**Movies**

Recommendation System

|  |
| --- |
| Team members |
| Mohammed Yasser |
| Salma Fahmy Hassan |
| Rahma Ramadan Hassan |
| Tarek Mostafa |
| Mostafa Ossama |

Contents

[Introduction 4](#_Toc198036061)

[**Dataset Overview** 4](#_Toc198036062)

[**Summary of ratings.csv** 4](#_Toc198036063)

[Movies Dataset Analysis (movies\_df): 5](#_Toc198036064)

[**Data Cleaning & Feature Engineering** 5](#_Toc198036065)

[Users and Movies Statistics: 5](#_Toc198036066)

[Movies Data: 5](#_Toc198036067)

[Ratings Data: 5](#_Toc198036068)

[Tags Processing: 6](#_Toc198036069)

[Sample Extraction: 6](#_Toc198036070)

[Final Movies Table Preparation: 6](#_Toc198036071)

[Final Preprocessing Steps: 6](#_Toc198036072)

[Cleaned ratings\_full: 7](#_Toc198036073)

[**Streamlit Dashboard:** 7](#_Toc198036074)

[**Algorithms :** 9](#_Toc198036075)

[Models Implemented : 9](#_Toc198036076)

[Content-Based Recommendation Model: 9](#_Toc198036077)

[Neural Collaborative Filtering (NCF) Model: 11](#_Toc198036078)

[Hybrid Recommendation Model: 12](#_Toc198036079)

[**Evaluation & Comparison** 14](#_Toc198036080)

[**Deployment** 15](#_Toc198036081)

[**Challenges & Limitations:** 15](#_Toc198036082)

[**Conclusion:** 16](#_Toc198036083)

[**Future Work:** 17](#_Toc198036084)

[**References:** 17](#_Toc198036085)

[**Appendix :** 17](#_Toc198036086)

# Introduction

The rapid growth of online platforms has significantly increased the importance of personalized recommendation systems, especially in domains like movies, music, and e-commerce. This report explores the MovieLens 30M dataset — a widely used benchmark dataset for evaluating recommendation algorithms — and presents a comprehensive analysis and implementation of multiple recommendation models.

Our primary goal is to understand the structure and content of the dataset, clean and preprocess it effectively, and build models that can provide meaningful movie recommendations. We begin by exploring the ratings and movies data, followed by thorough data cleaning, feature engineering, and the application of three state-of-the-art recommendation algorithms: Content-Based Filtering and Neural Collaborative Filtering (NCF) and Hybrid Recommendation. Finally, we visualize key insights and evaluate model performance using standard metrics.

# **Dataset Overview**

## **Summary of ratings.csv**

* **Total ratings**: 33,832,162
* **Columns**:
  + userId: ID of the user who gave the rating
  + movieId: ID of the movie that was rated
  + rating: The rating score (from 0.5 to 5.0)
  + timestamp: Time of the rating (UNIX epoch format)
* **Timestamp Note:**
* Mean timestamp ≈ 1269361676, which corresponds to **2010-03-23**
* Max timestamp ≈ 1689843213, which is **2023-07-20**
* **User Ratings Analysis (ratings\_df)**
* **Missing values (nulls)**: 0 — No missing data.
* **Duplicate entries**: 0 — No duplicates found.
* **Unique users**: 330,975
* **Unique rated movies**: 83,239
* **Timestamp conversion**: Successfully converted using pd.to\_datetime.

## Movies Dataset Analysis (movies\_df):

* **Total movies**: 86,537
* **Movies in movies\_df but not rated in ratings\_df**: 3,298
* **Most frequent title**: Alone (2020) appeared 4 times.
* **Most common genre**: Drama in 12,246 movies.
* **Unique genres**: 1,796
* **Unique titles**: 86,330 (about 207 duplicates)

# **Data Cleaning & Feature Engineering**

## Users and Movies Statistics:

* Per user:
  + rating\_count
  + rating\_avg
* Per movie:
  + ave\_rating

## Movies Data:

* Extracted year from title.
* Converted genres to one-hot encoded format using get\_dummies.
* Merged ave\_rating into the movies table.

## Ratings Data:

* Merged ratings\_df with user statistics.
* Converted timestamp to datetime format.
* Exported cleaned files to CSV.

## Tags Processing:

* Combined all tags per movie into a column all\_tags.
* Applied **TF-IDF Vectorization** to transform text data.
* Merged TF-IDF results with movies.

