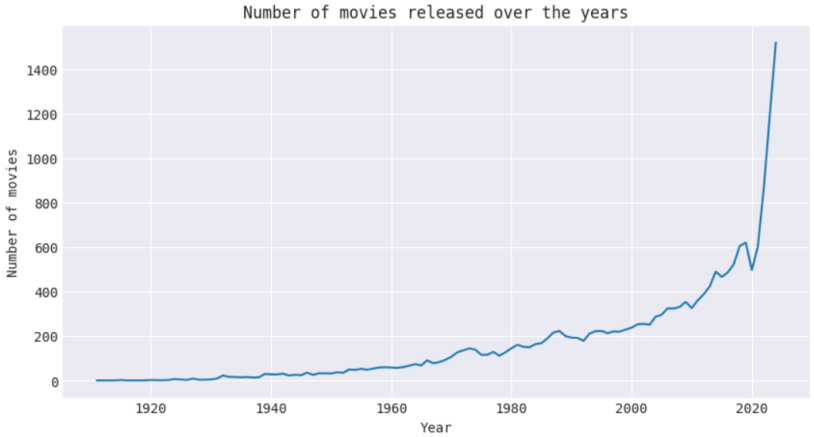


Figure 17: Number of movies released over the year

Figure 18 shows the top 10 countries by the number of movies rated 6 or higher. When we explored the dataset further, we noticed that the United States was significantly leading this pack, with more than 6,000 films receiving good ratings. This is to be expected since Hollywood alone contributes a lot to this figure, and the United States produces a large portion of the world's films. The United Kingdom follows, with close to 2,000 films, reflecting the important, though relatively smaller, cinematic output of the country compared to the U.S. We then find France and India in the next positions, which we attribute to their rich cinematic legacies and ongoing contributions to global cinema. Japan, Italy, and Canada are also ranked highly, probably because of their cultural influence and the quality of their movies that consistently receive high ratings. It further points out countries such as Germany, South Korea, and Spain among the top 10. By far, in recent years, South Korea stands out, which can be related to the recognition around the world and international success of films like Parasite, showing its rising power in global cinema. In the overall interpretation, while the dominance of the United States is crystal clear, one can't deny the strong presence of European, Asian, and other international representation, each contributing a remarkable number of critically acclaimed films. This ranking therefore provides important insights into world cinema and the increasing diversity in filmmaking across the globe.

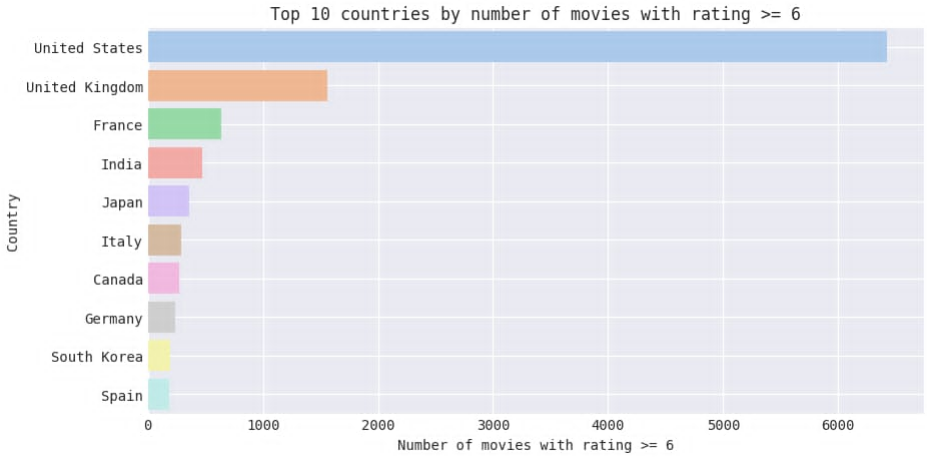


Figure 18: Top 10 countries by number of movies with rating >= 6

Figure 19 shows the top 20 countries with the highest average movie ratings for movies with over 1,000,000 ratings. From the analysis of this chart, we can see that India has the highest average rating, closely followed by Japan and New Zealand. In fact, as part of the analysis, we wanted to see trends across regions to understand where not only are movies being viewed, but well-received. We notice that countries such as Turkey and South Korea also feature in high ranks, which corresponds to their rising international recognition of film industries. From this position, it could be inferred that such regions constantly produce movies that strike a positive note with viewers from across the globe. Further, the findings bring into focus countries like Denmark, Hong Kong, and Sweden, which have become well-known for their artistic and critically acclaimed films. Moving further down the list, Brazil and Norway bring up the rear, indicating a strong global appreciation for their cinema as well. Therefore, by focusing on countries with over 1,000,000 ratings, we ensured our results would reflect movies with significant audience participation and minimize the impact of outliers. We identified regions that could be combined to represent high-quality storytelling with widespread popularity to develop the insight behind global trends in movie ratings.

