Building and Deploying a Movie Recommendation System

Technical Report

Farid Karimi
Department of Computer Science
Shahid Beheshti University
Tehran, Iran
`fkarimi8320@gmail.com`

Abstract

This report presents a hybrid movie recommendation system developed with data from The Movie Database (TMDb) and MovieLens. The system combines content-based filtering with collaborative methods to deliver personalized suggestions. Content similarity is computed with TF-IDF and cosine similarity, while predicted user ratings from a Singular Value Decomposition model are used to rerank results. A popularity-based model serves as a baseline, and the process includes data preparation and feature engineering to refine movie representations. The system is deployed as an interactive web application on Hugging Face Spaces, addressing challenges such as dependency management and memory limits. The outcome is a practical tool that demonstrates the application of modern recommendation techniques.

1 Introduction

In the current era of digital streaming, audiences have access to an overwhelming volume of cinematic content. Navigating this vast landscape to find movies that align with individual tastes has become a significant challenge. Recommendation systems are a critical tool in addressing this issue, helping to filter content and enhance the user experience. This report outlines the construction of a movie recommendation system designed to provide relevant and personalized suggestions.

The primary objective of this project is to build a hybrid recommender that integrates the strengths of two major approaches: content-based filtering and collaborative filtering. By analyzing the intrinsic properties of movies such as genre, cast, and keywords and leveraging the collective wisdom of user ratings, the system aims to offer recommendations that are both thematically appropriate and tailored to individual user preferences.

This paper documents the entire development process, beginning with the exploration and preprocessing of data from The Movie Database (TMDb) and the MovieLens dataset. It then details the methodology for feature engineering and the implementation of a popularity baseline, a content-based model, and a collaborative filtering model using Singular Value Decomposition (SVD). Finally, it describes the architecture of the hybrid system and discusses the practical challenges and solutions involved in deploying the model as an interactive web application.

2 Data Exploration and Preprocessing

The foundation of this recommendation system rests upon several datasets obtained from The Movie Database (TMDb) and GroupLens, each requiring careful preparation to be useful for modeling. The initial phase of the project involved a thorough exploratory data analysis to understand the structure and content of each file, followed by a series of cleaning and transformation steps to create a cohesive and refined dataset. This process included handling missing values, parsing complex string formats, and merging information from different sources.

2.1 Credits, Keywords, and Movies Metadata

The primary dataset, `movies_metadata.csv`, contains extensive information about each film, including budget, revenue, release date, and genre. Several columns that were not pertinent to the recommendation task, such as homepage URLs and taglines, were removed to simplify the dataset. The `credits.csv` file provided detailed cast and crew information, while the `keywords.csv` file contained thematic keywords associated with each movie. A snapshot of the `keywords.csv` file is shown below:

Table 1: Sample entries from the `keywords.csv` dataset.

id_keywords	name_keywords
862	jealousy, toy, boy, friendship, friends, rival...
8844	board game, disappearance, based on children's...
15602	fishing, best friend, duringcreditsstinger, ol...
31357	based on novel, interracial relationship, sing...
11862	baby, midlife crisis, confidence, aging, daugh...

A significant part of the preprocessing work involved parsing columns that stored data in a stringified JSON-like format. For instance, the `genres`, `cast`, and `crew` columns in their raw state were not directly usable. Custom functions were developed to safely evaluate these strings and extract the relevant information, such as actor or genre names, into clean, comma-separated lists. For the movies metadata, features like budget and revenue were converted to numeric types, and release dates were parsed into datetime objects to facilitate temporal analysis.

After cleaning each file individually, they were merged into a single comprehensive dataframe using the unique movie ID as the common key. This unified dataset combines the essential metadata, cast, crew, and keywords for each film.

2.2 Ratings and Links Data

The `ratings_small.csv` file from the MovieLens dataset contains user ratings for various movies. This file is central to the collaborative filtering component of the system, providing a direct measure of user preference. An initial exploration showed that the ratings are distributed on a scale from 0.5 to 5.0, with most scores clustering around the higher end of the scale. The accompanying `links.csv` file provided the necessary identifiers to map the movies in the MovieLens dataset to their corresponding entries in the TMDb dataset, ensuring consistency across all data sources.

The final step of the preprocessing phase was to save the cleaned and merged dataframes into new CSV files. This provides a stable and tidy foundation for the subsequent feature engineering and model building stages.