## Sample Extraction:

* Selected a random sample of 1 million ratings:
* Purpose:
  + Speed up testing
  + Reduce computational load
  + Ensure reproducibility

## Final Movies Table Preparation:

* Merged tags and ratings into a unified movies\_full table.
* Removed duplicates.
* Dropped rows with missing movieId.
* Cleaned additional columns (year.1, etc.) and adjusted column names:

## Final Preprocessing Steps:

**Null Values in year**

* Found 618 missing values in the year column.
* Used fetch\_year\_from\_omdb(title) to query the OMDb API and fill missing years.
* Final missing values: 116, then filled with 0 using:

**Column Cleanup**

* Left out odd columns like zombies, york, etc. as they are byproducts of TF-IDF — not deleted, just noted.
* Ensured type consistency:
  + Converted genres to list
  + timestamp to datetime
  + title to string

## Cleaned ratings\_full:

* Found and removed 1251 duplicates:
* rating\_full.drop\_duplicates(inplace=True)
* Converted timestamp to datetime.
* Normalized rating\_count and rating\_avg using MinMaxScaler.

# **Streamlit Dashboard:**

Sample of The dashboard components:

* **Filter Options**: Filters for movie genre, release year, and user rating.
* **Rating distribution by time :**

A screenshot of a computer

AI-generated content may be incorrect.

* **Top-rated Movies**: A table showing the highest-rated movies.

A screenshot of a computer

AI-generated content may be incorrect.

* **Yearly Average Ratings**: Line chart of average ratings per genre over the years.

A screenshot of a computer

AI-generated content may be incorrect.

* **Statistical Summary Table**: Includes total ratings, average, median, and standard deviation per movie.
* **Download Option**: Allows users to export summary data in CSV format.

A screenshot of a computer

AI-generated content may be incorrect.

* **Sample Data Preview**: Displays filtered data based on selected options.

A screenshot of a computer

AI-generated content may be incorrect.

The dashboard supports interactive analysis, enables identification of trends, and helps in understanding genre popularity and rating patterns over time.

# **Algorithms :**

## Models Implemented :

In this project, we implemented three distinct recommendation models: a **Content-Based Recommendation Model** , a **Neural Collaborative Filtering (NCF) Model** and **Hybrid Recommendation Model**. Each model was built using a tailored pipeline, leveraging different data representations and learning strategies to generate personalized movie recommendations. Below is a breakdown of each model:

## Content-Based Recommendation Model:

This model is based on the assumption that if a user liked certain movies, they will also like similar ones based on content attributes (such as genres, release year, and movie titles). The model was constructed using a custom pipeline with rich feature engineering and neural network learning.

**Key Steps and Code Insights:**

1. **Data Preprocessing:**
   * The custom class EnhancedMovieDataProcessor was used to clean and process the dataset
   * Missing values were handled using strategies like filling genre columns and computing user-based preferences.
   * Genre columns were one-hot encoded and filtered to remove rare categories.
   * TF-IDF was applied on movie titles to generate textual features
2. **Feature Engineering:**
   * The model created sophisticated user and movie features including:
     + User rating average, standard deviation, number of ratings
     + Movie rating average, release year, genre vector, TF-IDF title vector
     + Temporal behavior (time of rating)
   * Final features were assembled and standardized.
3. **Model Building and Training:**
   * Implemented using TensorFlow and Keras through the EnhancedRecommenderSystem class.
   * The model was trained using a feed-forward neural network:
   * The model was compiled and trained using Mean Squared Error loss
4. **Evaluation:**
   * Evaluation metrics included RMSE and MAE:
     + **Train RMSE** ≈ 0.744
     + **Test RMSE** ≈ 0.859
     + **Train MAE** ≈ 0.563
     + **Test MAE** ≈ 0.643
   * Precision@K and Recall@K scores were low, highlighting limitations in top-K ranking.
   * A Precision-Recall curve was plotted for additional performance insights.
5. **Saving and Inference:**
   * The trained model was saved and could be reloaded later.

## Neural Collaborative Filtering (NCF) Model:

The second model utilized **collaborative filtering**, relying entirely on the interaction matrix of users and movies without considering the actual content of movies. We used TensorFlow Recommenders (TFRS) to implement a deep learning model that learns embeddings for users and items.

