Technical Report: Movie Recommendation System (Hybrid Model)

1. Overview

This project focuses on building an intelligent **Movie Recommendation System** that predicts and suggests movies to users based on their past ratings, interests, and overall movie preferences. Using the **MovieLens 100K dataset**, the system applies multiple recommendation algorithms — starting from simple baseline models to advanced hybrid techniques — to improve recommendation accuracy and personalization.

The system integrates **Content-Based Filtering (CBF)** and **Collaborative Filtering (CF)** techniques through a **Hybrid Model**, effectively addressing the **cold-start problem**, **sparse data issues**, and enhancing the quality of movie recommendations.

2. Objectives

- To analyze and preprocess user–movie rating data.
- To implement various recommendation algorithms including Random, Popularity, KNN, SVD, and NMF.
- To develop **Content-Based** and **Collaborative Filtering** models.
- To combine both approaches into a **Hybrid Recommendation System** for enhanced performance.
- To evaluate all models using metrics such as **RMSE**, **MAE**, **Precision@K**, and **Recall@K**.
- To design and deploy a **Streamlit web application** for real-time movie recommendations.

3. Dataset Description

The project utilizes the **MovieLens 100K dataset**, a standard benchmark dataset widely used in recommendation system research.
It contains **100,000 user ratings** (scale: 1–5) from **943 users** for **1,682 movies**.

Dataset Characteristics:

Feature	Description
Users	943 unique users, each rated at least 20 movies.
Movies	1,682 movies with title, release year, and genres.
Ratings	Integer values between 1 (lowest) and 5 (highest).
Genres	19 unique genres available in the dataset.

Key Files:

File Name	Description
u.data	Contains user–movie–rating–timestamp information.
u.item	Contains movie titles, release year, and genres.
u.user	Includes user demographics (age, gender, occupation).
u.genre	Lists available movie genres.

4. Methodology

The project was developed through systematic stages — **data preprocessing**, **exploratory data analysis (EDA)**, **model development**, **evaluation**, and **deployment**.

Step 1: Data Preprocessing

- Merged u.data, u.item, and u.user files.
- Removed missing and duplicate records.
- Encoded categorical features (genres) using multi-hot encoding.
- Normalized the user–item matrix for uniform analysis.

Step 2: Exploratory Data Analysis (EDA)

- Analyzed rating distribution across users and movies.
- Identified popular movies and highly active users.
- Visualized relationships using bar charts and histograms.
- Observed that most ratings lie between **3 and 4**, indicating moderate satisfaction among users.

5. Model Implementation

Multiple recommendation models were implemented progressively, from basic to advanced, for performance comparison.

A. Baseline Models

1. Random Recommender

- Generates random movie recommendations.
- Serves as a control baseline.

Performance:
RMSE = 1.8814 | MAE = 1.5067

2. Popularity-Based Recommender

- Recommends movies with the highest average ratings.
- Independent of user-specific preferences.

Performance:
RMSE = 0.9945 | MAE = 0.7936

B. Similarity-Based Models (KNN)

1. User–User KNN

- Finds users with similar preferences using cosine similarity.
- Recommends movies liked by similar users.

Performance:
RMSE = 0.9817 | MAE = 0.7769

2. Item–Item KNN

- Computes similarity between movies based on user ratings.
- Suggests movies similar to the ones a user liked.

Performance:
RMSE = 0.9823 | MAE = 0.7786

C. Matrix Factorization Models

1. Singular Value Decomposition (SVD)

- Decomposes the user–item rating matrix into latent user and movie factors.
- Captures hidden patterns to improve personalized recommendations.

Performance:
RMSE = 0.9352 | MAE = 0.7458

2. Non-Negative Matrix Factorization (NMF)

- Similar to SVD but enforces non-negative factors for interpretability.
- Performs well on sparse matrices.

Performance:
RMSE = 0.9481 | MAE = 0.7529

D. Content-Based Filtering (CBF)

- Uses movie metadata (title, genre, description) for recommendations.
- Implemented **TF-IDF vectorization** and **cosine similarity** to find movies similar to those the user already liked.

Key Libraries Used:
TfidfVectorizer, cosine_similarity (Scikit-learn)

Example:
If a user likes *Inception*, the system recommends similar movies such as *Interstellar* and *The Matrix*.

E. Hybrid Model

To achieve a balance between personalization and popularity, a **Hybrid Model** was developed, combining **CBF** and **SVD-based Collaborative Filtering** results.

Formula:

$$FinalScore = \alpha \times \text{CBF Score} + (1 - \alpha) \times \text{CF Score}$$

[where **α = 0.5** for equal weighting.]

Advantages:

- Balances user-specific and global recommendations.
- Handles new users and sparse ratings effectively.
- Increases recall and diversity of recommendations.

6. Evaluation Metrics

Metric	Description
RMSE (Root Mean Squared Error)	Measures deviation between predicted and actual ratings.
MAE (Mean Absolute Error)	Average magnitude of prediction errors.
Precision@K	Fraction of recommended movies that are relevant.
Recall@K	Fraction of relevant movies successfully recommended.

7. Experimental Results

Model	RMSE	MAE
Random	1.8814	1.5067
Popularity	0.9945	0.7936
User–User KNN	0.9817	0.7769
Item–Item KNN	0.9823	0.7786
SVD	0.9352	0.7458
NMF	0.9481	0.7529

8. Deployment (Streamlit Web App)

An interactive web interface was developed using **Streamlit** to demonstrate the model's functionality.

Features:

- Search for a movie or select a user ID.
- Displays top movie recommendations dynamically.
- select movie → Recommend similar movie
- Shows movie posters, ratings, and metadata using **OMDb API and TMDB API**.
- Clean and responsive UI for real-time interaction.

Technologies Used:

- **Backend:** Python
- **Frontend:** Streamlit
- **API Integration:** OMDb API for movie information
- **Libraries:** Pandas, NumPy, Surprise, Scikit-learn

9. Tools and Technologies

Category	Tools / Libraries
Programming Language	Python
Data Handling	Pandas, NumPy
Algorithms	Scikit-learn, Surprise
Visualization	Matplotlib, Seaborn
Deployment	Streamlit
Dataset	MovieLens 100K
External API	OMDb API

10. Conclusion

The **Movie Recommendation System** demonstrates the practical application of multiple recommendation algorithms and their performance comparison.
While simpler models like **Popularity** and **KNN** provided a reasonable baseline, advanced **Matrix Factorization** and **Hybrid** approaches significantly improved accuracy and personalization.

The **Hybrid Model** achieved the lowest RMSE and highest Precision/Recall, proving its effectiveness in combining user preferences with movie similarities.
Through a visually interactive **Streamlit dashboard**, users can now experience personalized movie recommendations efficiently and intuitively.

11. Future Enhancements

- Integrate **Deep Learning** models such as Neural Collaborative Filtering (NCF).
- Include **user demographics** and **movie reviews** as additional features.
- Deploy on **cloud platforms** for real-time scalability.
- Introduce **feedback-based learning** for continuous model improvement.

12. References

1. GroupLens Research. (1998). *MovieLens 100K Dataset*. University of Minnesota. Retrieved from https://grouplens.org/datasets/movielens/100k/
2. Streamlit Inc. (2023). *Streamlit Documentation*. Retrieved from https://docs.streamlit.io
3. OMDb API. (2024). *The Open Movie Database API*. Retrieved from https://www.omdbapi.com