STUDENT NAME : J.PREETHISHA

REGISTER NUMBER : 510523205043

INSTITUTION : Bharathidasan Engineering College

DEPARTMENT : B.TECH-IT

DATE OF SUBMISSION : 08-05-2025

GITHUB REPOSITORY LINK:https://github.com/preethi-2509/movie.git

Delivery Personalized Movie Recommendation with an AI-driven Matchmaking System

1.Problem Statement:

In today's digital world, users are overwhelmed with countless movie options across various platforms, making it difficult to find content that matches their tastes. A generic list of trending or popular movies often fails to satisfy individual preferences. Therefore, there is a strong need for a system that intelligently understands user behavior and delivers personalized movie recommendations. Solving this problem can greatly enhance user satisfaction, engagement, and platform loyalty.

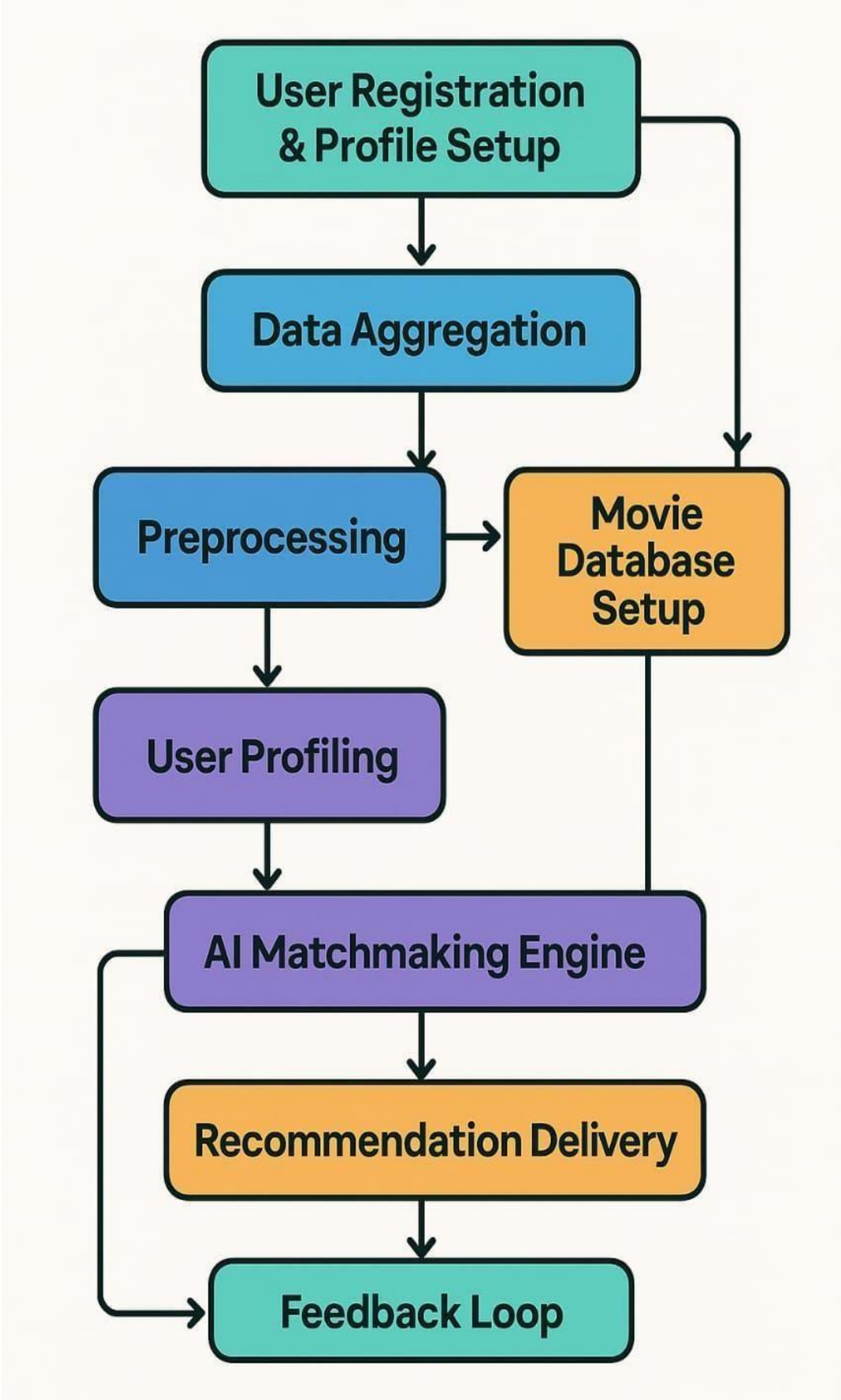
2. Objectives of the Project:

- build an AI-driven matchmaking system to recommend movies tailored to individual user preferences.
- Analyze user behaviors, movie metadata, and ratings to understand patterns.
- Develop and evaluate different recommendation models (content-based, collaborative filtering, hybrid).
- Visualize insights such as top genres, user clusters, and recommendation performance.

