Big Data Analytics

Final Project Report

Developing a Movie Recommender System Using Apache Spark

Group Number:

8

Group Members:

Janelle Bauske

Sandra Chiwike

Nandita Ghildyal

1. Introduction

Recommender systems are vital in industries like e-commerce, social media, and streaming, helping users navigate vast information. Movie recommender systems personalize viewing experiences by analyzing user interactions, demographic data, and item attributes such as genres, actors, and directors. Despite their utility, they face challenges like data sparsity and cold-start problems, which limit their ability to provide accurate and diverse recommendations, especially for new users or items (Harper & Konstan, 2015).

To address these challenges, two main approaches are used: collaborative filtering (CF) and content-based filtering (CB). CF leverages user-item interaction patterns for personalized recommendations but struggles with sparse data. CB relies on item attributes like genres and plot summaries, working well without user interaction data but often generating repetitive suggestions that hinder novelty and exploration (Kumar Singh, 2020). While both methods are effective, they are insufficient when used independently in complex movie recommendation scenarios.

The primary goal of this project is to build a hybrid recommender system that combines collaborative filtering with metadata-driven content-based recommendations. This approach leverages the strengths of both methods to enhance personalization, accuracy, and diversity while addressing data sparsity and cold-start problems. By effectively capturing user behavior through collaborative filtering and utilizing rich metadata for content-based insights, the system ensures relevant and novel recommendations, even for users with minimal interaction history. The model also promotes exploration by introducing lesser-known movies, fostering a more engaging and tailored user experience.

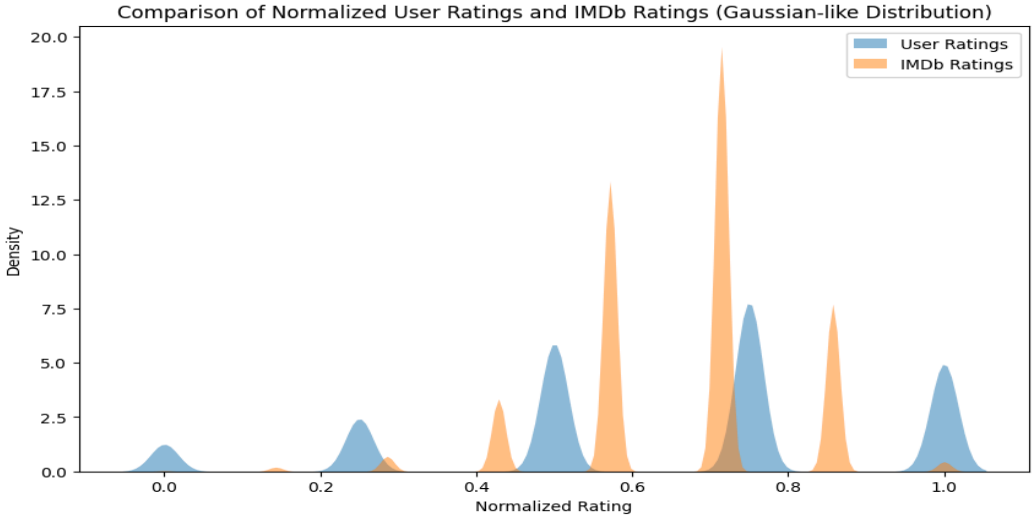
2. Data

The MovieLens User Review and MovieLens Tag Genome datasets served as the foundation for our data integration and enrichment process. To implement a hybrid recommender system we combined user ratings with detailed movie metadata pulled from the IMDb API. The original dataset included essential information such as user demographics, genres, timestamps, average ratings, and movie tags. For additional movie information, we extracted each unique IMDb ID from the MovieLens dataset to create pull requests for the IMDb API. This pulled data included cast members, localized titles, runtimes, IMDb ratings, plot outlines, and director names. The process was automated using parallel processing with a thread pool. The retrieved data was then converted to a Spark DataFrame with a defined schema for seamless integration with the MovieLens dataset.

The datasets were then merged through an inner join on the IMDb ID field, creating a unified dataset of 916,355 rows and 20 columns (The methods for the MovieLens data merge as well as the IMDb API data pull are included in our zipped directory's Code Appendix, as they were very computationally taxing).