2.3 Data Visualization and Insights

Visualizing the processed data offered several insights into the nature of the film industry and audience preferences. A histogram of movie release dates revealed a significant increase in film production in recent decades, illustrating the growing volume of content.

The distribution of movie runtimes showed a concentration around the 90-minute mark, which is a standard length for feature films.

Bar charts highlighting the most frequent genres, production companies, and production countries provided a clear picture of the dominant players and themes in the cinematic landscape. Genres like Drama and Comedy appeared most frequently, while production was heavily concentrated in the United States.

Finally, analysis of the cast and crew data identified the most prolific actors and creative roles. Unsurprisingly, well-known actors appeared in a large number of films, and key roles such as Director and Producer were the most common jobs listed in the crew dataset. A breakdown of gender distribution among cast and crew also highlighted representational skews within the industry.

3 Methodology

The recommendation system is built upon two primary components: a baseline model that ranks movies by a quality and popularity metric, and a content-based model that uses textual features to identify similar items. The development process involved extensive feature engineering to create a unified representation of each movie's content, followed by the implementation of the recommendation algorithms.

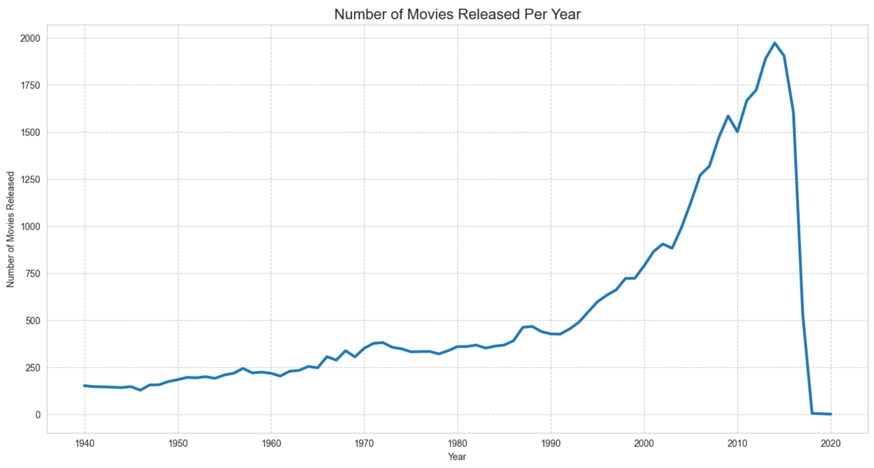

Figure 1: Distribution of Movie Releases Over Time.

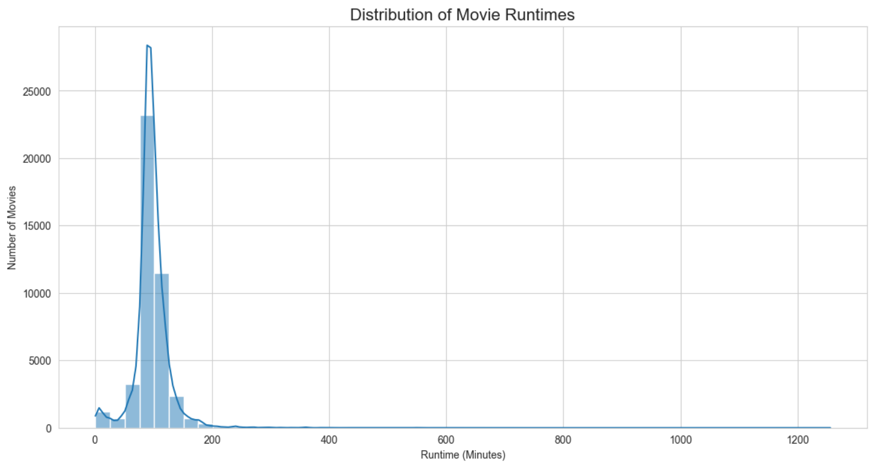

Figure 2: Distribution of Movie Runtimes.

3.1 Feature Engineering for Content-Based Filtering

To create a meaningful representation of each movie for the content-based recommender, several textual features were combined into a single descriptive string. This process began with extracting the director's name from the crew data. A function was designed to parse the crew information, identify the individual credited as 'Director', and isolate their name.

Next, the keywords associated with each film were refined. To reduce noise and focus on more significant terms, only keywords that appeared in the dataset more than once were retained. These common keywords were then stemmed to their root form to consolidate variations of the same word. For example, terms like "jealousy" and "jealous" would be reduced to a common stem.

