Machine Learning Project Proposal

1. Project Title

SmartFlix: A Movie Recommendation System

SmartFlix is a machine learning-based movie recommender system that intelligently suggests movies to users based on their preferences. By utilizing both collaborative filtering and content-based filtering techniques, SmartFlix enhances user satisfaction by predicting what movies users would like to watch next.

2. Team Members

FC110583 – D.M.A.M.D.S Andradi – Team Lead / Project Coordinator

FC110597 – R.G.V.Dilsara - Data Handler

FC110601 - M.S.C.Udagedara - Model Trainer

FC110577 - P.D.K.Padukka - Deployment Lead

FC110605 – S.K.C.Dilshika - Developer

3. Problem Statement

In the digital entertainment era, users are overwhelmed by thousands of movie choices on platforms like Netflix, Amazon Prime, and others. Manually searching for movies that match one's taste can be time-consuming and often frustrating. Without intelligent recommendations, users may miss movies they would genuinely enjoy, leading to dissatisfaction and lower platform engagement.

The goal of this project is to build a machine learning-based movie recommendation system that can automatically suggest relevant movies to users based on their previous interactions (e.g., ratings, preferences). By combining collaborative filtering and content-based filtering techniques, the system aims to learn from user behavior and movie features to make accurate predictions. This project focuses on prediction and ranking as the core tasks of the ML pipeline.

Such a system enhances user experience, supports better content discovery, and demonstrates how machine learning can personalize digital services effectively.

4. Dataset Information

Include details about the dataset you will use:

- **Dataset Name**: MovieLens 100K Dataset
- **Source**: https://grouplens.org/datasets/movielens/100k/
- **Size**: 100,000 ratings by 943 users on 1,682 movies
- **Features**: User IDs, Movie IDs, Ratings (1-5), Timestamps, Movie titles, Genres
- **Why it is suitable**: This dataset is small, clean, and ideal for beginners. It includes real-world user-movie interactions and metadata, supporting both collaborative and content-based filtering techniques.

- **Preprocessing Plans**:

  Merge ratings and movie metadata

  Clean genre data (convert to TF-IDF features)

  Create user-item interaction matrix

  Normalize and scale rating values

5. Machine Learning Approach

ML Task:
Collaborative Filtering and Content-Based Filtering

Planned Models:

- Collaborative Filtering using Matrix Factorization

  This method looks at user behavior (what movies they rated) and finds similar users. If two users liked the same movies, the system assumes they have similar taste.

- Content-Based Filtering using cosine similarity on genres

  This method looks at the features of the movies. It recommends movies that are similar to ones the user already liked, based on those features.

- Optionally explore Hybrid Approaches

We may combine both methods to get better results. This helps fix the cold-start problem (for new users or new movies) and gives more accurate recommendations.

Justification:

- SVD works well with sparse user-item data

In most movie rating datasets, users have only rated a few movies. So, the table of users and their ratings is mostly empty this is called a sparse dataset. SVD is good at handling this kind of data. It helps find hidden patterns between users and movies, even when a lot of ratings are missing. It reduces the size of the data and fills in the blanks by predicting what a user might rate a movie, based on similar users.

- Content-based filtering helps in cold-start problems

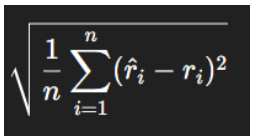
  One problem with collaborative filtering is the cold-start problem when there is a new user or a new movie with no ratings yet. Content-based filtering doesn't need ratings from other users. It looks at the features of the movies (like genre, actors, etc.) and matches them with the user's preferences. It's very helpful when we don't have enough user data yet.

6. Evaluation Metrics

To evaluate the performance of the SmartFlix movie recommendation system, we will use the following metrics:

1) Root Mean Square Error (RMSE)

RMSE is a standard metric used to evaluate how close the predicted ratings are to the actual user ratings. It is particularly useful for collaborative filtering models, as it penalizes large errors more heavily.