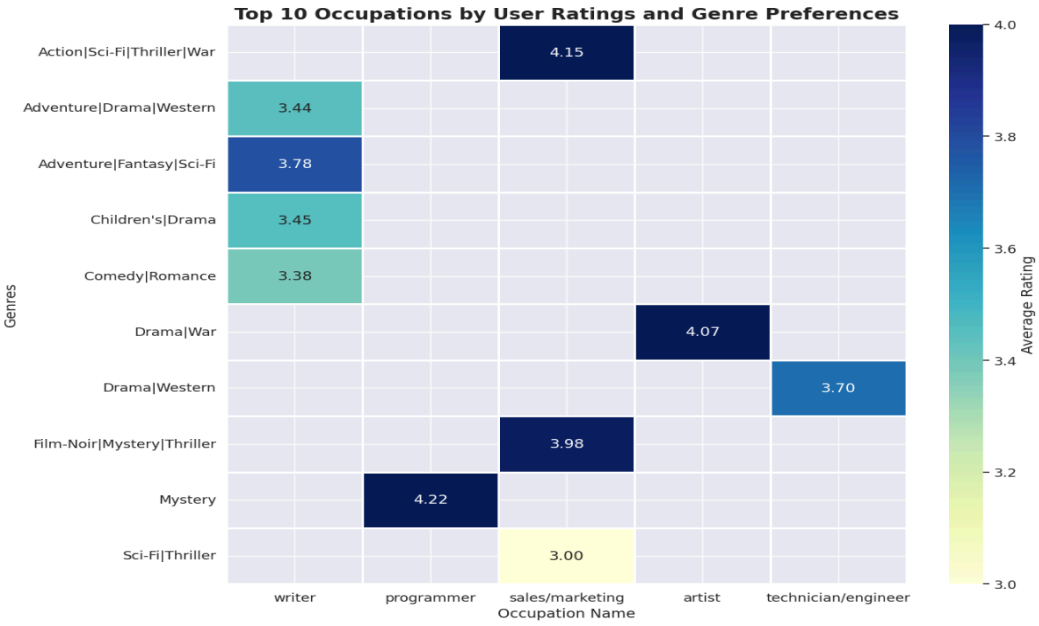
The figure below compares user and IMDb ratings. IMDb ratings (orange) are tightly clustered around 0.8, indicating consistency, while user ratings (blue) show more variability, peaking near 0.9. To align the scales, user ratings (originally out of 5) were normalized to match IMDb ratings (out of 10). Both distributions show smaller peaks at lower values (0.2–0.4), reflecting minority

ratings. The differences suggest that users tend to rate more generously than IMDb. Despite this, the two distributions are similar, indicating that the user ratings sample is not skewed.



2.1 Normal Distribution for User vs IMDb Ratings

The heatmap matrix below presents the average user ratings for different genre combinations across several occupations. Each occupation is associated with specific genres, and the values represent the average rating users have given to movies in these genres.