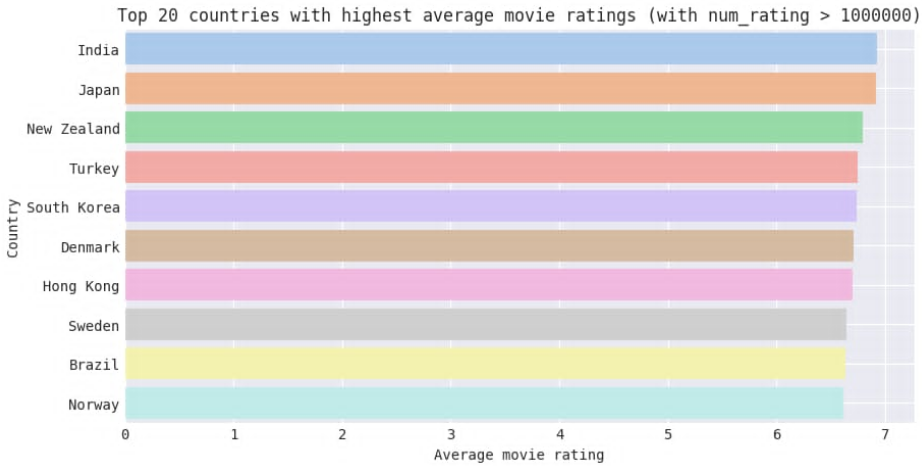


Figure 19: Top 20 countries with highest average movie ratings

Figure 20 depicts movie ratings distribution across various popularity strata, where popularity was divided into quintiles, based on the number of ratings: Very Low, Low, Medium, High, and Very High. We are thus interested in finding out how the rating count for a movie will reflect its distribution. Looking at the boxplots, we can feel that the median rating tends to increase gradually as the popularity increases. The ratings for movies with Very Low and Low popularity show a larger spread around the median with a high variation, represented by wider IQRs and multiple outliers on both extremes of the scale, 0-2 and 9-10. This indicates that less popular movies tend to receive highly polarized ratings, which may come from a niche and more opinionated audience. Moving towards the Medium, High, and Very High popularity categories, we observe that ratings become concentrated around higher values. Most notably, the Very High popularity group has a narrower IQR and a higher median close to 7.0; widely popular movies tend to enjoy more consistent favorable ratings. A reduction in extreme outliers here indicates that larger audiences stabilize the ratings-smoothening extremes and biases. Through this process, we identified that popularity is a factor affecting rating distributions, where less popular movies show greater variance and extremes while highly popular movies reflect higher consistency and stronger audience agreement. This insight will help us to understand rating behavior as a function of movie visibility and audience size.

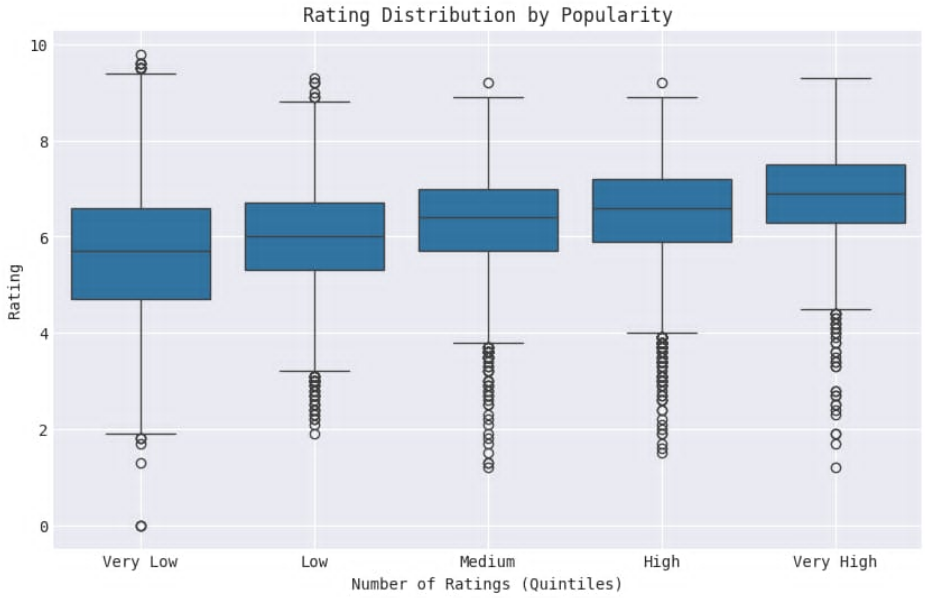


Figure 20: Rating distribution by popularity

4 Modelling

We define the problem of *collaborative filtering* (CF) in the following setting. The problem can be modeled by the random triplet (U, I, R), where

- $U = \{1, \ldots, N\}$ is the *user identifier* (N is the number of users),
- $I = \{1, \ldots, M\}$ is the *item identifier* (M is the number of items)
- R taking values from $X = \{1, 2, \ldots, 10\}$ is the rating value.

A realization of (U, I, R) denoted by (u, i, r) means that user u rated item i with value r.

In practice, the distribution of (U, I, R) is not known: we are only given a finite sample, $T' = \{(u_1, i_1, r_1), (u_2, i_2, r_2), \ldots, (u_t, i_t, r_t)\}$, generated by it. The sample T' can be used for training predictors. We assume *sampling without replacement* in the sense that (user ID, item ID) pairs are unique in the sample, which means that users do not rate items more than once.

Let $\mathcal{T} = \{(u, i) : \exists r : (u, i, r) \in T'\}$ be the set of (userID, itemID) pairs such that userID rated itemID. Note that $|T'| = |\mathcal{T}|$, and typically $|\mathcal{T}| \ll N \cdot M$, because most of the users rate only a small subset of the entire set of items.

The sample can be represented as a partially specified matrix denoted by $\boldsymbol{R} \in \mathbb{R}^{N \times M}$, where the matrix elements are known in positions $(u, i) \in \mathcal{T}$, and unknown in positions $(u, i) \notin \mathcal{T}$. The value of the matrix $\boldsymbol{R}$ at position $(u, i) \in \mathcal{T}$, denoted by r_{ui}, stores the rating of user u for item i.