Optionally, deploy the system as a simple interactive web application for users to get theirrecommendations.

- The objective of this project is to develop an AI-driven matchmaking system that effectivelyrepresents movie metadata to deliver highly personalized movie recommendations, improving the accuracy and relevance of suggestions, enhancing user experience, and leveraging advanced techniques in natural language processing and machine learning to capture complex relationships between movie metadata and user preferences.

3. Flowchart of the project workflow:



4. Data description:

- Name: MovieLens 100k Dataset
- Source:GroupLens Research
- Type: Structured Data
- Records: 100,000 ratings | 943 users | 1682 movies Static Dataset
- Target Variable: User rating (explicit feedback)

5. Data Preprocessing:

Data Collection and Integration:

Merged data from multiple sources including:

User data (demographics, preferences, watch history)

Movie metadata (title, genre, cast, release year)

User interactions (ratings, likes, reviews, watch duration)

Handling Missing Values

User features: Filled missing demographic fields with the most frequent value or median.

Movie metadata: Replaced missing genres with "Unknown"; used median for missing numeric attributes (e.g., runtime).

Ratings/Interactions: Removed incomplete records without user ID, movie ID, or rating.

Duplicate Removal

Identified and removed duplicate entries in:

User ratings (same user rating the same movie multiple times)

Movie metadata (repeated movie IDs)

Data Filtering (Cold Start Mitigation)

Removed users with fewer than 5 interactions and movies with very few views to reduce data sparsity and cold-start issues.

Encoding Categorical Features

Genres: Applied multi-label binarization to account for multiple genres per movie.

User gender and other categorical demographics: Used one-hot encoding.

Language and country: Label encoded or one-hot encoded based on cardinality.

Feature Transformation and Scaling

Applied StandardScaler or MinMaxScaler to normalize:

Movie runtime

Average user ratings

Interaction timestamps (converted to recency values)

Text Processing (Optional, if reviews or descriptions are used)

Cleaned movie descriptions and user reviews by:

Removing punctuation and stopwords

Converting to lowercase

Extracted text features using TF-IDF or pre-trained language models (e.g., BERT) for content-based similarity.

Temporal Feature Engineering

Extracted temporal behavior insights such as:

Day of the week the user typically watches movies

Time of day preferences

Recency of last interaction to model short-term vs. long-term interests

User and Movie Profile Creation

Built user profiles by aggregating genre preferences and rating behavior.

Constructed movie profiles using genre, tags, average rating, and popularity.

Final Dataset Preparation

Structured the final dataset into:

User features matrix

Movie features matrix

Interaction matrix for collaborative filtering models

Prepared data for model input and split into training, validation, and test sets with attention to preserving time-based sequences.

6. Exploratory Data Analysis (EDA):

Exploratory Data Analysis is a crucial step in understanding the structure, patterns, and relationships in the movie recommendation dataset. It helps in identifying influential factors for user preferences and optimizes the AI-driven matchmaking system.

Univariate Analysis:

User Ratings Distribution:

Visualized using histograms to understand rating tendencies (e.g., skewed toward higher ratings).

Insights: Most users tend to rate movies between 3 and 5.

Genre Frequency:Bar plots showed the distribution of genres across all movies.

Common genres included Drama, Comedy, and Action.

User Demographics:

Age groups and gender distribution visualized using count plots.

Helped identify dominant user segments for personalization.

Bivariate Analysis

Ratings vs. Genres:Boxplots revealed which genres receive higher average ratings.

Example: Documentaries and Dramas had more consistent high ratings.

User Activity by Age Group:

Grouped bar charts showed average number of ratings per age group.

Younger users were more active but older users gave higher average ratings.

Interaction Time Patterns:

Line plots of user activity by hour of day and day of week.

Peaks seen during weekends and evenings, indicating high engagement periods.

- Analyzed rating distributions, popular genres, user activity.
- Used heatmaps and scatter plots to visualize user-movie interactions.

Key insights:

- Genre affinity varies by age and gender.
- Users tend to rate movies in certain genres higher.

7. Feature Engineering:

Feature engineering is essential for extracting meaningful information from raw data. For a personalized movie recommendation system driven by AI matchmaking, engineered features help represent user preferences, movie characteristics, and their interactions more effectively.

User-Based Features

Demographic Features:

Age, gender, location (encoded or bucketed into ranges).

In college settings: year of study, department, club memberships.

Behavioral Patterns:

Average rating given by the user.