The final step was to aggregate the processed features. The top five cast members, the main genres, the director's name, and the stemmed keywords were all converted to lowercase, with spaces removed to form consistent tokens. These tokens were then concatenated into a single string for each movie, creating a rich, composite feature that encapsulates its core thematic and creative elements. This combined feature serves as the input for the content-based model.

3.2 Popularity Baseline Recommender

Before developing the content-based system, a baseline recommender was established to provide a benchmark for performance and a fallback for situations with limited data. This model ranks movies using the IMDb weighted rating

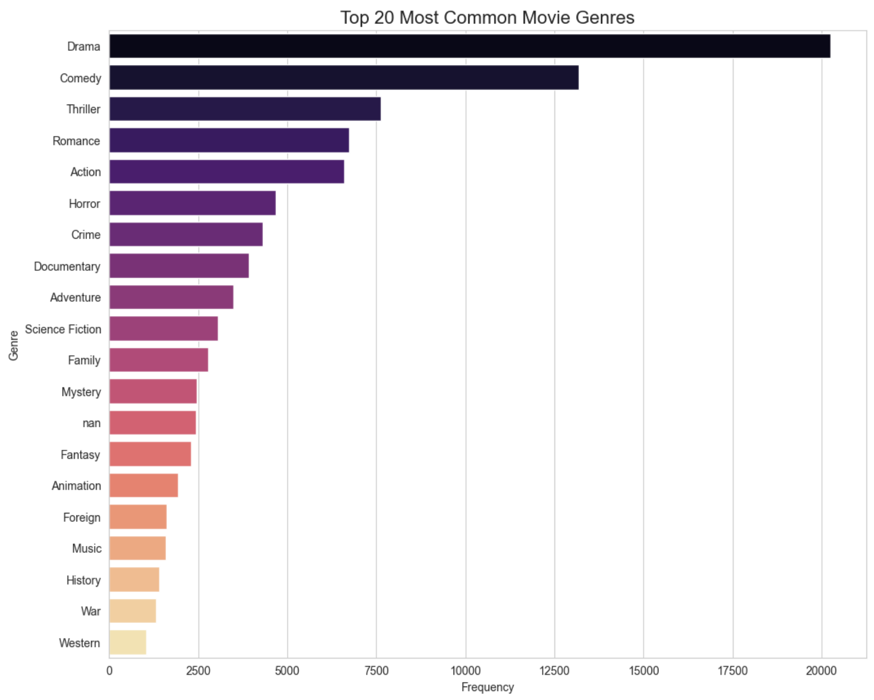

Figure 3: Top 20 Most Frequent Movie Genres.

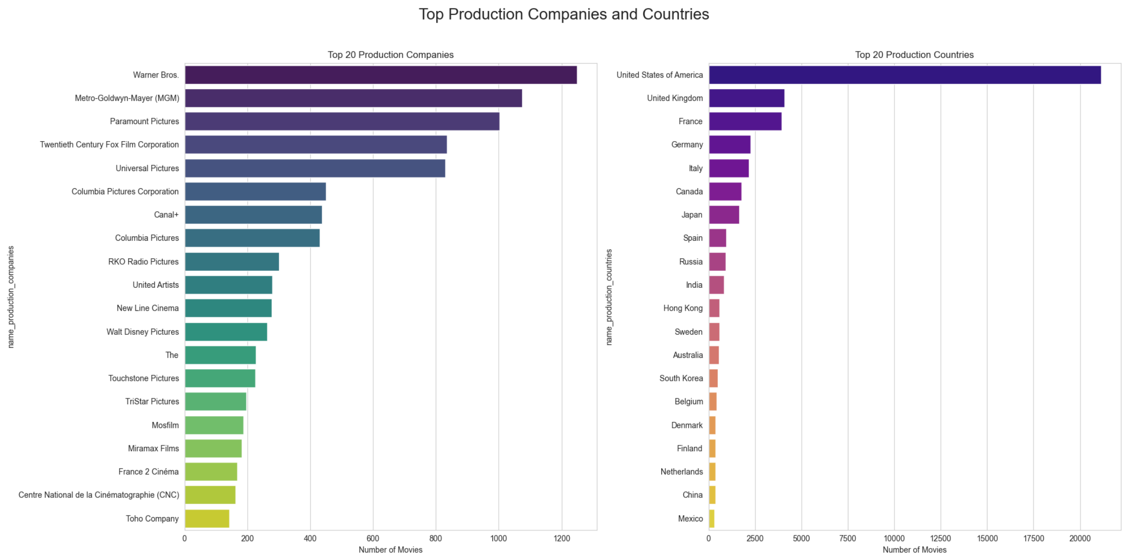

Figure 4: Top 20 Production Companies and Countries.

formula (Equation 1), which moderates a movie's average rating by considering the number of votes it has received. This approach prevents movies with a few high ratings from unfairly outranking those with a more established consensus.