4.1 User/item-based Collaborative Filtering (CF)

4.1.1 User-User CF

The User-User Collaborative Filtering (CF) algorithm works by computing similarities between users based on their rating behaviors. The steps are as follows:

1. Normalize Ratings: For each user u, compute the mean rating μ_u of user u across all rated items. Then subtract μ_u from each rating r_{ui} of user u to obtain normalized ratings $\hat{r}_{ui} = r_{ui} - \mu_u$. If a user has no ratings, set $\mu_u = 0$. From this, we get the matrix $\hat{\boldsymbol{R}}$ of normalized ratings.

2. Similarity Calculation: Construct the similarity matrix $\boldsymbol{S}$ using normalized ratings. Compute pairwise user similarities $\boldsymbol{S}_{uv}$ using a similarity function, such as cosine similarity:

$$\boldsymbol{S}_{uv} = \frac{\hat{\boldsymbol{R}}_u \cdot \hat{\boldsymbol{R}}_v}{\|\hat{\boldsymbol{R}}_u\| \|\hat{\boldsymbol{R}}_v\|}.$$

   Here, $\hat{\boldsymbol{R}}_u$ is the u-th row of $\boldsymbol{R}$.

3. Prediction: For a target user u and an item i, predict the rating using the formula:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in \mathcal{N}(u)} \text{sim}(u, v) \cdot \hat{r}_{vi}}{\sum_{v \in \mathcal{N}(u)} |\text{sim}(u, v)|},$$

   where $\mathcal{N}(u)$ represents the top k most similar users who have rated item i.

4. Incremental Update: To efficiently update the model when a new rating is added, the following steps are performed:

   (a) Map the new user and item IDs to numerical indices if they are not already present, and update the number of users N and the number of items M if needed.

(b) Update the rating matrix $\boldsymbol{R}$ with the new rating.

(c) For each user u with a new rating:

- Compute the updated mean rating μ_u:

$$\mu_u^{\text{new}} = \frac{\mu_u^{\text{old}} \cdot (n_u - 1) + r_{ui}}{n_u},$$

where n_u is the number of ratings by user u after adding r_{ui}.

- Adjust normalized ratings for all previously rated items of user u:

$$\Delta = \mu_u^{\text{new}} - \mu_u^{\text{old}}, \quad r_{uj}^{\text{new}} = r_{uj}^{\text{old}} - \Delta.$$

- Compute the new normalized rating.

(d) Recompute pairwise similarities $\boldsymbol{S}_{uv}$ involving affected users if necessary.

4.1.2 Item-Item CF

This is a dual approach of User-User CF where we change the role of user and item, which means that we predict r_{ui} based on items rated by user u.

4.2 Content-based Collaborative Filtering (CF)

The core of the content-based CF model is the similarity matrix $\boldsymbol{S}$, which quantifies the relationships between items based on their feature embeddings. For two given items i and j, their similarity is computed using a cosine similarity metric:

$$s_{ij} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{q}_j}{\|\boldsymbol{q}_i\| \|\boldsymbol{q}_j\|} \tag{1}$$

where $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ are the feature embeddings of items i and j. Using the similarity matrix, the prediction of a user's preference for an item is determined as a weighted average of the user's ratings for similar items. For a user u and an item i, the predicted rating $\hat{r}_{u,i}$ is computed as:

$$\hat{r}_{u,i} = \frac{\sum_{j \in \mathcal{T}_u} s_{i,j} \cdot r_{u,j}}{\sum_{j \in \mathcal{I}_u} s_{i,j}}, \tag{2}$$

where $\mathcal{T}_u$ denotes the set of items rated by user u, and $s_{i,j}$ is the similarity between items i and j. In experiments, the similarity matrix $\boldsymbol{S}$ was derived from item embeddings generated by advanced language models (for example: 'text-embedding-004' of Google), and we use this query to aggregate information of a movie:

```
movies_df['genres'] = movies_df['genres'].fillna('Unknown')
movies_df['director'] = movies_df['director'].fillna('Unknown')
movies_df['stars'] = movies_df['stars'].fillna('Unknown')
movies_df['duration'] = movies_df['duration'].fillna(0)  # Default
    duration as 0
movies_df['rating'] = movies_df['rating'].fillna(0.0)    # Default
    rating as 0.0
movies_df['oscar'] = movies_df['oscar'].fillna(0)        # Default
    oscar count as 0
movies_df['country'] = movies_df['country'].fillna('Unknown')
movies_df['overview'] = movies_df['overview'].fillna('No overview
    available')
```

```
movies_df['info'] = 'Genres: ' + movies_df['genres'] + '\n' \
                    + 'Director: ' + movies_df['director'] + '\n' \
                    + 'Cast: ' + movies_df['stars'] + '\n' \
                    + 'Duration: ' + movies_df['duration'].astype(str
                        ) + ' minutes \n' \
                    + 'Average rating: ' + movies_df['rating'].astype
                        (str) + '\n' \
                    + 'Number of oscars: ' + movies_df['oscar'].
                        astype(str) + '\n' \
                    + 'Country: ' + movies_df['country'] + '\n' \
                    + 'Overview: ' + movies_df['overview']
```

4.3 Matrix Factorization (MF) and variants

First, the goal of Matrix Factorization methods in CF is to estimate R from (U, I) such that the root mean squared error (RMSE) of the estimate,

$$\text{RMSE} = \sqrt{\mathbb{E}\{(\hat{R} - R)^2\}}$$

is minimal, where $\hat{R}$ is the estimate of R.

Matrix factorization is one of the most often applied techniques for CF problems. Numerous different MF variants have been already published, and we refer to the MF variants mentioned by Takács [8].