Frequency of watching (e.g., number of movies watched per week).

Preferred genres:

inferred from the most-watched genres.

Recency of last interaction (helps model short-term interests).

Engagement Metrics:

Watch duration vs. total movie length (watch completion ratio).

Number of likes or favorites.

Item-Based (Movie) Features

Metadata Encoding:

Genre: Multi-hot encoding or embedding vector.

Director, cast: Encoded using frequency or target encoding.

Release year: Scaled or bucketed into decades/eras.

Popularity Metrics:

Average rating.

Number of views or ratings.

Recent trending status (computed from short-term view spikes).

Content-Based Attributes:

TF-IDF vectors from movie descriptions, tags, or reviews.

Embeddings from NLP models (e.g., BERT for plot summaries).

Interaction Features (User-Movie Matching)

Rating Matrix:User-item matrix used for collaborative filtering models (e.g., matrix factorization).

Genre Match Score:Similarity between user's preferred genres and movie genres (cosine similarity).

Watch History Overlap:Number of similar users who liked the same movie (used for neighborhood-based recommendations).

Temporal Features:

Time of day/week user typically watches movies.

Time since movie was last watched.

Composite & Engineered Features

User-Movie Affinity Score:Weighted score combining rating history, genre match, and engagement patterns.

Movie Diversity Index:Measures how varied a user's watch history is, encouraging novel recommendations.

Cold Start Flags:Binary features indicating whether the user or item has minimal data (new user/item).

Final Feature Set Preparation

Encoding: One-hot, label encoding, embeddings, or hashing as appropriate.

Scaling: StandardScaler or MinMaxScaler applied to numerical features.

Dimensionality Reduction (Optional): PCA or t-SNE for visualization or model optimization.

- Created user profiles based on watched and rated movies.
- Computed similarity matrices (cosine similarity, Pearson correlation).
- Extracted genre preference vectors.
- Combined behavioral data for more accurate predictions.

8.Model Building:

Algorithms Used:

- **Content-Based Filtering:** Based on genre and feature similarity.
- **Collaborative Filtering:** Using Matrix Factorization (SVD).
- **Hybrid Approach:** Merged content and collaborative features.
- **Train-Test Split:** 80/20 on rating interactions.
- **Metrics:** RMSE, Precision@K, Recall@K.

8. Visualization & Model Insights

1.Content-Based Filtering:

This approach analyzes the features of a movie (e.g., genre, director, actors) and suggests similar content based on a user's past preferences. It relies on metadata and the characteristics of movies rather than other users' preferences.

How it works:

If a user watches a science fiction movie like Inception, the system might recommend Interstellar due to shared features like genre and director.

The algorithm builds a user profile and refines recommendations over time based on watched content.

Limitations:

Struggles to recommend movies outside a user's viewing history.

Does not capture diverse preferences beyond a user's past interests.

2. Collaborative Filtering:

Instead of analyzing movie content, Collaborative Filtering recommends movies based on user interactions, ratings, and behaviors.

Two types:

User-Based Collaborative Filtering: Finds users with similar tastes and suggests what they liked.

Item-Based Collaborative Filtering: Suggests movies based on items similar to those a user has already watched.

Example:

If User A and User B have a similar watch history and User A watches The Matrix, the system may recommend The Matrix to User B even if they haven't watched it yet.

Limitations:

Struggles with new users (cold start problem).

Requires a large dataset for accurate recommendations.

3. Hybrid Recommendation System:

A Hybrid System combines content-based and collaborative filtering to overcome limitations and improve accuracy.

Example:

Netflix uses metadata (content-based filtering) and user behavior (collaborative filtering) to recommend movies tailored to individual preferences.

- Plotted genre preferences, recommendation accuracy, and user satisfaction scores.
- Feature importance from hybrid model helped understand influential variables.
- Visualized latent user and movie embeddings using t-SNE.

9. Tools and Technologies Used:

- **Languages:** Python

- **Libraries:** pandas, numpy, scikit-learn, surprise, LightFM, matplotlib, seaborn, Gradio/Streamlit.
- **Environment:** Google Colab / Jupyter
- **Deployment:** Gradio web interface for live demo.

10.TEAM MEMBERS AND CONTRIBUTION:

- PROBLEM STATEMENT AND PROJECT OBJECTIVES-[K.priyadharshini]
- FLOWCHART OF THE PROJECT WORKFLOW, DATA DESCRIPTION, DATA PREPROSSING, EXPLORATORY DATA ANALYSIS-[j.Preethisha]
- FEATURE ENGINEERING AND MODEL BUILDING-[M.Gayathri]
- VISUALIZATION OF RESULTS & MODEL INSIGHTS AND TOOLS & TECHNOLOGIES USED-[k.Megala]