Use Case Title: AI-POWERED MOVIE RECOMMENDATION SYSTEM

Student Name: S.SUJI

Register Number: 511023205004

Institution: JEI MATHAAJEE COLLEGE OF ENGINEERING

Department: B.TECH (INFORMATION TECHNOLOGY)

Date of Submission: 18.05.2025

---

1. Problem Statement

Build a Movie Recommendation System that suggests movies based on user input or preferences using machine learning techniques. The system should recommend movies similar to a selected title or based on a user's rating pattern. Your application should be intuitive, fast, and visually appealing. It can be built as a web app or a command-line tool and should be trained on a real-world movie dataset.

2. Proposed Solution

1. Data Collection

    Gather user ratings, watch history, and feedback from the platform

    Fetch movie metadata such as genres, cast, director, release year, and plot summaries using APIs like The Movie Database (TMDb)

2. Data Preprocessing

    Clean data by handling missing or inconsistent entries

    Normalize user ratings to a common scale (e.g., 1 to 5 stars)

    Transform movie metadata into feature vectors (e.g., genres encoded as one-hot vectors)

3. Recommendation Approach

Hybrid Recommendation System combining:

Collaborative Filtering: Uses user-item interaction matrix to recommend movies liked by similar users

Content-Based Filtering: Uses movie features to recommend movies similar to those a user has liked before

4. Model Development

Build a matrix factorization model (e.g., using Singular Value Decomposition) for collaborative filtering

Use cosine similarity or other distance metrics to find similar movies based on content features

Combine results by weighting or ranking to create final recommendations

5. Evaluation

Use metrics like Root Mean Square Error (RMSE), Precision@K, and Recall@K to evaluate model accuracy and relevance

Perform cross-validation and test with unseen user data

6. Deployment

Develop a simple web or mobile interface where users can:

Rate movies

Receive personalized recommendations

Search for movies and see details

Backend server handles data processing, model inference, and serving recommendations in real-time or batch mode

7. Feedback Loop

Continuously collect new user ratings and interactions

Periodically retrain models to adapt to changing user preferences and new movie releases

3. Technologies & Tools Considered

This refers to the specific software, programming languages, frameworks, and libraries used to build the recommendation system. For example:

Programming Languages: Python (commonly used for machine learning and data analysis)

Data Processing: Pandas, NumPy

Machine Learning Libraries: Scikit-learn, TensorFlow, PyTorch, or specialized recommendation libraries like Surprise

Databases: SQL databases like PostgreSQL or NoSQL like MongoDB for storing user and movie data

Web Frameworks: Flask, Django, or FastAPI if building a web app interface

APIs: For fetching movie data (e.g., The Movie Database API)

Tools: Jupyter Notebook for experimentation, Docker for deployment, Git for version control

4. Solution Architecture & Workflow

This outlines how the system is structured and the step-by-step process it follows to recommend movies:

Data Collection: Gather user ratings, movie metadata, and possibly user interaction data

Data Preprocessing: Clean data, handle missing values, normalize ratings

Model Building: Choose recommendation approach (Collaborative Filtering, Content-Based Filtering, or Hybrid)

Collaborative Filtering uses user-item interaction matrix

Content-Based uses movie attributes like genre, director, cast

Training & Evaluation: Train the model on historical data, validate accuracy using metrics like RMSE or Precision@K

Recommendation Generation: For each user, generate a ranked list of movie recommendations

User Interface: Display recommendations in an app or website

Feedback Loop: Collect new user data to continuously improve the model

5. Feasibility & Challenges

This discusses whether the project is doable and what obstacles might arise:

Feasibility:

Availability of quality data (user ratings, movie metadata)

Computational resources for training models

Time and expertise for development

Challenges:

Cold Start Problem: Recommending movies to new users or for new movies with little data

Data Sparsity: Most users rate only a small subset of movies, leading to sparse matrices

Scalability: Handling large datasets efficiently as users and movies grow

Bias and Fairness: Avoiding recommendations skewed by popular movies only

Real-time Updates: Keeping recommendations fresh as new data comes in

6. Expected Outcome & Impact

What you expect the system to achieve and its benefits:

Outcome:

Accurate and personalized movie recommendations

Improved user engagement and satisfaction

A user-friendly platform to discover movies tailored to individual tastes

Impact:

Helps users find movies they might otherwise miss

Drives higher usage or sales for streaming platforms or movie rental services

Provides insights into user preferences and trends

7. Future Enhancements

- **Incorporate Deep Learning:** Use neural networks for better pattern recognition in user preferences

- - **Hybrid Models:** Combine collaborative and content-based filtering for improved accuracy

- **Context-Aware Recommendations:** Factor in time, location, or mood for dynamic suggestions

- **Explainability:** Provide users with reasons why a movie is recommended

- **Social Features:** Include friend recommendations or social media integration

- **Multi-Language Support:** Cater to a global audience by including international movies and languages