RMSE = 

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{r}_i - r_i)^2}$$

- $r\hat{}_i$ = predicted rating
- r_i = actual rating
- n = number of ratings

Lower RMSE values indicate better prediction accuracy.

2) Precision@K and Recall@K

These metrics are used to evaluate the ranking quality of the top-K movie recommendations given to a user.

- Precision@K: Measures the proportion of recommended movies in the top-K that are relevant to the user.

Precision@K = Number of relevant movies in top-K / K

- Recall@K: Measures the proportion of all relevant movies that appear in the top-K recommendations.

Recall@K = Number of relevant movies in top-K / Total number of relevant movies

These metrics help assess the practical usefulness of the system from a user experience perspective.

3) Mean Average Precision at K (MAP@K)

MAP@K is a commonly used metric to evaluate ranking-based recommendation systems. It considers both the precision and the position of relevant items in the recommendation list.

$$\text{MAP@K} = \frac{1}{N}\sum_{u=1}^{N}\frac{1}{k}\sum_{i=1}^{k}P(i)$$

Where P(i) is the precision at position i in the ranked list for user u, and N is the total number of users.

Definition of Success

A successful recommendation system will:

- Achieve **low RMSE** values (indicating accurate rating predictions).
- Deliver **high Precision@K and Recall@K**, meaning the top recommendations are both relevant and cover a large portion of what the user would have liked.
- Improve **MAP@K**, showing the system ranks relevant items higher.
- Provide personalized, satisfying movie recommendations to users.

7. Deployment Plan

Framework

We will use **Flask**, a lightweight Python web framework, to build the backend. It allows easy integration of our machine learning models for generating movie recommendations using collaborative and content-based filtering.

Hosting

The app will be deployed on **Heroku**, a cloud platform ideal for quick and scalable deployment of Flask apps. It supports Git-based deployment and provides free hosting for small-scale projects.

User Interface

Input fields for:

- Favorite movies, genres, actors, or directors.
- Option to rate a few known movies (for better personalization).

User Interaction:

- Users select preferences through a simple form.
- Submit button triggers the recommendation engine.

Recommendation Display:

- A list of recommended movies appears instantly.
- Each recommendation includes the movie title, poster, and a short description.

Output

Upon submission, the Flask backend will process the user's preferences using trained machine learning models. The system will generate and display a list of recommended movies instantly on the browser, providing a seamless and personalized viewing experience.

8. Project Timeline

Week	Task	Responsible Member(s)
Week 1	Dataset Collection, Exploratory Data Analysis (EDA), Preprocessing	Data Handler

	(Cleaning, Encoding, Feature Engineering)	
Week 2	Model Selection, Training, and Evaluation (Collaborative Filtering & Content-Based Filtering)	Model Trainer, Model Evaluator
Week 3	Hyperparameter Tuning, Model Finalization, Recommendation Performance Analysis	Model Evaluator
Week 4	Development of Flask-based UI and Backend Integration	Deployment Lead, Developer
Week 5	Deployment to Heroku, System Integration, Frontend & Backend Testing	Deployment Lead, Developer
Week 6	Final Report Preparation, Documentation Submission	Team Lead , Deployment Lead

9. Challenges and Risks

- Cold-start problem

  This happens when a new user joins or a new movie is added, and the system doesn't have enough data to make good recommendations.

- Data sparsity

  In large datasets, most users have only rated a few movies. This makes the user-item rating table very empty (sparse), and hard for the model to learn from.

- Model overfitting

  Overfitting happens when the model learns the training data too well but performs poorly on new data.

- Deployment challenges

Sometimes it’s hard to put the model on a website or app for users to use easily.

10. References

List any datasets, tools, libraries, tutorials, or papers you are using.

- MovieLens Dataset: https://grouplens.org/datasets/movielens/100k/
- Scikit-learn: https://scikit-learn.org/stable/
- Streamlit: https://docs.streamlit.io/
- Surprise ML Library: https://surpriselib.com/
- TF-IDF in NLP: https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction