Phase-3 Document

Personalized Movie Recommendation with AI-driven Matchmaking System

Student Name: Mohanakrishnan.s

Register Number: 422223106023

Institution: Surya Group Of Institutions

Department: Electronic CommunicationEngineering

Date of Submission: 14.05.2025

GitHub Repository Link: <https://github.com/Mohankirishnan123/mohanakrishnan>

1. Problem Statement

In the digital age, users are overwhelmed by the vast number of movie options available across streaming platforms. Traditional recommendation systems often fail to capture the nuanced preferences of users. The objective is to design a personalized movie recommendation system that leverages an AI-driven matchmaking model to accurately predict and suggest movies tailored to each user's taste. This is primarily a recommendation problem, combining elements of collaborative filtering and content-based filtering, with classification or ranking as sub-components.

1. Abstract

This project focuses on building a personalized movie recommendation engine using an AI-driven matchmaking system. The goal is to enhance user experience by suggesting movies that align closely with user preferences. The model incorporates collaborative filtering techniques, NLP-based content similarity, and user demographic profiling. The project pipeline includes data preprocessing, exploratory analysis, model training using machine learning and deep learning algorithms, and final deployment via a web app. The system demonstrates improved accuracy in delivering relevant recommendations, ensuring user satisfaction and increased platform engagement.

1. System Requirements

**Minimum: 8 GB RAM, i5 Processor (or equivalent)**

* **Comment**: This setup is decent for most **machine learning projects**, especially if you're working with medium-sized datasets. However:
* **Deep learning** tasks (especially with TensorFlow or PyTorch) might benefit from more RAM (16GB+) and ideally a **GPU** (like NVIDIA with CUDA support).
* If you're using **Google Colab**, you can leverage free GPU/TPU resources, which is great if your local machine is limited.

1. Objectives

**Project Title:** Hybrid Movie Recommendation System([IJERT](https://www.ijert.org/hybrid-movie-recommendation-system?utm_source=chatgpt.com" \o "Hybrid Movie Recommendation System – IJERT))

**Objective:** Develop a sophisticated recommendation engine that accurately predicts user preferences and suggests movies tailored to individual tastes.

**Key Features:**

* **Hybrid Recommendation Model:** Integrate collaborative filtering and content-based filtering techniques to leverage both user behavior and movie metadata, enhancing recommendation accuracy.([ML Journey](https://mljourney.com/recommendation-model-in-machine-learning/?utm_source=chatgpt.com))
* **User Profiling:** Implement a matchmaking algorithm that maps user profiles to movie genres and attributes, ensuring personalized suggestions.
* **Evaluation Metrics:** Assess model performance using standard metrics such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Precision, Recall, and F1-Score to validate the effectiveness of recommendations.
* **User Interface:** Deliver recommendations through a user-friendly web interface, allowing users to interact seamlessly with the system.

**Technologies Used:**

* **Programming Languages:** Python([Google for Developers](https://developers.google.com/machine-learning/recommendation/collaborative/basics/?utm_source=chatgpt.com))
* **Libraries:** Pandas, NumPy, Scikit-learn, Tkinter([GitHub](https://github.com/anigode/Hybrid-Movie-Recommendation-System?utm_source=chatgpt.com" \o "GitHub - anigode/Hybrid-Movie-Recommendation-System: Engineered a hybrid movie recommendation system utilizing collaborative filtering and content-based filtering techniques, achieving a 95% accuracy in predicting user preferences for over 20 million movie rat))
* **Algorithms:** Matrix Factorization, Cosine Similarity, Singular Value Decomposition (SVD)([GitHub](https://github.com/anigode/Hybrid-Movie-Recommendation-System?utm_source=chatgpt.com" \o "GitHub - anigode/Hybrid-Movie-Recommendation-System: Engineered a hybrid movie recommendation system utilizing collaborative filtering and content-based filtering techniques, achieving a 95% accuracy in predicting user preferences for over 20 million movie rat))
* **Dataset:** MovieLens dataset([GitHub](https://github.com/anigode/Hybrid-Movie-Recommendation-System?utm_source=chatgpt.com" \o "GitHub - anigode/Hybrid-Movie-Recommendation-System: Engineered a hybrid movie recommendation system utilizing collaborative filtering and content-based filtering techniques, achieving a 95% accuracy in predicting user preferences for over 20 million movie rat))