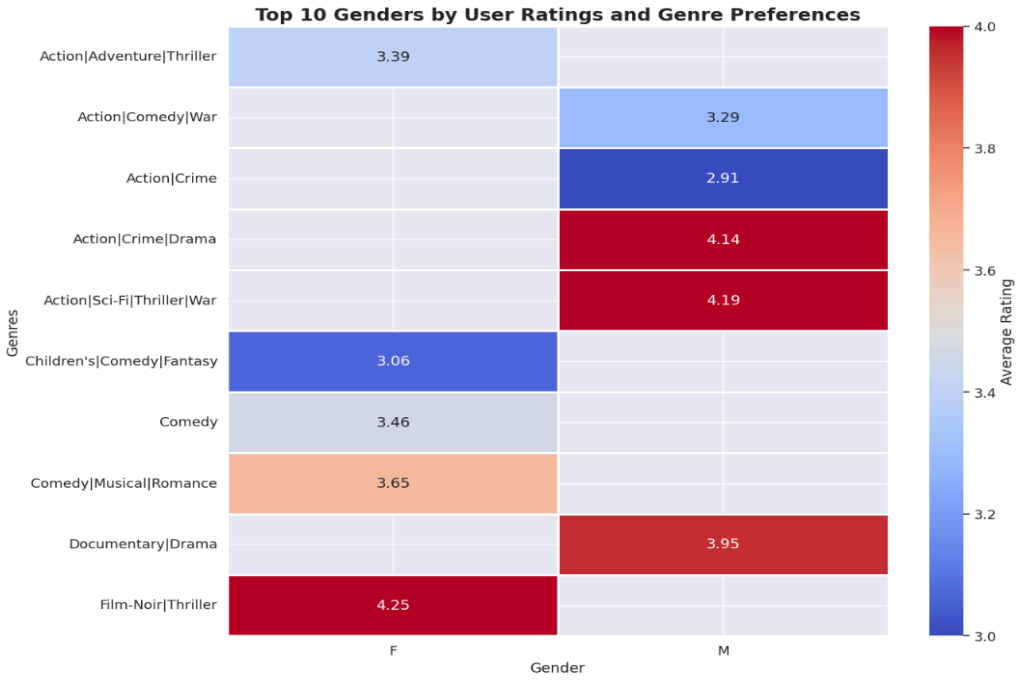


2.3 Occupation-Based Ratings for Popular Genres

The ratings reveal distinct genre preferences across occupations. Writers show a strong liking for Action|Sci-Fi|Thriller, while Programmers prefer Mystery. Sales and Marketing users rate

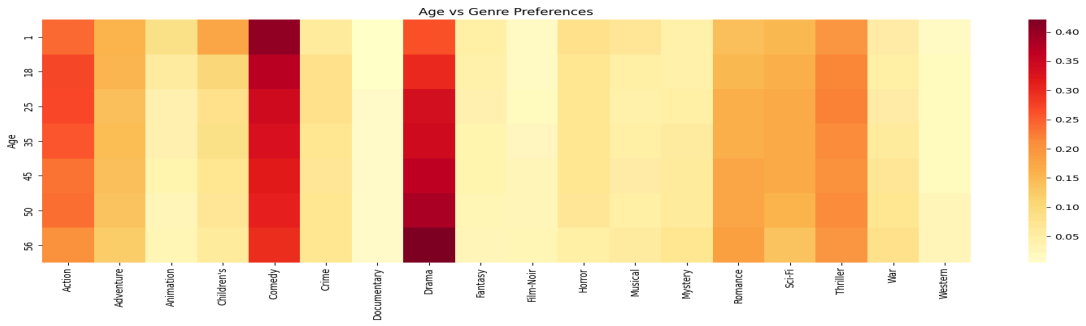
Action|Sci-Fi|Thriller highly, and Artists favor Drama| genre more. These insights highlight how occupation-specific tastes can inform more personalized film recommendations.

The graph below presents the average user ratings across various gender and genre combinations. For female users, ratings are available for genres such as Action, Comedy, Film Noir, indicating a broad spread of preferences across different types of content.



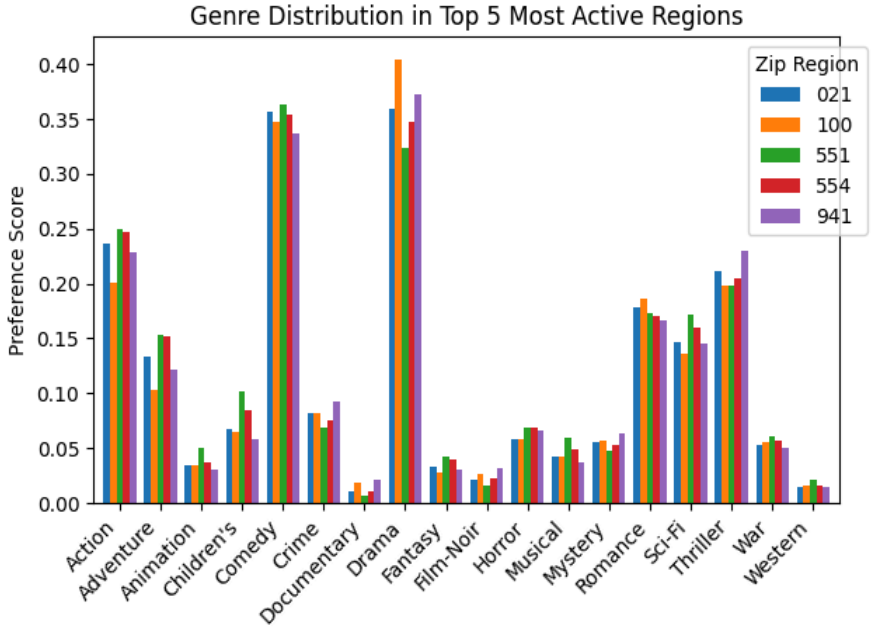
2.4 Gender-Occupation Based Ratings

Male users show ratings for genres with Action and War, but there are also gaps in ratings for genres like "Children's|Comedy|Fantasy" and others. These missing values highlight areas with sparse user engagement. Overall, this data provides insights into gender-specific preferences across genres and showcases the need for targeted recommendation systems that account for varying levels of engagement in different genres.