4.3.1 Incremental Simultaneous Matrix Factorization (ISMF)

The idea behind MF techniques is very simple. Suppose we want to approximate the matrix $\boldsymbol{R}$ as the product of two matrices:

$$\boldsymbol{R} \approx \boldsymbol{P}\boldsymbol{Q},$$

where $\boldsymbol{P}$ is an $N \times K$ and $\boldsymbol{Q}$ is a $K \times M$ matrix. We call $\boldsymbol{P}$ the user feature matrix, $\boldsymbol{Q}$ the item feature matrix and K is the number of features in the given factorization. Typically, $K \ll N$ and $K \ll M$, and the number of parameters to describe $\boldsymbol{R}$ can be reduced from $|\mathcal{T}|$ to $NK + KM$. Note that $\boldsymbol{Q}$ and $\boldsymbol{P}$ typically contain real numbers, even when $\boldsymbol{R}$ contains only integers.

The approximation task can be defined as follows. Let p_{uk} denote the elements of $\boldsymbol{P} \in \mathbb{R}^{N \times K}$, and q_{ki} the elements of $\boldsymbol{Q} \in \mathbb{R}^{K \times M}$. Further, let p_u denote a row (vector) of $\boldsymbol{P}$, and q_i a column (vector) of $\boldsymbol{Q}$. Then:

$$\hat{r}_{ui} = \sum_{k=1}^{K} \boldsymbol{p}_{uk}\boldsymbol{q}_{ki} = \boldsymbol{p}_u\boldsymbol{q}_i \tag{3}$$

$$e_{ui} = r_{ui} - \hat{r}_{ui} \quad \text{for } (u, i) \in \mathcal{T} \tag{4}$$

$$SSE = \sum_{(u,i)\in\mathcal{T}} e_{ui}^2 = \sum_{(u,i)\in\mathcal{T}} (r_{ui} - \boldsymbol{p}_u\boldsymbol{q}_i)^2 \tag{5}$$

$$RMSE = \sqrt{\frac{SSE}{|\mathcal{T}|}} \tag{6}$$

Let

$$(\boldsymbol{P}^*, \boldsymbol{Q}^*) = \arg\min_{\boldsymbol{P},\boldsymbol{Q}} RMSE = \arg\min_{\boldsymbol{P},\boldsymbol{Q}} SSE.$$

In order to minimize RMSE, which in this case is equivalent to minimizing SSE, we apply a simple incremental gradient descent method to find a local minimum of SSE, where one gradient step intends to decrease the square of the prediction error of a single rating, which is e_{ui}^2. For the incremental gradient descent method, suppose that we are at the $(u, i)-$th training example, r_{ui} and its approximation $\hat{r}_{ui}$ is given. We compute the gradient of e_{ui}^2:

$$\frac{\partial}{\partial p_{uk}} e_{ui}^2 = -2e_{ui} \cdot q_{ki}$$
$$\frac{\partial}{\partial q_{ki}} e_{ui}^2 = -2e_{ui} \cdot p_{uk}$$

We update the weights in the direction opposite to the gradient:

$$p'_{uk} = p_{uk} + \eta \cdot e_{ui} \cdot q_{ki}$$
$$q'_{ki} = q_{ki} + \eta \cdot e_{ui} \cdot p_{uk}$$

That is, we change the weights in $\boldsymbol{P}$ and $\boldsymbol{Q}$ to decrease the square of actual error, thus better approximating r_{ui}. Here, η is the learning rate. This basic MF method is referred to as ISMF, that is, incremental simultaneous MF, due to its distinct incremental and simultaneous weight updating method in comparison to other MF methods. When training has been completed, each value of R can be easily calculated using equation (3), even at unknown positions. In other words, the model $(\boldsymbol{P}^*, \boldsymbol{Q}^*)$ provides a description of how an arbitrary user would rate any item. For new ratings, in MF methods, we just need to update the weights using gradient descent for that.

4.3.2 Regularized Incremental Simultaneous Matrix Factorization (RISMF)

The matrix factorization presented in the previous section can overfit if η and the extent of the change in $\boldsymbol{Q}$ are both small for users with few (no more than K) ratings: assuming that the feature vectors of the items rated by the user are linearly independent and $\boldsymbol{Q}$ does not change, there exists a user feature vector with zero training error. A common way to avoid overfitting is to apply L2 regularization to penalize the weights, which results in a new optimization problem:

$$e'_{ui} = e_{ui}^2 + \lambda \cdot \boldsymbol{p}_u \cdot \boldsymbol{p}_u^T + \lambda \cdot \boldsymbol{q}_i^T \cdot \boldsymbol{q}_i$$
$$SSE' = \sum_{(u,i) \in \mathcal{T}} e'_{ui}$$
$$(\boldsymbol{P}^*, \boldsymbol{Q}^*) = \arg\min_{(\boldsymbol{P}, \boldsymbol{Q})} SSE'$$

Here $\lambda \geq 0$ is the regularization factor. We call this MF variant RISMF that stands for regularized incremental simultaneous MF. We compute the gradient of e'_{ui}:

$$\frac{\partial}{\partial p_{uk}} e'_{ui} = 2 \times (-e_{ui} \cdot q_{ki} + \lambda \cdot p_{uk})$$
$$\frac{\partial}{\partial q_{ki}} e'_{ui} = 2 \times (-e_{ui} \cdot p_{uk} + \lambda \cdot q_{ki})$$

We update the weights in the direction opposite to the gradient:

$$p'_{uk} = p_{uk} + \eta \cdot (e_{ui} \cdot q_{ki} - \lambda \cdot p_{uk}),$$
$$q'_{ki} = q_{ki} + \eta \cdot (e_{ui} \cdot p_{uk} - \lambda \cdot q_{ki}).$$

For the training algorithm used in RISMF, see Algorithm 1. Note that the matrices are initialized randomly by $\mathcal{N}(0, \frac{1}{K})$, because if both $\boldsymbol{P}$ and $\boldsymbol{Q}$ are initialized with a constant value, that is, both are rank 1, the weight update will not increase the rank, which is equivalent to the case $K = 1$.