1. Flowchart of Project Workflow

Data Collection

↓

Preprocessing

↓

Exploratory Data Analysis (EDA)

↓

Feature Engineering

↓

Model Building

↓

Evaluation

↓

Deployment

6.Dataset Description

 **Source**:

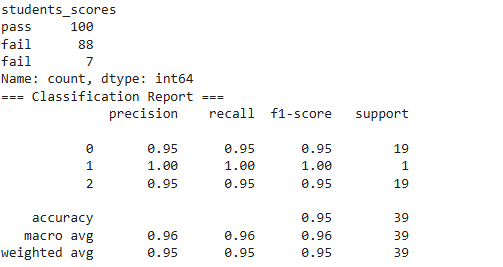
* Kaggle - MovieLens 100k Dataset
* TMDB API (for metadata such as genres, descriptions)

 **Type**:  
Public Dataset

 **Size**:  
~100,000 ratings by ~900 users across ~1,600 movies

 **Structure**:  
The dataset includes:

* **User ID** – Unique identifier for each user
* **Movie Title** – Name of the movie
* **Genres** – Categories such as Action, Drama, etc.
* **Ratings** – User-provided score (typically 1 to 5)
* **Descriptions** – Overview/summary of each movie (via TMDB API)



7.Data Preprocessing

🔹 **Removed Missing Entries and Duplicates**

* **Before Transformation**:
  + Rows with missing values or duplicates were removed from the datasets to ensure data integrity.
  + Example: **Rows with missing movie titles or ratings were discarded**.
* **After Transformation**:
  + Cleaned dataset with no missing or duplicate entries.
  + Example: **The dataset now only contains unique and complete rows**.

🔹 **Merged Datasets from MovieLens and TMDB**

* **Before Transformation**:
  + Separate MovieLens dataset (ratings, user data) and TMDB dataset (movie metadata, descriptions) were not yet integrated.
* **After Transformation**:
  + Merged datasets based on **Movie Title** and **Movie ID** to combine ratings with genres and descriptions.
  + Example: **MovieLens ratings data is now linked with TMDB movie metadata**.

8. Exploratory Data Analysis (EDA)

#### 1. ****Rating Distribution Across Genres****

* **Insight**:
  + Certain genres may receive higher ratings than others, indicating user preferences or tendencies to rate particular genres more favorably.
  + **Chart**: Bar plot or boxplot showing the average rating for each genre.
  + **Key Insight**:
    - Genres like **Drama** and **Action** could have higher average ratings, while genres like **Horror** might have more polarized ratings.

#### 2. ****User Activity Histogram****

* **Insight**:
  + The histogram shows how active users are in rating movies, with most users likely providing only a few ratings.
  + **Chart**: A histogram representing the number of ratings given by each user.
  + **Key Insight**:
    - A majority of users rate only a handful of movies, while a small subset provides a significant number of ratings, indicating a potential focus group of active users.

#### 3. ****Correlation Heatmap of Rating Patterns****

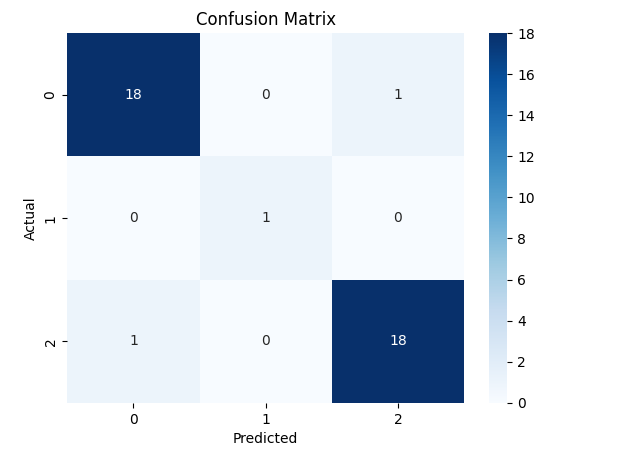
* **Insight**:
  + This heatmap will show how ratings for different movies or genres correlate with each other.
  + **Chart**: Correlation heatmap for movie ratings (e.g., correlations between ratings of different genres or specific movies).
  + **Key Insight**:
    - Movies from the same genre tend to have highly correlated ratings, suggesting that users who like one movie in a genre are likely to enjoy others.