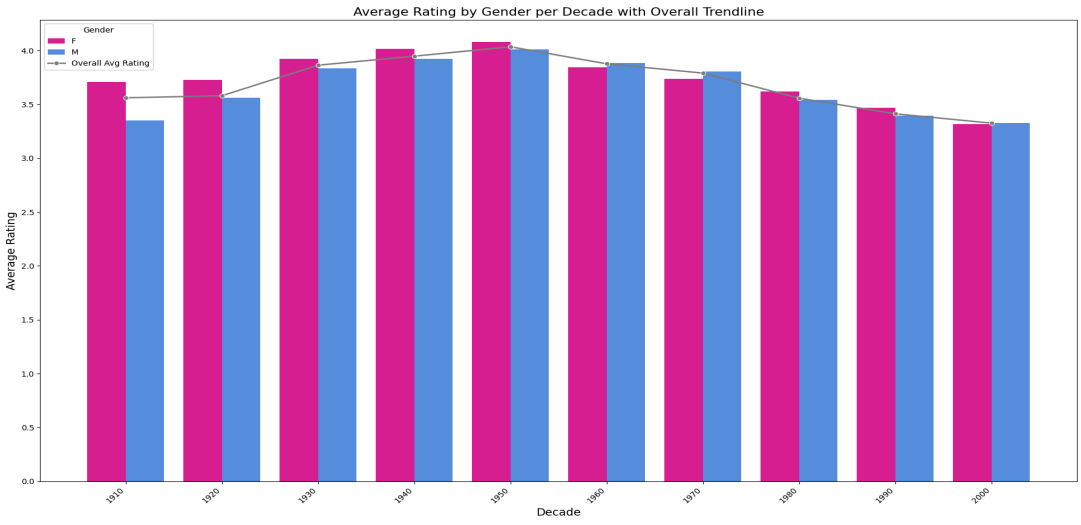
2.5 Age Based Genre Preferences

The heatmap above displays top genre preferences by age and the graph below shows the top genre preferences of the top 5 most active zip code regions in our dataset. Active, in this case, is defined as regions with the most users creating the most movie ratings. Both graphs show a general trend of preference towards Drama, Comedy, and Action movies.



2.6 Genre Distribution across Zip Code Regions

Graph below showcases an analysis of rating trends between genders that provide insight into general movie popularity as well. We can see a peak in average rating for movies from the 1950s in both genders. However, there is a spike in average ratings from female users (F) for movies from the 1910s. Both genders seem to rate newer movies lower on average, however, this could be due to a higher volume of ratings being produced for modern films.