In [7], Takács et al. observed that the learning curve of epochs (RMSE on the validation set as a function of the number of epochs) is usually convex, regardless of λ, that is why we use early stopping.

Algorithm 1 Training algorithm for RISMF

Input: $\mathcal{T}'$: training set, η: learning rate, λ: regularization factor
Output: $\boldsymbol{P}^*, \boldsymbol{Q}^*$: the user and item feature matrices
Partition $\mathcal{T}'$ into two sets: ${\mathcal{T}'}_{\text{I}}$, ${\mathcal{T}'}_{\text{II}}$ (validation set)
Initialize $\boldsymbol{P}$ and $\boldsymbol{Q}$ with $\mathcal{N}(0, \frac{1}{K})$.
while the algorithm isn't terminated **do**
  for each (u, i, r_{ui}) element of ${\mathcal{T}'}_{\text{I}}$ **do**
    compute e'_{ui};
    compute the gradient of e'_{ui};
    update $\boldsymbol{p}_u$, the u-th row of $\boldsymbol{P}$, and $\boldsymbol{q}_i$, the i-th column of $\boldsymbol{Q}$;
  end for
  calculate the RMSE on ${\mathcal{T}'}_{\text{II}}$;
  if the RMSE on ${\mathcal{T}'}_{\text{II}}$ was better than in any previous epoch **then**
    Let $\boldsymbol{P}^* = \boldsymbol{P}$ and $\boldsymbol{Q}^* = \boldsymbol{Q}$.
  end if
  terminal condition: RMSE on ${\mathcal{T}'}_{\text{II}}$ does not decrease during two epochs.
end while

4.3.3 Biased Regularized Incremental Simultaneous Matrix Factorization (BRISMF)

Some users tend to rate all items higher or lower than the average. The same may hold for items: some items can be very popular. Although MF can reconstruct the original matrix exactly when K is large enough and $\lambda = 0$, this is not the case when overfitting is to be avoided.

There is a straightforward way to extend RISMF to be able to directly model this phenomenon, by extending MF with biases for users and items. In [6], Paterek et al. mentioned the idea of bias feature by incorporating constant values in the MF, based on that, we incorporate bias features into RISMF by fixing the first column of P and the second row of Q to the constant value of 1: $\boldsymbol{p}_1 = \mathbf{1}$ and $\boldsymbol{q}_2 = \mathbf{1}$ even when updating.

We refer to this method as BRISMF which stands for biased regularized incremental simultaneous MF. This simple extension speeds up the training phase and may yield a more accurate model with better generalization performance.

4.3.4 Incorporating Contextual Information into Matrix Factorization

Neighbor based (NB) approaches exploit the observation that similar users rate similar items similarly. The answer of the predictor is then obtained by combining the ratings of similar users (items) for the active item (user), which we called the user neighbor-based and item neighbor-based approach, respectively.

MF and NB approaches complement each other well:

- The MF approach views the data from a high level perspective. MF can identify the major structural patterns in the ratings matrix. An appealing property of MF is that it is able to detect the similarity between 2 items, even if no user rated both of them.
- The NB approach is more localized. It is typically good at modeling pairs of users/items and not so good at modeling interdependency within larger sets of users/items. NB methods are memory based, therefore they do not require any training.

It is known that the combination of MF and NB can lead to very accurate predictions [3]. However, the price of additional accuracy is paid by the decreased scalability. The idea is that we try to improve an existing MF model $(\boldsymbol{P}, \boldsymbol{Q})$ by adding a neighbor-based correction term to its answer in the prediction phase. Assuming the item neighbor based-scheme, the corrected answer for query

(u, i) is the following:

$$\hat{r}_{ui} = \boldsymbol{p}_u \boldsymbol{q}_i + \gamma \frac{\sum_{j \in \mathcal{T}_u \setminus \{i\}} s_{ij} (\boldsymbol{p}_u \boldsymbol{q}_j - r_{uj})}{\sum_{j \in \mathcal{T}_u \setminus \{i\}} s_{ij}},$$

where s_{ij} is the similarity between $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$, and $\mathcal{T}_u$ is set of the items rated by user u, and γ is the correction term. This model can be seen as a unification of the MF and NB approaches.

The similarity s_{ij} can be defined in many different ways:

- 1. Normalized scalar product based similarity.

$$s_{ij} = \left(\frac{\max\{0, \sum_{k=1}^{K} q_{ki} q_{kj}\}}{\sqrt{\sum_{k=1}^{K} q_{ki}^2} \cdot \sqrt{\sum_{k=1}^{K} q_{kj}^2}} \right)^{\alpha}$$

  where α is an amplification parameter.

- 2. Normalized Euclidean distance based similarity.

$$s_{ij} = \left(\frac{\sum_{k=1}^{K} (q_{ki} - q_{kj})^2}{\sqrt{\sum_{k=1}^{K} q_{ki}^2} \cdot \sqrt{\sum_{k=1}^{K} q_{kj}^2}} \right)^{-\alpha}$$

  In the first two cases, the value s_{jk} can be calculated in O(K) time, thus $\hat{r}_{ui}$ can be calculated in O($K \cdot |\mathcal{T}_u|$).

