

PROJECT REPORT

Smart Movie Recommender System

ML-Powered Web Application for Content-Based Movie
Recommendation

Presented by
Saiful Islam Rupom

Github Repository:
<https://github.com/saiful-islam-rupom/smart-movie-recommender-system.git>

Objective

The objective of this project is to build a smart, content-based movie recommendation system using machine learning techniques, and deploy it as a fully interactive, user-friendly web application using Streamlit. The system enables users to discover new movies similar to their preferences by analyzing movie metadata, generating recommendations based on textual similarity, and enhancing the experience with real-time data such as posters, overviews, and actor details via the TMDb API.

Abstract

This project demonstrates the use of machine learning and text vectorization techniques to recommend movies based on their content. The application processes metadata from a static movie dataset, performs NLP-based preprocessing using NLTK, converts text into vector form using CountVectorizer, and applies K-Nearest Neighbors (KNN) with cosine similarity to find similar movies. Additionally, the application integrates with the TMDb API to fetch real-time movie details, posters, and actor information. The model is optimized for performance and is deployed using Streamlit Cloud for public access.

Tools and Technologies

Category	Tools / Libraries
Language	Python
ML / NLP	scikit-learn, NLTK
Data Processing	pandas, numpy
Text Vectorization	CountVectorizer
Similarity Algorithm	Nearest Neighbors (cosine similarity)
API Integration	TMDb (The Movie Database) API
Web Framework	Streamlit
Deployment	Streamlit Community Cloud
Model Compression	joblib (compressed to 2.7 MB)

Workflow

1. Data Collection & Preprocessing

- The datasets are collected from online source- MovieLens where two million tag applications applied to 87,585 movies by 200,948 users. (Collected 10/2023 Released 05/2024)
- There are three datasets- 'movies.csv', 'tags.csv' & 'links.csv' containing movie metadata was including: movieId, title, genres, tags, imdbId, tmdbId etc.
- These datasets were cleaned and merged into a single dataset
- Basic NLP techniques were applied to normalize the text.

2. Natural Language Processing with NLTK

The Natural Language Toolkit (NLTK) was used for:

- Text normalization (e.g., converting to lowercase)
- Stopword removal — removing common non-informative words like “the”, “and”, “in”
- Stemming — reducing words to their root form using PorterStemmer (e.g., "playing" → "play") This step ensures better vector representation by eliminating noise from the textual data.

3. Feature Extraction with CountVectorizer

- CountVectorizer was applied to the processed tags column to convert it into a numerical vector space.
- Resulting vectors were integer counts of keywords for each movie.
- To optimize deployment, these vectors were converted to float32 to reduce memory usage. (With 0% data loss)

4. Model Training using Nearest Neighbors

- A K-Nearest Neighbors (KNN) model was trained using cosine similarity to find movies with similar feature vectors.
- The model computes the top 5 most similar movies based on content.
- Trained model was saved using joblib with compression (compress=3) for efficient deployment.

5. Model Serialization

Serialized artifacts:

- df.pkl: Preprocessed movie dataframe with TMDb IDs
- cv.pkl: Fitted CountVectorizer object
- nn_model_compressed.pkl: Compressed trained Nearest Neighbors model

6. Frontend Web App with Streamlit

A Streamlit web app was developed to:

- Let users select any movie
- Show movie poster, description, and top actors
- Display clickable recommendations
- Auto-refresh and update the page on new selections

7. TMDb API Integration

The app fetches:

- Posters
- Descriptions
- Actor names and profile images All using the TMDb API key.

8. Model Optimization for Deployment

Original 0.47GB model was reduced to 2.7 MB (With 0% data loss) by:

- Using float32 instead of float64 or int64
- Compressing with joblib This ensures fast loading on free-tier cloud platforms.

9. Deployment on Streamlit Cloud

- All code and assets pushed to GitHub
- App deployed at no cost using Streamlit Community Cloud
- App is fully public, fast, and ready for user interaction

Key Features

- Typable movie dropdown
- Poster + Overview of the selected movie
- Top 9 actors with name & profile pictures
- 5 content-based recommended movies
- Clickable recommendations (looping refresh)
- Fallback images if actor images are missing
- Optimized model loading (<3 MB)

Screenshot

Share ☆ ✎ 🔄 ⋮

Smart Movie Recommender System


Developed by [Saiful Islam Rijoom](#)

Select or type a movie:


Lie with Me (2005) ▼

Lie with Me


Happily unattached, the sexually voracious Leila satisfies her desires with a host of rapidly changing bed partners, unconcerned about the emotional consequences. But that all changes when she meets an artist looking for a deeper commitment.




Actors:




Lauren Lee Smith




Eric Ballou




Poly Shannon




Kate Lynch




Michael Raccello




Kristin Lehman



Ron White




Mayra Nguyen




Don Francis

You might also like:


(Click any of these recommended movies for further recommendation.)



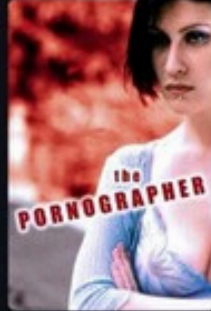
Silver (1993)




Intimacy (2000)



Bad Guy (Nabbeun namja) (2001)



Pornographer, The (Le pornographe) (2001)



Q (2011)

← Manage app

Live App: [Click to Try the App](#)
(Make sure Desktop-site is Turned-on for better user experience.)

Limitations

- Uses only content-based filtering (no user interaction history)
- Dataset is static, not auto-updated
- No support for multi-language or regional content
- API requests limited by TMDb rate limits
- No personalization or collaborative filtering
- Cold-start delay if hosted app is inactive

Future Enhancements

- Add collaborative filtering using user ratings and behavior
- Introduce user profiles for real-time personalized recommendations
- Replace static dataset with real-time updates from TMDb API
- Build a hybrid model combining content-based + collaborative filtering
- Incorporate movie popularity and ratings into recommendation ranking
- Support multiple languages and regional content via TMDb internationalization
- Implement fuzzy search and autocomplete for better movie selection
- Allow filtering of recommendations by genre or mood
- Enable filtering of movies by specific actors
- Improve UI/UX with loading animations and mobile responsiveness
- Deploy as a scalable REST API using FastAPI or Flask + React frontend

Conclusion

This project demonstrates the practical application of data science, ML and NLP techniques in a real-world scenario by developing a fully functional movie recommendation engine. The project showcases not just model building but also performance optimization, API integration, and public deployment. It is designed to be lightweight, scalable, and user-friendly.