**Key Steps and Code Insights:**

1. **Data Preparation:**
   * The dataset was preprocessed by selecting the necessary columns: userId, movieId, and rating.
   * The ratings were normalized to a range between 0 and 1 using MinMaxScaler.
2. **Model Architecture:**
   * Implemented using tfrs.Model, the architecture included:
     + Embedding layers for users and movies
     + A multi-layer perceptron (MLP) to model interactions
     + Regression output layer for rating prediction
3. **Training Process:**
   * The model was compiled using Mean Squared Error loss and Adam optimizer.
   * Trained for 20 epochs with early stopping and learning rate reduction callbacks.
4. **Evaluation and Metrics:**
   * Evaluated on standard metrics:
     + **Test RMSE** = 1.0629
     + **Test MAE** = 0.8321
   * Additional classification metrics after converting ratings ≥ 3 to binary labels:
     + **Precision** = 0.8723
     + **Recall** = 0.8189
     + **F1-Score** = 0.8444
     + **NDCG** = 0.8963
5. **Recommendations:**
   * The model successfully generated personalized recommendations and retrieved similar movies based on embedding distance.

## Hybrid Recommendation Model:

This model combines the strengths of both content-based and collaborative filtering approaches, leveraging both content attributes and user interaction data to generate more accurate and personalized recommendations. The hybrid system integrates the predictions from both the Content-Based and Neural Collaborative Filtering models, refining the recommendations further by utilizing a neural network.

1. **Key Steps and Code Insights:**

* Data Integration and Preprocessing:

The hybrid system combines predictions from both the Content-Based and Neural Collaborative Filtering models.

* Missing ratings in the interaction matrix are handled by filling them with default values (e.g., zero or the mean rating).
* A unified dataset is formed, which includes predictions from both models alongside actual ratings for further training.

1. **Feature Engineering:**

* Input Features:

Content-based and collaborative filtering predictions are used as input features.

* Additional features such as user and movie metadata (e.g., genres, titles) may also be incorporated.
* These features are standardized before training to ensure they are in a comparable scale.

1. **Model Building and Training:**

* The hybrid model utilizes a simple feed-forward neural network:
* Inputs: Predictions from both the Content-Based and NCF models are concatenated as inputs to the network.
* Architecture: The network includes multiple fully connected (Dense) layers to capture interactions between features.
* Output Layer: A regression output layer predicts the final rating.
* The model is trained using:

Loss Function: Mean Squared Error (MSE)

* Optimizer: Adam optimizer
* Callbacks: Early stopping and learning rate reduction to avoid overfitting and optimize training.
* Evaluation and Metrics:

The model's performance was evaluated using several metrics:

RMSE: 0.9287

MAE: 0.7170

Precision: 0.8881

Recall: 0.6211

F1-Score: 0.7310

NDCG: 0.9893

* Interpretation:

RMSE indicates the average error between the predicted and actual ratings.

MAE reflects how far off the predictions are from actual ratings.

Precision and Recall highlight the model's performance in recommending relevant items and retrieving a complete set of recommendations, respectively.

NDCG measures the ranking quality, with a high score indicating the model's effectiveness in ranking items.

1. **Recommendations:**

* The hybrid model successfully generates personalized recommendations by combining the strengths of both content-based and collaborative approaches.
* The embeddings from the collaborative filtering model are used alongside content-based predictions, yielding a comprehensive recommendation system.

# **Evaluation & Comparison**

The Hybrid Recommendation Model was evaluated alongside the Content-Based and Neural Collaborative Filtering (NCF) models to compare performance across various metrics. The hybrid approach, which integrates both content-based and collaborative filtering techniques, was expected to improve the recommendation quality by leveraging the strengths of each individual model.

**Content-Based Model Evaluation:**

- Train RMSE: 0.744

- Test RMSE: 0.859

- Train MAE: 0.563

- Test MAE: 0.643

- Precision K and Recall K: Both metrics were lower than expected, indicating that while the model performed reasonably well in predicting individual ratings, it had limitations in the top-K ranking of recommendations.

- Precision-Recall Curve: A curve was plotted to provide additional insights, highlighting areas where the model could be improved.

**Neural Collaborative Filtering Model Evaluation:**

- Test RMSE: 1.0629

- Test MAE: 0.8321

- Precision: 0.8723

- Recall: 0.8189

- F1-Score: 0.8444

- NDCG: 0.8963

- The NCF model performed well in terms of classification-based metrics like Precision, Recall, F1-Score, and NDCG, suggesting it was effective in providing personalized recommendations. However, its RMSE and MAE values were higher compared to the Content-Based model, which indicates that while it captured user preferences well, it was less precise in rating predictions.