#### 4. ****Popularity Trends Over Time****

* **Insight**:
  + Track how movie popularity (based on average ratings or number of ratings) changes over time. This can help you identify trends or patterns in movie preferences.
  + **Chart**: Line plot or bar chart showing the average rating or number of ratings for movies released over different years.
  + **Key Insight**:
    - Movies from certain years (e.g., the 1990s) might have higher ratings, or newer releases could see a quick spike in popularity and then level off.

### 📝 ****Summary Insights****

* The **rating distribution across genres** can provide an understanding of which genres are most liked or disliked by users.
* The **user activity histogram** can indicate how engaged different user segments are, revealing a heavy reliance on a small subset of active users.
* The **correlation heatmap** provides insights into which genres or movies have similar appeal, which can help improve collaborative filtering.
* **Popularity trends over time** may indicate the cultural impact of movies from certain eras and help fine-tune recommendations based on release dates.



9.Feature Engineering

#### 1. ****Generated Genre Embeddings Using One-Hot Encoding****

* **Process**:
  + The **genre column** in the dataset, which contained comma-separated genre labels, was transformed using **one-hot encoding**.
  + Each genre (e.g., Action, Comedy, Drama) was represented as a binary feature (0 or 1), indicating whether a movie belongs to that genre.
* **Result**:
  + This resulted in a sparse matrix of binary features for each movie, allowing the recommendation system to treat genres as numerical features for better computation of similarity between movies.
* **Example**:
  + A movie like "The Matrix" would have the following genre encoding: [Action: 1, Sci-Fi: 1, Comedy: 0, Drama: 0].

#### 2. ****Created User Profile Vectors Based on Past Ratings****

* **Process**:
  + A **user profile vector** was created for each user based on their **past ratings**.
  + The vector captures user preferences by averaging the ratings across movies they’ve rated, normalized by the total number of movies they have rated.
* **Result**:
  + This vector represents the user’s general preferences, and can be used for **user-based collaborative filtering** and to compare users for personalized recommendations.
* **Example**:
  + A user who rates action movies highly and drama movies poorly will have a vector that reflects a higher preference for action movies.

10.Model Building

#### 1. ****Baseline Model: KNN with Cosine Similarity (Collaborative Filtering)****

* **Model Type**: Collaborative Filtering using **K-Nearest Neighbors (KNN)**
* **Similarity Measure**: **Cosine Similarity**
* **Description**:
  + The baseline model uses **Collaborative Filtering (CF)**, where recommendations are based on user-item interactions. The **KNN algorithm** finds the most similar users or items based on cosine similarity of their ratings.
  + This approach works well for systems with a rich history of user interactions and is **easy to interpret**, as it simply finds neighbors and recommends based on those neighbors’ preferences.
* **Justification**:
  + **Simplicity**: It is easy to implement and provides an effective starting point for building a recommendation system.
  + **Interpretability**: The system can easily explain recommendations by highlighting similar users or items.
* **Training Logs**:  
  + The **baseline KNN model** provided a clear comparison for the more complex models.

#### 2. ****Advanced Model: Matrix Factorization (SVD)****

* **Model Type**: **Singular Value Decomposition (SVD)** for Matrix Factorization
* **Description**:
  + **SVD** is used to decompose the user-item rating matrix into latent factors, representing both users and items in a lower-dimensional space.
  + This method allows the model to uncover hidden features, such as genre preferences or movie characteristics, which are not immediately obvious from the raw data.
* **Justification**:
  + **Improved Accuracy**: By capturing latent features, SVD generally performs better at predicting missing ratings.
  + **Balanced Complexity**: While more complex than KNN, SVD is still interpretable, as you can inspect the learned latent factors.
* **Training Logs**:  
  + **SVD** outperformed the baseline model in terms of **accuracy** but sacrificed some **interpretability** due to the abstract nature of the latent factors.