- 3. If we have more information of items, like in movies_details.csv, we can find the similarities of those items based on their features. A query that can be used to find the similarity can be:

```
movies_df["content"] = movies_df.apply(
    lambda x: f"{x['title']} is a {x['genres']} movie
        directed by {x['director']} starring {x['stars']}.
        Overview: {x['overview']}",
    axis=1
)
```

  We can use this query for each item, and then calculate the similarities of these queries based on the cosine similarity of TF-IDF vectors, or the vectors generated by Transformer's Embedding.

This model can be seen as simple, scalable, and accurate unification of the MF and NB approaches. The training is identical to the regular MF training. The prediction consists of an MF and a NB term.

4.3.5 Retraining

For other approaches, we can think about the fact that the incremental gradient descent weight update of RISMF has a serious drawback: item features change while we iterate through users. If the change is significant, user features updated at the beginning of an epoch may be inappropriate at the end of the epoch. We can solve this by completely recompute the user features after the learning procedure. This method can serve as an efficient incremental training method for recommender systems.

5 Experiments

After our explanation about data visualization and analysis and the theoretical properties of learning models we chose to predict the target variable, this section will describe how we applied theories with previous sections into practice. Starting from the training strategy in Subsection 5.1, where we describe how to prepare the data and which techniques we tried to improve our results. After that, we chose suitable evaluation metrics to evaluate the performances of this classification problem. Then we will describe how we trained and assessed our models. Finally, in 5.3 and 5.4, we will present our results in quantitative matters. Final outcome of our project is website demonstration using ReactJS and Flask.

5.1 Main workflow

Our classification workflow is broken down into the following steps [2]:

1. Data Collection
2. Data Cleaning and Preprocessing
3. Exploratory data analysis
4. Modeling
5. Evaluation
6. Deployment

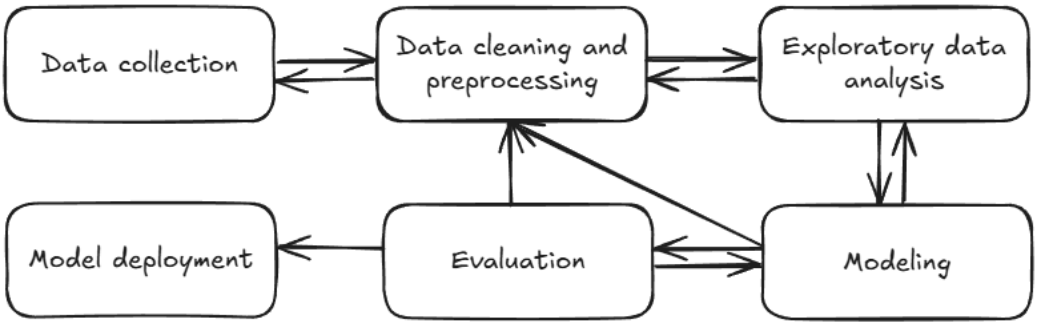


Figure 21: Main Workflow

5.2 Evaluation metrics

5.2.1 Hit Ratio (HR)

Hit Ratio (HR) is a simple and intuitive metric that evaluates whether the target item (e.g., a ground truth item in a recommendation task) appears in the top-H recommendations provided by a model. It serves as a binary metric for success - either the target item is present within the top-H list (a "hit"), or it is not. The HR metric is particularly useful for measuring the recall capability of a recommendation system. For a single user u, the hit ratio at H of a target item is defined as:

$$HR_u@H = \begin{cases} 1, & \text{if the target item is in the top-}H\text{ recommendations} \\ 0, & \text{otherwise.} \end{cases}$$

For a set of n users, the overall HR is the average across all users:

$$HR@H = \frac{1}{n}\sum_{u=1}^{n} HR_u@H.$$

We can see that, the overall HR values range from 0 to 1, with higher values indicating better performance, but does not account for the ranking of the target item within the top-K; as long as it is present, it counts as a "hit."

5.2.2 Normalized Discounted Cumulative Gain (NDCG)

Normalized Discounted Cumulative Gain (NDCG) is a ranking-based metric that evaluates both the presence of the target item in the recommendation list and the quality of its rank. It assigns higher scores when the target item appears closer to the top of the ranked list, reflecting the intuition that users are more likely to interact with items ranked higher. For a single user u, the Discounted Cumulative Gain (DCG) at H of a target item is defined as:

$$DCG@H = \sum_{i=1}^{K} \frac{\text{rel}_i}{\log_2(i+1)},$$

where rel_i is the relevance of the item at rank i. For binary relevance (common in recommendation tasks), $\text{rel}_i = 1$ if the item at rank i is the target item, and $\text{rel}_i = 0$ otherwise.

The Ideal DCG (IDCG) is the maximum possible DCG@H, calculated by placing the target item in the highest possible rank:

$$IDCG@H = \sum_{i=1}^{\min(K,1)} \frac{1}{\log_2(i+1)}.$$

Finally, the NDCG for a single user u is:

$$NDCG_u@H = \frac{DCG@H}{IDCG@H}.$$

For a set of n users, the overall NDCG is the average across all users:

$$NDCG@H = \frac{1}{n}\sum_{u=1}^{n} NDCG_u@H.$$

Unlike HR, NDCG takes the position of the target item into account, rewarding higher ranks more heavily. Note that, in the protocol, each time we only consider one target item, so $NDCG@H = DCG@H$.