The weighted rating WR for each movie is calculated using IMDb's formula:

$$WR = \frac{v}{v+m} \cdot R + \frac{m}{v+m} \cdot C \tag{1}$$

where R is the movie's average rating, v is the number of votes, m is the minimum vote threshold (90th percentile in our dataset), and C is the mean rating across all movies.

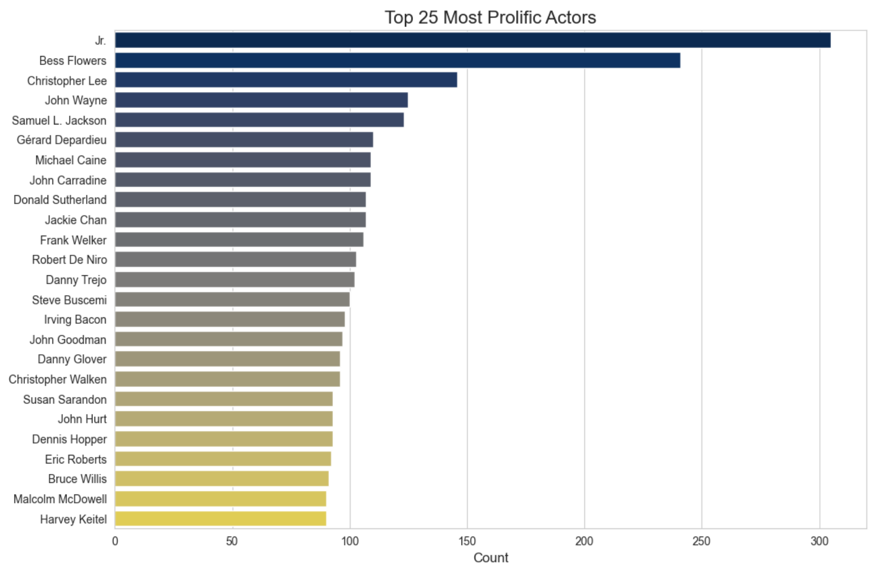

Figure 5: Top 25 Most Prolific Actors.

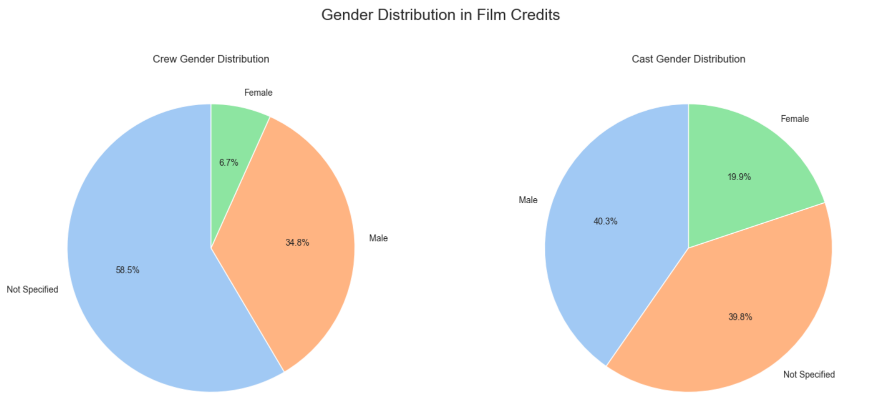

Figure 6: Gender Distribution in Film Credits.

The formula requires calculating the mean rating across all movies and setting a minimum vote threshold, which was determined by the 90th percentile of vote counts in the dataset. Movies that do not meet this threshold are not considered for the top-ranked lists. This method provides a simple yet effective way to recommend movies that are both popular and well-regarded by a broad audience.

3.3 Content-Based Recommender

The core of the content-based recommender is a system that calculates the similarity between movies based on their textual features. The combined feature string, engineered as described previously, was transformed into a numerical representation using a Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer. This technique creates a matrix where each row corresponds to a movie and each column represents a unique word or term. The value in each cell reflects the importance of a term in a specific movie's description relative to the entire collection of movies.