2.7 Rating Trends by Gender by Movie Release Decade

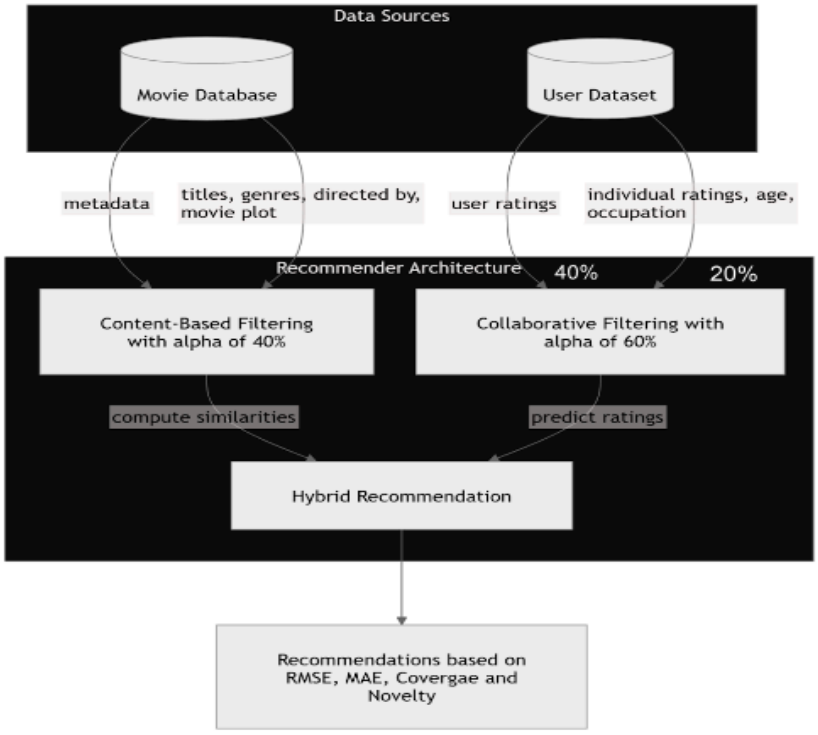
3. Methods

In our implementation of a movie recommender system, we developed and tested over 16 models through the preliminary, intermediate and final stages of the project. Each took a different approach to combining collaborative and content-based filtering methods.

The initial hybrid recommendation models combine two popular approaches: **collaborative filtering** and **content-based filtering**, aiming to improve the accuracy of movie recommendations. In the **collaborative filtering** step, the model uses **Alternating Least Squares (ALS)**, a matrix factorization technique.

ALS identifies latent factors from user-item interactions, predicting a user's rating for an unseen movie based on their past behavior and the behavior of similar users. A training set is used to build the ALS model, which is then tested on a separate set to evaluate its predictions. The **content-based filtering** step uses movie **genres** as features to recommend items based on their attributes. First, genres are tokenized and transformed into **TF-IDF vectors** to quantify the significance of each genre. Additionally, the model computes a **content-based score** by averaging movie ratings, which gives an indication of how well a movie might appeal to a user based on its genre.

Finally, the model combines the results of both methods using a weighted approach—60% from collaborative filtering and 40% from content-based filtering. While this hybrid model improves recommendations, it has some misgivings. The weighting scheme is arbitrary and was not optimal. Finally, it did not address cold-start problems effectively, where new users or movies lack sufficient data for accurate recommendations.



3.1 Architecture for the Final Model

This final recommendation model combines feature engineering and collaborative filtering using ALS (Alternating Least Squares). The process begins by converting the Pandas DataFrame into a PySpark DataFrame. We then apply **StringIndexer** to transform categorical features, such as **Gender** and **Occupation**, into numerical representations. These indexed columns are assembled into a feature vector using **VectorAssembler**, which prepares the data for model training. For collaborative filtering, the **ALS algorithm** is employed to predict user ratings based on the interactions between users and items. The ALS model learns latent factors for both users and items, capturing hidden relationships between them.

The model is trained on user-item interaction data (UserID, MovieID, UserRating), with validation and testing essential to evaluate its performance. A train-test split was used to optimize the model's parameters, including tuning the alpha parameter in ALS, which controls regularization. The weighting system was adjusted to create a more balanced hybrid approach: 40% from collaborative filtering, 40% from content-based scores, and 20% from IMDb ratings.

When predicting for a new user, the model's alpha is adjusted to give more weight to content-based filtering and item features, such as genres, directors, or actors, which are more reliable when user-item interaction data is sparse. This adjustment ensures that recommendations are relevant even before the new user has rated or interacted with many items. As the user interacts with the system, the influence of collaborative filtering gradually increases, while content-based features continue to dominate early recommendations.

4. Results

We used multiple metrics to assess prediction accuracy and recommendation quality of our model. Two such metrics were Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), to measure prediction accuracy. This was done by calculating differences between predicted and actual ratings. RMSE squares these differences before averaging, making it more sensitive to large errors, while MAE provides a simpler average of absolute differences. These metrics matter because accurate rating predictions form the foundation of reliable recommendations.

On our 5-star rating scale, our initial model achieved RMSE of 0.90 and MAE of 0.72, while the final model showed values of 0.83 and 0.46. The improvements in RMSE and MAE indicate a significant boost in prediction accuracy for the recommendation system. The RMSE decreased from 0.90 to 0.83, signaling a reduction in the overall error magnitude, making predictions closer to actual ratings. The MAE improved more notably, dropping from 0.711 to 0.46, suggesting that the model's predictions are, on average, just 0.46 stars off, which is a considerable improvement in precision. Catalog coverage measures what percentage of available items get recommended to users. It helps evaluate whether a system is too focused on popular items or effectively uses its full catalog. High coverage indicates the system can serve diverse user interests and aid content discovery.

Our system's coverage of 94.11% means it actively recommends about 471,200 different titles from our dataset of 500,000 movies, helping users discover lesser-known films matching their interests.

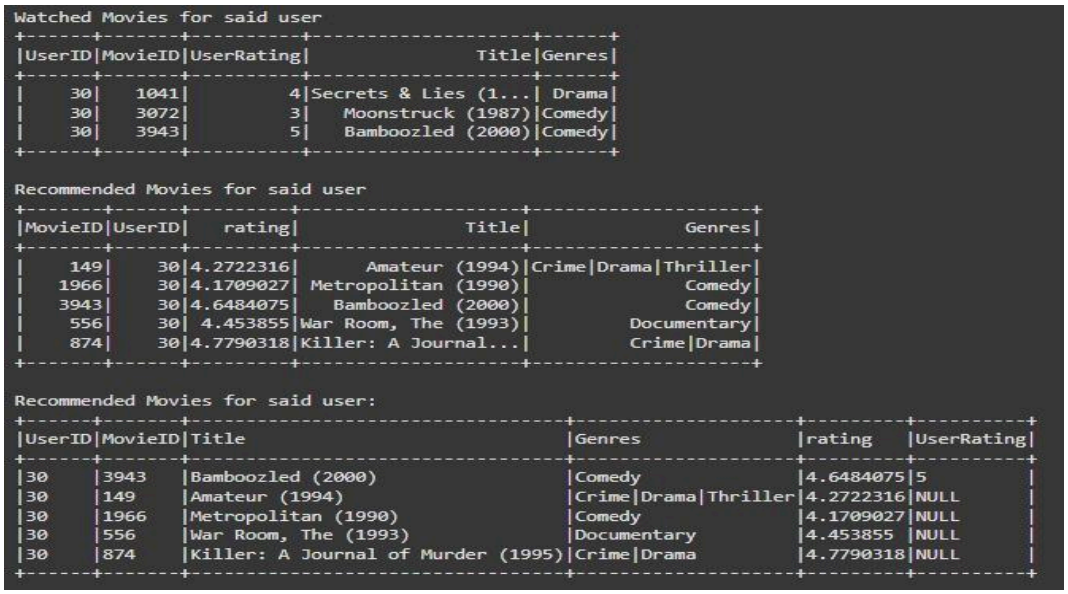
However, there was a decrease in coverage in the final model, which achieved a coverage of 89.55%. This reduction suggests a shift towards more popular recommendations but may limit the

exposure of niche content. Despite this, the model still covers a large portion of the catalog, ensuring users receive a wide range of relevant suggestions.

Hit Rate@K tracks whether users find at least one relevant item in their top K recommendations. This metric directly reflects user satisfaction since users typically focus on their first few recommendations. A high hit rate suggests users quickly find content they enjoy, building trust in the system. Our Hit Rate@10 for the final model reached 0.999, meaning 99.9% of users receive at least one movie they would rate 4 stars or higher in their top 10 recommendations.

Precision and Recall work together to evaluate recommendation relevance. (Deutchman, 2023). These metrics help balance recommending safe choices versus introducing users to new content . For the final model, precision@10 of 0.850 means about 8-9 out of every 10 recommended movies would receive high ratings (4+ stars). The Recall@10 of 0.766 shows we identify and recommend over two-thirds of all movies a user would potentially enjoy.

Shown below is an example for a new user, illustrating how the system adapts its recommendations, particularly in the early stages, when content-based features dominate before collaborative filtering takes over.



Watched Movies for said user

UserID	MovieID	UserRating	Title	Genres
30	1041	4	Secrets & Lies (1...	Drama
30	3072	3	Moonstruck (1987)	Comedy
30	3943	5	Bamboozled (2000)	Comedy

Recommended Movies for said user

MovieID	UserID	rating	Title	Genres
149	30	4.2722316	Amateur (1994)	Crime|Drama|Thriller
1966	30	4.1709027	Metropolitan (1990)	Comedy
3943	30	4.6484075	Bamboozled (2000)	Comedy
556	30	4.453855	War Room, The (1993)	Documentary
874	30	4.7790318	Killer: A Journal...	Crime|Drama