**Hybrid Model Evaluation:**

- RMSE: 0.9287

- MAE: 0.7170

- Precision: 0.8881

- Recall: 0.6211

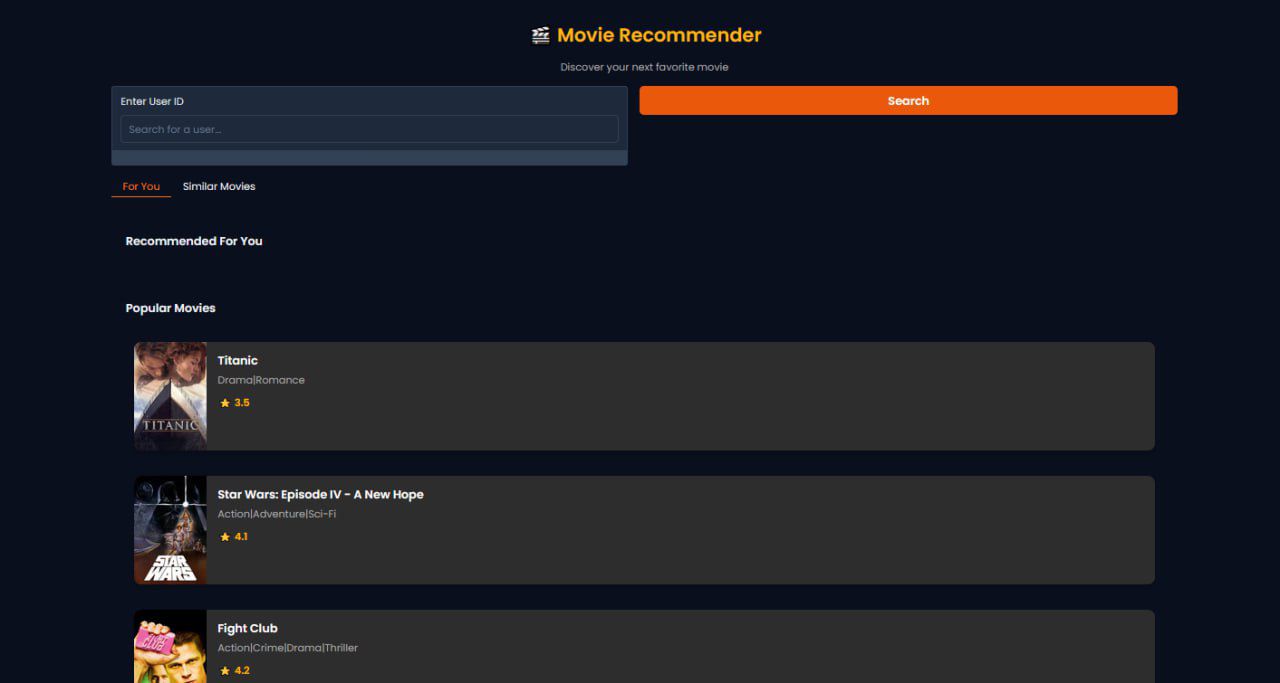
- F1-Score: 0.7310

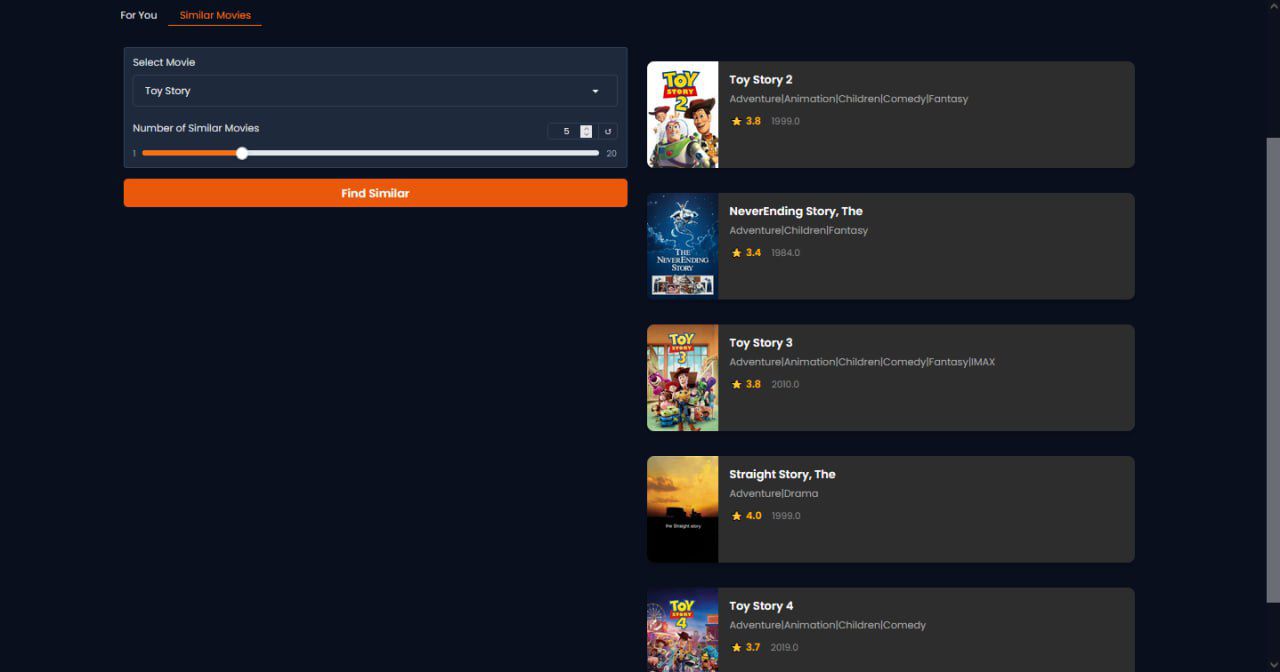
- NDCG: 0.9893

- The hybrid model showed a moderate improvement over both individual models, with particularly high NDCG, indicating that it ranked relevant items highly for users. While Precision was improved compared to the NCF model, the Recall slightly decreased, suggesting that the hybrid model might prioritize certain items but may miss some relevant ones. This tradeoff is typical when combining models to enhance precision.

# **Deployment**

We will outline the deployment of three distinct movie recommendation systems: a Hybrid Recommendation System, a Content-Based Recommendation System, and a Collaborative Filtering-based Recommendation System. Each of these systems uses different methodologies to recommend movies to users, offering diverse options for personalized movie discovery.





## 1. Hybrid Movie Recommendation System Deployment

The Hybrid Movie Recommendation System combines Collaborative Filtering and Content-Based Filtering to deliver personalized movie recommendations. This system recommends movies based on both user preferences and movie features, providing more accurate and diverse suggestions. It is deployed using a Gradio interface, offering a user-friendly web application for users to interact with the system.

**Objective**

The goal of this system is to:

* **User-based Recommendations**: Suggest movies based on user preferences.
* **Movie-based Recommendations**: Recommend similar movies based on a selected movie title.

**Model Architecture**

The hybrid model integrates:

* **Collaborative Filtering**: Uses user-item interactions to suggest movies based on what similar users liked.
* **Content-Based Filtering**: Suggests movies based on movie features like genre, director, and cast.

**Deployment with Gradio**

The Gradio interface allows users to:

* Input their user ID for personalized recommendations.
* Choose a movie title to find similar movies.
* View movie details including poster, genres, and ratings.

**Key Functions**

* get\_movie\_poster: Fetches movie posters from the TMDB API.
* generate\_user\_recommendations: Generates personalized recommendations for a user.
* find\_similar\_movies: Suggests similar movies to a selected movie.

## 2. Content-Based Movie Recommendation System Deployment

The Content-Based Movie Recommendation System suggests movies based on movie features such as genre, rating, and metadata. It uses the MovieLens dataset and is deployed via a Gradio interface for easy user interaction.

**Key Features**

* **Personalized Recommendations**: Tailored suggestions based on a user’s ID.
* **Similar Movies**: Recommends similar movies based on the features of a selected movie.
* **Movie Details**: Displays movie posters, titles, genres, and ratings fetched from the TMDB API.

**Technical Components**

* **Gradio Interface**: Provides sections for user input, personalized recommendations, and similar movies.
* **TMDB API Integration**: Fetches dynamic movie posters based on the movie title and year.
* **Content-Based Filtering**: Uses movie metadata (e.g., genres, ratings) to find similar movies.

**User Interaction**

* **User Recommendations**: Enter a user ID to receive personalized movie recommendations.
* **Similar Movies**: Select a movie to get suggestions for similar movies based on features.

**Deployment**

The system is launched locally using Gradio with the command interface.launch(). A public link can be generated using the share=True option.

## 3. Collaborative Filtering Movie Recommender Deployment

This system uses the Neural Collaborative Filtering (NCF) model to provide personalized movie recommendations based on a user's past ratings and preferences. It utilizes deep learning techniques for more sophisticated predictions.