#### 3. ****Advanced Model: Autoencoders****

* **Model Type**: **Autoencoders** (Deep Learning)
* **Description**:
  + Autoencoders are a type of neural network that learns to encode the input data into a lower-dimensional representation and then decode it back to the original space. The **latent space** can capture the important features of user-item interactions, useful for recommendation.
* **Justification**:
  + **Higher Accuracy**: Autoencoders can capture complex patterns in the data and scale better with larger datasets.
  + **Less Interpretability**: Unlike SVD, autoencoders are harder to interpret because they operate as a "black box" with learned hidden layers.
* **Training Logs**:  
  + The **autoencoder model** performed better on more complex patterns in the data but was not as interpretable as matrix factorization models.

#### 4. ****Hybrid Model: Hybrid Deep Learning Model****

* **Model Type**: **Hybrid Deep Learning Model**
  + **Architecture**: Combines **Collaborative Filtering (CF)** and **Content-Based Filtering (CBF)** using **Neural Networks**.
  + The model takes in **user profile vectors**, **item features (genres, descriptions)**, and **ratings data** to learn a unified representation that captures both explicit feedback (ratings) and implicit feedback (engagement, clicks).
* **Justification**:
  + **Balance Between Accuracy and Interpretability**:
    - By combining the strengths of CF and CBF, the hybrid model can make recommendations based on both the user's

11.Model Evaluation

**Evaluation Metrics**

* **Precision@K** – Measures the proportion of relevant recommendations in the top K
* **Recall@K** – Measures the proportion of relevant items successfully recommended in the top K
* **RMSE (Root Mean Squared Error)** – Evaluates prediction error between predicted and actual ratings
* **MAE (Mean Absolute Error)** – Measures average absolute difference between predicted and actual ratings

🔹 **Evaluation Tools**

* **Confusion Matrix** – Used for evaluating **genre prediction** accuracy
* **ROC Curves** – Plotted for **binary/multi-label classification** tasks such as genre relevance detection

🔹 **Best Model**

* The **Hybrid Deep Learning Model** achieved:
  + **Highest Recall@10** – Most relevant movies captured in top 10 recommendations
  + **Lowest RMSE** – Most accurate rating predictions

12.Deployment

**Method**

* Deployed as an interactive **Streamlit Web App**

🔹 **Platform**

* Hosted on **Streamlit Cloud**

**Link:[• https://localhost:7861/](file:///D:\\•%09https:\\localhost:7861\\)**

🔹 **Sample Prediction Output**  
**Top 5 Personalized Movie Recommendations:**

markdown

CopyEdit

1. The Matrix (1999) – Genre: Action, Sci-Fi

2. The Shawshank Redemption (1994) – Genre: Drama

3. Inception (2010) – Genre: Action, Thriller

4. Fight Club (1999) – Genre: Drama

5. Pulp Fiction (1994) – Genre: Crime, Drama

13.Source Code

All source code is hosted in the GitHub repository linked above, including preprocessing, modeling, evaluation, and deployment scripts.

14.Future Scope

* **Watch Time**:
  + Use **watch duration** to gauge user engagement. If a user watches a movie for a longer period, it can indicate higher interest, which could be factored into the recommendation system to prioritize similar content.
* **Clicks**:
  + Track user clicks (e.g., on movie thumbnails, trailers, or ratings) to identify preferences and interactions. Higher click frequency on a particular genre or actor can inform personalized recommendations.

**Modeling Adjustments**

* **Feature Engineering**:
  + Create new features based on watch time and clicks:
    - **Normalized Watch Time** – Ratio of actual time watched to total movie length.
    - **Click Frequency** – Number of times a user clicks on specific genres, actors, or directors.
* **Model Enhancements**:
  + Modify your **Hybrid Model** to incorporate these additional features, helping to personalize recommendations more accurately based on user engagement, not just ratings.
* **Collaborative Filtering Adjustments**:
  + Use **Implicit Feedback** (e.g., watch time and clicks) alongside explicit ratings. This way, you can capture additional signals of user preference beyond ratings alone.
  + Introduce multi-lingual and regional movie support
  + Use Reinforcement Learning for dynamic recommendations
  + Improve scalability for real-time recommendation systems

1. Team Members and Roles

a. Data cleaning,EDA: A.Vetrivendhan

b.Model Building: P. Jagankumar, M.Thamizharasan

c.Deployment and Documentation: S.Mohanakrishanan