Recommended Movies for said user:

UserID	MovieID	Title	Genres	rating	UserRating
30	3943	Bamboozled (2000)	Comedy	4.6484075	5
30	149	Amateur (1994)	Crime|Drama|Thriller	4.2722316	NULL
30	1966	Metropolitan (1990)	Comedy	4.1709027	NULL
30	556	War Room, The (1993)	Documentary	4.453855	NULL
30	874	Killer: A Journal of Murder (1995)	Crime|Drama	4.7790318	NULL

4.1 Results for a New User from Test Data

For a new user in the test set, the recommendation system adapts to the sparse interaction data by relying on content-based filtering and hybrid approaches to suggest relevant movies. In this case, the new user has rated a few movies: "Secrets & Lies" (1996), "Moonstruck" (1987), and "Bamboozled" (2000).

Since the user has limited movie ratings, there’s insufficient data for collaborative filtering, which typically relies on user-item interaction. As a result, the system emphasizes content-based filtering, which recommends movies based on attributes such as genres, directors, and cast.

For instance, the system suggests movies like "Amateur" (1994), "Metropolitan" (1990), "War Room" (1993), and "Killer: A Journal of Murder" (1995), all of which share genres or themes with the movies the new user has rated highly, such as Drama and Comedy. This ensures that the recommendations are aligned with the user's apparent interests, even without prior interaction data.

Moreover, the system's hybrid approach allows the model to adjust its recommendations as the user interacts more with the platform. Initially, content-based filtering dominates, but as the user rates more movies, the system begins to introduce collaborative filtering, using user preferences and ratings data to refine its suggestions. This way, the new user is gradually exposed to more personalized recommendations based on both content similarity and community behavior, ensuring relevant suggestions from the outset and adapting as their interaction history grows.

5. Challenges

In implementing our movie recommender system, we encountered two challenges that influenced our data processing capabilities and model performance. They were mainly around computational limitations in pulling and processing big data. The first challenge emerged from processing plotline text data. Due to the extensive nature of movie plot descriptions, we had to reduce our dataset from 1,000,000 to 500,000 rows to manage computational resources effectively. This reduction decreased the diversity of movies available for recommendations and limited the number of user interactions we could analyze. With fewer user-movie interactions, our model's ability to identify and match user preferences to movies became more constrained.

The second major challenge involved issues with the IMDb API integration. The API's instability made data collection difficult and unreliable, particularly for recent movies. This created a significant gap in our dataset, as newer releases were missing from our recommendation pool. This limitation affected our system's ability to recommend current films, reducing its relevance for users interested in newer content.

6. Conclusion

In conclusion, our movie recommender system successfully improved recommendation accuracy and adaptability through the development and testing of over 16 models. The hybrid approach, combining collaborative filtering with content-based methods, played a key role in enhancing performance. By using ALS for collaborative filtering and genre-based features for content-based filtering, the system was able to deliver relevant recommendations to users.

The final model showed significant improvements, with a reduction in RMSE and MAE, indicating better prediction accuracy. It also demonstrated high catalog coverage, recommending a wide variety of movies, although there were slight reductions in coverage for niche content. Metrics such as Hit Rate, Precision (0.850), and Recall (0.766) reflected the system's effectiveness in providing relevant recommendations, achieving a good balance between safe and novel suggestions. The system adapted well to new users by prioritizing content-based filtering when interaction data was sparse, gradually incorporating collaborative filtering as more interactions occurred.

Despite challenges with computational limitations in processing large datasets and issues with the IMDb API affecting data availability, the system proved effective at providing relevant, novel, and personalized recommendations for both new and existing users.

References

1. Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets. *ACM Transactions on Interactive Intelligent Systems*, *5*(4), 1–19. https://doi.org/10.1145/2827872
2. Kumar Singh, Pradeep, et al. "(PDF) Collaborative Filtering in Recommender Systems: Technicalities, Challenges, Applications, and Research Trends." *ResearchGate*, July 2020, www.researchgate.net/publication/340828169_Collaborative_Filtering_in_Recommender_Systems_Technicalities_Challenges_Applications_and_Research_Trend
3. Deutschman, Zuzanna. "Recommender Systems: Machine Learning Metrics and Business Metrics." *neptune.ai*, 7 Aug. 2023, neptune.ai/blog/recommender-systems-metrics.