**Model Overview**

* **Collaborative Filtering**: Predicts user preferences based on interactions with movies, using neural networks to learn patterns between users and movies.
* **Benefits**: Personalizes recommendations and can scale well with large datasets.

**Deployment with Gradio**

The system is deployed with a Gradio interface, where users can:

* Enter their user ID to receive personalized recommendations.
* View movie recommendations with details such as title, genre, rating, and poster.

**TMDB Integration**

Movie posters are fetched dynamically from TMDB based on the movie title using the get\_movie\_poster function.

**Recommendation Generation**

The generate\_user\_recommendations function uses the NCF model to predict top movies for a given user. The results are displayed in a formatted HTML format with movie cards.

The deployment of these three movie recommendation systems—Hybrid, Content-Based, and Collaborative Filtering—demonstrates the versatility and practicality of combining different recommendation techniques. Each system was successfully deployed using Gradio, providing an intuitive interface for users to interact with. Future work will focus on further improving personalization, scalability, and incorporating real-time recommendations.

# **Challenges & Limitations:**

1. **Scalability:**  
   The MovieLens 30M dataset is large, and even with sampling techniques, processing the data efficiently required substantial computational resources. In production scenarios with larger datasets, the computational load might become a significant challenge.
2. **Cold Start Problem:**  
   New users or new movies without any interaction data pose challenges for collaborative filtering methods. Both content-based and neural collaborative filtering models struggle to provide recommendations for these users or items at the start.
3. **Sparse Data Issues:**  
   The rating matrix used in collaborative filtering is sparse, meaning most users rate only a small subset of movies. This sparsity can lead to challenges in building accurate models, as there is limited data to identify similarities between users or items.
4. **Overfitting in Neural Models:**  
   Neural Collaborative Filtering (NCF) models, while powerful, tend to overfit when there is limited training data or when the model complexity is too high. Hyperparameter tuning and regularization techniques were essential in mitigating overfitting.
5. **Evaluation Metrics:**  
   Evaluation metrics like RMSE and MAE do not always align with user experience in recommendation systems. Metrics such as precision and recall were useful, but achieving good performance in top-k recommendations was still challenging in some cases.
6. **Interpretability of Models:**  
   While neural models like NCF can generate accurate predictions, their "black-box" nature makes them less interpretable than simpler models, such as content-based filtering. Understanding why a model recommends a particular movie can be important for user trust and system transparency.

# **Conclusion:**

* The Hybrid Model effectively combines content-based features and collaborative filtering embeddings to provide a more well-rounded recommendation system. The NDCG score of 0.9893 demonstrates its strong ranking capabilities, making it particularly effective for user-specific rankings.
* The Content-Based Model is better at predicting individual ratings with a lower RMSE and MAE, but it struggles in ranking relevance, as indicated by the lower Precision K and Recall K.
* The NCF Model excels in generating high-quality recommendations and personalized user interactions, evidenced by its impressive Precision and Recall, but its RMSE and MAE are not as strong as the Content-Based model, reflecting its challenge in accurately predicting ratings.

In summary, the Hybrid Model strikes a good balance between rating accuracy and recommendation relevance, offering a strong solution for personalized, ranked recommendations. It capitalizes on the strengths of both content-based and collaborative filtering models, making it the most robust of the three.

# **Future Work:**

1. **Real-Time Recommendations:**  
   The current models were trained offline, and future work can focus on incorporating real-time data to provide dynamic and up-to-date movie recommendations as users interact with the system.
2. **Incorporating More Features:**  
   Including additional features such as movie posters, reviews, and user demographics could improve recommendation accuracy. Moreover, integrating temporal factors like trends over time or seasonality could enhance the models.
3. **Exploring Multi-Objective Optimization:**  
   Moving beyond accuracy to incorporate other objectives, such as diversity and novelty in recommendations, could improve user satisfaction and engagement.

# **References:**

* **MovieLens 30M Dataset**. GroupLens Research. [Link](https://grouplens.org/datasets/movielens/)
* **TensorFlow and Keras Documentation**. TensorFlow. [Link](https://www.tensorflow.org/guide/keras)
* **Neural Collaborative Filtering**. He, X., Liao, L., Zhang, H., et al. (2017). Neural Collaborative Filtering. Proceedings of the 26th International Conference on World Wide Web, 173-182. [Link](https://dl.acm.org/doi/10.1145/3038912.3052569)
* **Scikit-learn Documentation**. Scikit-learn. [Link](https://scikit-learn.org/stable/)

# **Appendix :**

* [github link](https://github.com/moyasserelkotp/Recommendation-System-Depi)