Phase-2 Submission Template

Student Name: Sooriyakumar S

Register Number: 421623104148

Institution: Mailam Engineering college

Department: Computer science engineering

Date of Submission: 08/05/2025

Github Repository Link: https://github.com/suryasurya510/Delivering-personalized-movie-recommendations-with-an-AI.git

---

1. Problem Statement

Refined Problem:

With the exponential growth of digital media, users face difficulties in selecting movies that match their preferences. Traditional recommendation systems often lack personalization and context-awareness, leading to user dissatisfaction.

Type of Problem:

Classification and recommendation system (can also involve clustering and regression based on the model used).

Why this problem matters:

Providing users with personalized movie recommendations based on their preferences improves user engagement, satisfaction, and viewing experience. An Al-driven matchmaking system ensures accuracy, scalability, and relevance in recommendations, addressing the overwhelming content availability challenge.

2. Project Objectives

Key Technical Objectives:

Develop a user profiling system based on viewing history and ratings.

Apply machine learning algorithms (e.g., collaborative filtering, content-based filtering, or hybrid models) to recommend movies.

Integrate a real-time feedback system to fine-tune recommendations.

Build a responsive front-end interface for user interaction.

Host the system using cloud services or GitHub Pages (for frontend) with a backend using Flask/Django if needed.

3. Flowchart of the Project Workflow

Workflow:

1. Collect Movie Data (ratings, genres, user behavior)
2. Preprocess Data (cleaning, handling missing/duplicate values)
3. Build User and Movie Profiles
4. Train Recommendation Model (Collaborative Filtering/Content-Based/Hybrid)
5. Evaluate Model (accuracy, RMSE, etc.)
6. Deploy Model via Web Interface
7. Collect User Feedback and Retrain Model

4. Data Description

Dataset name and origin: MovieLens dataset (from Kaggle)

Type of data: Structured; includes ratings, timestamps, genres, and user IDs

Number of records and features: 20 million ratings by 138,000 users on 27,000 movies

5. Data Preprocessing

Missing values: Handled by imputation (average ratings) or removal (if irrelevant)

Duplicate records: Removed based on user ID + movie ID combinations

Data normalization: Ratings scaled if necessary for model performance

Feature encoding: Genres one-hot encoded, timestamps parsed if used

Outlier Treatment:

Detected outliers in user ratings using boxplots and z-scores.

Treated extreme values to improve model stability.

Data Type Consistency:

Converted data types (e.g., rating as float, dates as datetime).

Ensured all formats are uniform.

Encoding Categorical Variables:

Used Label Encoding for user IDs and movie IDs.

Applied One-Hot Encoding for movie genres.

6. Exploratory Data Analysis (EDA)

Univariate Analysis:

Plotted histograms and boxplots for rating distribution and genre counts.

Found majority ratings between 3.0 and 4.5.

Bivariate/Multivariate Analysis:

Used correlation matrix and pairplots to find relationships between user activity, movie ratings, and popularity.

Created scatterplots of average user ratings vs. number of ratings.

Insights Summary:

Key Observations: Popular movies tend to have slightly higher average ratings.

Influential Features: Number of user ratings, genre, and recency of movie release influ nodel predictions.

7. Feature Engineering

Created new features: for

User's average rating

Movie popularity score (based on number of ratings)

Genre vectors for each movie

Combined/Split columns:

Extracted year from movie release date

Encoded genres using multi-hot encoding

Techniques used:

Binning rating values into categories (e.g., low, medium, high)

Calculated user activity ratio (ratings per week/month)

Dimensionality Reduction (optional):

PCA used to red fewer compon genre vectors to or efficiency)

Justification:

Features enhance personalization by capturing user behavior and preferences in more detail, improving the recommendation accuracy.

8. Model Building

Selected Models:

Content-Based Filtering using Cosine Similarity

Collaborative Filtering using K-Nearest Neighbors (KNN)

Justification:

Content-based helps in cold-start problems (new users/movies).

Collaborative filtering captures hidden user-movie interactions effectively.

Data Split:

80% training, 20% testing

Stratified based on user IDs to maintain rating distribution

Training and Evaluation Metrics:

For regression (ratings prediction):

RMSE: Root Mean Square Error

MAE: Mean Absolute Error

R2 Score to measure prediction accuracy

9. Visualization of Results & Model Insights

Used confusion matrix to show classification accuracy.

Plotted ROC curve to evaluate model performance.

Created bar graphs to compare model scores.

Used feature importance plot to show key variables.

For regression, plotted residual vs actual values.

Random Forest gave best accuracy.

Top features: user activity, genre, past ratings.

Linear models underperformed on complex data.

Tree models captured non-linear patterns well.

Suggested ensemble models for better performance.

10. Tools and Technologies Used

Random Forest gave best accuracy.

Top features: user activity, genre, past ratings.

Linear models underperformed on complex data.

Tree models captured non-linear patterns well.

Suggested ensemble models for better performance.

11. Team Members and Contributions

Sooriyakumar S

Sivalingam

singaravelan

Raja S