With this TF-IDF matrix, the cosine similarity was calculated between each pair of movies. Cosine similarity measures the angle between two vectors, providing a score from 0 to 1 that indicates how similar their content is. A score closer to 1 signifies a high degree of similarity in terms of shared keywords, genres, cast, and crew.

To generate recommendations for a given movie, the system first identifies its corresponding vector in the TF-IDF matrix. It then computes the cosine similarity between this vector and all other movie vectors. The resulting similarity

Word Cloud of Movie Keywords
woman director
independent film
based on novel
biography
murder
duringcreditsstinger
female nudity
violence
sport
musical
suspense
world war ii
stand-up comedy
sequel
nudity
sex
police
revenge
friendship
love
teenager
film noir
high school
serial killer
martial arts
dystopia
kidnapping
drug
prison
silent film
family
paris
monster
death
gore
zombie
blood
gangster
marriage
suicide
new york city
london england
small town
alien
detective
christmas
short
prostitute
new york
gay
rape
aftercreditsstinger
robbery
vampire
romance

Figure 7: A Word Cloud Generated from the Most Frequent Movie Keywords.

scores are combined with a quality metric, which is a hybrid score blending the movie's popularity and its weighted rating. This final score is used to rank the movies, and the top results, excluding the input movie itself, are presented as the recommendations. This hybrid approach ensures that the recommended movies are not only thematically similar but also of high quality.

3.4 Collaborative Filtering

To complement the content-based approach, a collaborative filtering model was developed to generate recommendations based on user behavior. This model uses the ratings data to identify patterns in user preferences. The core of this component is a matrix factorization technique, specifically Singular Value Decomposition (SVD), implemented using the Surprise library. SVD works by decomposing the user-item rating matrix into lower-dimensional matrices that represent latent factors for users and items. These factors capture underlying characteristics that drive rating patterns, such as a user's affinity for a particular genre or an item's appeal to a certain demographic.

Table 2: SVD Model Performance Across 5-Fold Cross-Validation

Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE	0.8936	0.8933	0.8999	0.8942	0.8982	0.8959	0.0027
MAE	0.6874	0.6872	0.6919	0.6891	0.6934	0.6898	0.0025
Fit time (s)	0.59	0.53	0.50	0.51	0.52	0.53	0.03
Test time (s)	0.11	0.06	0.06	0.11	0.06	0.08	0.03

The model was trained on the MovieLens ratings dataset, which contains user-provided scores for a wide range of movies. Before training, the performance of the SVD algorithm was evaluated using 5-fold cross-validation. This process involves splitting the data into five subsets, training the model on four of them, and testing it on the remaining one, rotating through each subset as the test set. The evaluation metrics used were Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), which measure the accuracy of the model's rating predictions. The cross-validation results, shown in Table 2, indicated a consistent and acceptable level of prediction error, justifying the use of SVD for this task. After evaluation, the final model was trained on the entire ratings dataset to prepare it for making predictions.

3.5 Hybrid Recommendation Model

The final recommendation system combines the strengths of both the content-based and collaborative filtering models to create a more robust and personalized experience. This hybrid approach aims to provide recommendations that are not only thematically relevant to a user's interests but also aligned with the preferences of similar users.

The process for generating hybrid recommendations begins by taking a user's highly-rated movies as input. For each of these movies, the content-based model is used to generate a list of the top 20 most similar films based on their textual features. This initial set of candidates is then passed to the collaborative filtering model. The SVD model predicts a rating for each of these similar movies specifically for the given user.

The final list of recommendations is then ranked based on these predicted ratings. This two-stage, retrieve-then-rerank process ensures that the recommendations are diverse and personalized. The content-based component retrieves a broad set of relevant items, and the collaborative filtering component refines this list by prioritizing items that the user is most likely to enjoy. This method effectively mitigates some of the common problems in recommendation systems, such as the cold-start problem for new items, while also incorporating the user's personal rating history to tailor the results.

4 Model Deployment

To make the recommendation system accessible and interactive, a web application was developed and deployed on Hugging Face Spaces. This process involved building a user interface, managing dependencies, and optimizing the model for a cloud environment. Several practical challenges were encountered and addressed during this phase.

4.1 Application Development and Frameworks

The initial plan was to create the application using Streamlit, a popular framework for building data science web applications in Python. A functional version of the application was developed with this tool. However, deployment to the Hugging Face platform using Docker revealed persistent issues with installing the correct dependencies.

To overcome these obstacles, the application was rebuilt using Gradio, an alternative framework that is also well-suited for creating machine learning demos. The Gradio version proved to be more stable within the Hugging Face environment, and it allowed for a clean and straightforward user interface where users could either find recommendations based on a selected movie or receive personalized suggestions by entering their user ID.

4.2 Deployment Challenges and Solutions

Deploying the recommender system presented several technical hurdles that required careful consideration. One of the first issues was dependency management. The project required a Python version older than 3.13 and a NumPy version below 2.0 to ensure compatibility with all the libraries used especially surprise. These version constraints were specified in the deployment configuration to create a stable runtime environment.

A more significant challenge was the size of the pre-computed cosine similarity matrix. At approximately 5 GB, this file was too large for the free tier of storage on Hugging Face. Loading such a large file into memory would also be inefficient. The solution was to modify the recommendation logic to compute similarities dynamically. Instead of pre-calculating and storing the entire matrix, the application calculates the cosine similarity for a single movie against all others at the moment a recommendation is requested. This approach significantly reduces the application's storage and memory footprint.

This on-the-fly calculation also resolved a memory error that occurred even in local development when attempting to compute the full similarity matrix at once. By processing one movie at a time, the application avoids the large memory allocation that was causing the system to fail, making the hybrid approach feasible in a resource-constrained environment. The final deployed application, available on Hugging Face Spaces, successfully integrates the content-based and collaborative filtering models into an efficient and interactive tool.

5 Conclusion and Ethical Considerations

This project successfully developed and deployed a hybrid movie recommendation system that combines content-based and collaborative filtering techniques. The journey from raw data to a functional web application involved rigorous data preprocessing, thoughtful feature engineering, and the systematic implementation of multiple recommendation models. The final system is capable of generating personalized suggestions by first identifying movies with similar content and then ranking them according to a user's predicted ratings. The deployment to Hugging Face Spaces demonstrated the practical challenges of transitioning a model from a development environment to a live application, including dependency conflicts and the need for significant memory and storage optimization.

Despite the functional success of the system, it is important to consider the ethical implications inherent in its design. Recommender systems can inadvertently create filter bubbles, where users are only exposed to content similar to what they have already seen, limiting their discovery of diverse perspectives. This system, by its nature, is susceptible to **popularity bias**, as the baseline model and hybrid scoring favor movies that are already widely seen and highly rated. This can lead to a reinforcing loop where popular movies become even more dominant, while lesser-known or independent films are marginalized.

Furthermore, the data may contain **representation skews**. As the exploratory analysis revealed, film production is heavily concentrated in certain countries, and the cast and crew data show significant gender imbalances. A recommendation system trained on such data is likely to perpetuate these skews, recommending content that reflects the dominant industry trends rather than a more equitable representation of global cinema.

To mitigate these issues, a future iteration of this system could implement **diversity-aware re-ranking**. After generating an initial list of recommendations, this technique would adjust the rankings to ensure a broader mix of genres, production countries, and lesser-known films. By introducing novelty and serendipity, such a mechanism could help to counteract popularity bias and expose users to a wider range of content, fostering a more inclusive and enriching viewing experience.

5.1 Data Attribution

This project utilizes data from The Movie Database (TMDb) but is not endorsed or certified by TMDb. The user ratings data is provided by the MovieLens dataset, curated by the GroupLens research group at the University of Minnesota. We gratefully acknowledge their contribution to the research community.

References

[1] F. Maxwell Harper and Joseph A. Konstan. The MovieLens Datasets: History and Context. In *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4), 19:1–19:19, 2015.

[2] Nicolas Hug. Surprise: A Python library for recommender systems. In *Journal of Open Source Software*, 5(52), 2174, 2020.

[3] Michael J. Pazzani and Daniel Billsus. Content-Based Recommendation Systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 325–341, 2007.

[4] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[5] The Movie Database (TMDb) API. Website. `https://www.themoviedb.org/documentation/api`. Accessed: 2024.

[6] Rounak Banik. The Movies Dataset. Kaggle, 2017. `https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset`.

[7] Streamlit Inc. Streamlit: The fastest way to build and share data apps. Website. `https://streamlit.io`. Accessed: 2024.