5.2.3 Evaluation Method

To evaluate the performance of item recommendation, we use the one-plus-random methodology.

1. Data Splitting: Divide the dataset into three parts: training, validation, and test sets such that for every user, the ratio of ratings in each set is 3:1:1, respectively.

2. Candidate Generation: For each user, we focus on high ratings, which are ratings ≥ 8. So in the evaluation, we only use high-rated items in the test set to get the final results.

3. Model Evaluation: Since it is too time-consuming to rank all items for every user during evaluation, we followed the common strategy [5] that randomly samples 99 unrated items and ranks the test item among these 99 items. The performance of a ranked list is judged by the Hit Ratio at H ($HR@H$) and Normalized Discounted Cumulative Gain at H ($NDCG@H$). Here we set $H = 5$. As such, the HR intuitively measures whether the test item is present on the top-H list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at the top ranks. We calculated both metrics for each test user and reported the average score.

5.3 Results and Explanation

Here, we only show the result of the best model in each method. Because RISMF is ISMF when $\lambda = 0$, so we only show the result of the best RISMF model.

Method	HR@5	NDCG@5
User-User CF	**0.6683**	**0.4809**
Item-Item CF	0.0539	0.0290
Content-based Recommendation	0.0804	0.0584
RISMF ($K = 300, \lambda = 0.1, \texttt{learning\_rate} = 0.006$)	0.1228	0.0766
BRISMF ($K = 200, \lambda = 0.1, \texttt{learning\_rate} = 0.004$)	0.1100	0.0675
RISMF (with retraining)	0.1192	0.0754
RISMF (with item neighbor-based)	0.2395	0.1784

Firstly, it is evident that user-user CF outperforms well-tuned matrix factorization CF methods because our rating matrix is extremely sparse (the average number of ratings per user is less than 2). In sparse datasets, cold-start users or items, those with very few ratings, result in poorly trained embeddings due to insufficient data for generalizing latent factors. In contrast, CF methods based on user-user or item-item similarity exhibit greater robustness to sparsity because they compute similarities directly from observed ratings rather than relying on latent factor inference. For example, user-user CF recommends items that have been highly rated by users with similar rating patterns, maintaining effectiveness even with limited interactions.

Another contributing factor is the significant popularity bias present in our dataset, where a small number of users account for the majority of the ratings. User-user CF explicitly captures this bias, as users who rate a lot of products have high similarity with many other users. Conversely, matrix factorization models may struggle to fully leverage such patterns because their embeddings tend to oversmooth the data, treating all items uniformly. This bias also explains why user-user CF's performance is superior to item-item CF's one, a phenomenon that is clearly illustrated in Figure 22:

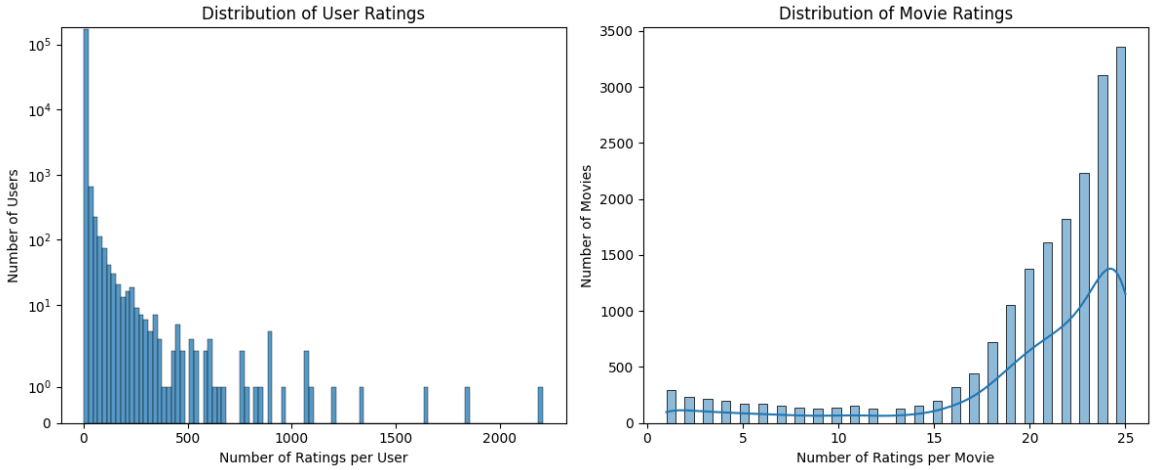


Figure 22: The distribution of user and item ratings

It can be observed that while many users have more than 50 ratings, and in some cases, even more than 500 ratings, a movie has at most 25 ratings. This discrepancy contributes to the increased computational time required for the user-user CF model, however, this greater data density for users results in improved model performance.

5.4 Web implementation

We implemented the web deployment of our project using ReactJS for the frontend and Flask for the backend to demonstrate the movie recommendation system with multiple features. Figure 23 illustrates an example of our application recommending movies based on the content (title, overview, director, etc.) of the movie "***Your Name***".

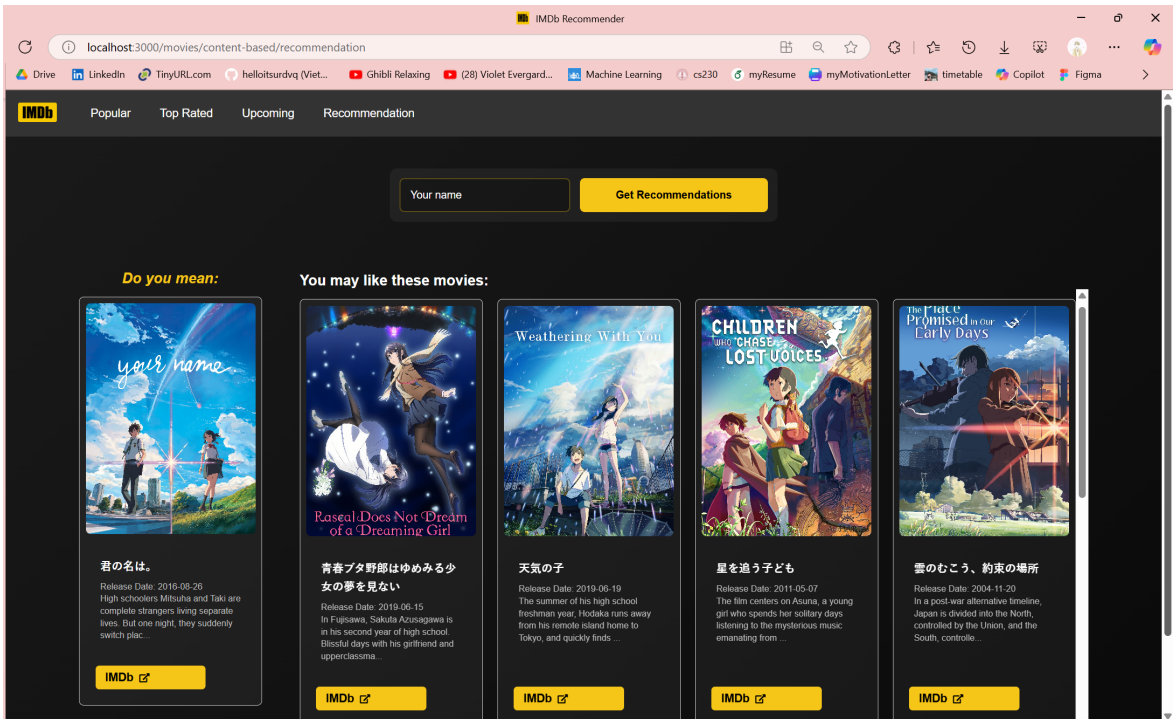


Figure 23: Example of content-based recommendation of the movie "***Your Name***".

5.4.1 Frontend

Here are the main route paths for the client-side:

1. `/` - Home page
2. `/movie/:id` - Individual movie detail page
3. `/movies/:type` - Movie listings by type (popular/top_rated/upcoming)
4. `/movies/recommendation` - Main recommendation page
5. `/movies/content-based/recommendation` - Content-based recommendation page
6. `/movies/collaborative-filtering/recommendation` - Collaborative filtering recommendation page

5.4.2 External and Backend API

1. TMDb API for movie data
2. IMDb API for additional movie information
3. `POST /api/content-based-recommend`
   - Input: `movie_title`
   - Returns: recommended movies and title
4. `POST /api/collaborative-filtering-recommend`
   - Input: `user_id`
   - Returns: personalized movie recommendations

To begin using our user interface, please review and follow the provided instructions below carefully:

First, clone the project from this github link: `https://github.com/helloitsurdvq/DsProject_Imdb` and then change the working directory to the project root, change the path of the project properly and then install required packages and libraries.

```
pip install -r requirements.txt
```

Application demo using commands for both client and server side:

```
# client
cd DsProject_Imdb/demo/client
npm i
npm start
```

```
# server
cd DsProject_Imdb/demo/server
python -m app
```

The website can be accessed at:

- Client: `http://localhost:3000`
- Server: `http://localhost:5000`

6 Conclusion and Possible Extension

So far, we have introduced and worked through our data science project. Starting with a dataset containing detailed information about movies and user ratings for each movie, we applied several data validation and preprocessing techniques to prepare a usable dataset for exploratory data analysis and modeling. The primary goal is to recommend movies that matches the users' preferences. Finally, we developed a user-friendly website interface to enhance accessibility and provide a smooth and intuitive experience for users to interact with our movie recommendation system.

One of the most challenging aspects of solving our problem was determining the appropriate training method for our dataset, as well as selecting the models and techniques for workload assignments among team members. Ultimately, we opted for the aforementioned techniques and models discussed in earlier sections. And finally, we decided to follow an aforementioned process.

As we can see from the table of results of different models, the model that shows the best recommending performance, in general, is the User-user Collaborative Filtering model which performed the most effectively with both movies and user rating dataset. Therefore, given more time on this project, we would have found better training strategy for each models and further explored new AI training models and analysis. Additionally, we would have delved into the application of self-supervised, semi-supervised learning or deep learning methods on our preprocessed dataset.

All the codes, notebooks, trained weights model files for the project with detailed instructions can be found in the submited zip file.

7 Member Assignments

- Nguyen Chi Long - 20210553: Modeling, theory.
- Ngo Xuan Bach - 20215181: Data preprocessing, evaluation.
- Le Xuan Hieu - 20215201: Modeling, evaluation.
- Dinh Viet Quang - 20215235: Data Collection, EDA, model deployment.
- Nguyen Viet Thuan - 20210826: EDA, evaluation.

References

[1] Press Room - IMDb. https://www.imdb.com/pressroom/stats/.

[2] Hust.edu.vn, 2023.

[3] Robert M Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 43–52. IEEE, 2007.

[4] Encyclopaedia Britannica. IMDb.

[5] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[6] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[7] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Major components of the gravity recommendation system. *Acm Sigkdd Explorations Newsletter*, 9(2):80–83, 2007.

[8] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656, 2009.