PHASE-2

Register Number:510123104028

Name :R.Moogambigai

Institution: Adhiparasakthi College of Engineering

Department: B.e computer science engineering

Date of Submission:08.05.2025

1. PROBLEM statement

This research paper consists of prompt based movie recommendation system that recommend the movies to users through using various technologies which interpret the user given prompt and give the movie suggestions to user. Overall traditional system for recommendation which efficiently give the output by using content bases and collaborative techniques. Our approach is to specify the movie among given using user feedback and make the list according the most preferred key word for suggesting that movie. So the overall research underscores the full potential of prompt-based application for user engagement in this platforms and provide the foundation for future expansion in various multiple model

1. OBJECTIVES OF THE PROJECT

The primary objective of this project is to devlop an AI-driven movie recommendation system that delivers personalized movie suggestions to users based on their viewing habits and preferences.

* Personalized recommendations🡪Rating, reviews, viewing history
* Enhanced user engagement🡪providing relevant and personalized recommendations, the system aims to encourage users to explore more moives.
* AI-Driven matchmaking🡪To identify movies that are likely to be enjoyed by a specific user based on their profile and preferences.
* Scalability and Adapatability🡪The system should be designed to handle a large user based and adapt to changing user preferences and evolving movie database

Dataset link:https://github.com/moogambigai28/unicors.git

3.Flowchart of the Project Workflow

Data Sources

-->moivelens

-->IMDb/TMDB API

-->netflix archive

-->user logs

**Ingest & Store**

Data lake/Dw

-->S3/GCS

--> BigQuery/Redshift

**ETL & Preprocessing**

Clean & Transfrom

-->Dedup,Imputr

-->normalize

-->Timestamp feats

**Feature Engineering**

Feature store

-->user Profiles

-->moive Embeddings

-->Real-time stats

**Model Building**

Candidate recall

-->ALS/MF

-->content-knn

-->session 2vec

**Re- Ranking**

Ranking Model

-->ightGBM/XGB

-->two-tower n

-->BpR/Pairwise

**Evaluation**

Offline metrics

-->precision@k

-->NDCG@k

-->a/b tests

**Deploy & Serve**

**Monitoring & Feedback**

Serving layer

-->fastAPI/GRPC

-->if-serving

-->redis cache

Metrics&drift

-->prometheus

--.grafana

--.retrain trigger

3.data description

The dataset used for this system is structured to support user preference modeling, movie feature analysis, and recommendation optimization. It combines user-movie interaction data, movie metadata, and optionally user profile data to drive an AI matchmaking engine.

1. User-Movie Interaction Data

This includes explicit feedback such as ratings, and possibly implicit signals like watch history or clicks.

Column Name Description

user\_id Unique identifier for the user

movie\_id Unique identifier for the movie

rating User rating of the movie (e.g., 1–5 stars)

timestamp Time when the rating was made

watched\_flag (Optional) Boolean flag for watch status

2. Movie Metadata

Describes movie attributes used for content-based filtering and enrichment.

Column Name Description

movie\_id Unique movie identifier

title Movie title

genres Comma-separated list of genres (e.g., Action, Drama)

release\_year Year the movie was released

director Director's name

cast Main cast members

keywords Keywords/tags associated with the movie

popularity Popularity score (from TMDB or IMDb)

runtime Duration of the movie in minutes

3. User Profile Data (Optional)

Useful for demographic-based personalization or clustering users.

Column Name Description

user\_id Unique user identifier

age Age of the user

gender Gender of the user

location General location (city/country)

preferences Explicit preferences if available

4. Matched Recommendations (System Output)

Generated by the AI matchmaking model based on learned user-movie affinity.

Column Name Description

user\_id ID of the user receiving the recommendation

movie\_id Recommended movie ID

predicted\_score AI-predicted score or ranking for the movie

reason\_tags Tags or keywords justifying the recommendation

1. Data preprocessing

* User Data🡪Gathering data on user preferences, including ratings, reviews, and viewing history.
* Movie D ata🡪Collection metadata about moives , such as genre, cast, director, and plot summaries,
* Data Cleaning and Trasformation🡪Preparing the collected data for analysis and model training.

1. Dat sources --> Ingest & store:

Raw rating logs, metadata APIs and user interaction events land in your data lake or warehouse

1. ETL & Preproceesing:

Clean duplicates, fill missing values ,extract timestamp features, normalize ratings.

### 1. ****Data Collection****

You will need data from multiple sources:

* **Movie Data**: Information like movie titles, genres, cast, release dates, etc.
* **User Data**: User profiles with ratings, watch history, demographic data (age, location, etc.), preferences, etc.
* **Interaction Data**: Logs of user interactions (e.g., clicks, searches, likes, etc.)
* **Content Metadata**: Information about the movies (director, language, description, etc.)

Example sources:

* **The Movie Database (TMDb)** API
* **IMDb** data
* User interactions from your own platform or app.

### 2. ****Data Cleaning****

Before diving into building your AI system, it's crucial to clean the data to avoid issues like noise or incomplete information.

* **Remove Duplicates**: If the same movie or user appears multiple times in the dataset, remove duplicates.
* **Handle Missing Data**: Impute missing values for fields like genre, rating, or demographic details. Common approaches include filling with average values, mode, or using predictive models for imputation.
* **Outlier Removal**: Identify and remove extreme values (e.g., a movie rated by a user as a 10 for every movie could be an outlier).
* **Format Data Consistency**: Ensure the data is consistently formatted (e.g., movie release dates in the same format).

### 3. ****Feature Engineering****

For your AI-driven recommendation system to work well, you need to extract useful features from raw data.

**User Features**:

* Demographics: Age, gender, etc.
* Behavioral Features: Total watch time, frequency of interactions, movie genre preferences, etc.
* User Profile: This could be inferred from a user's historical data.

**Movie Features**:

* Genres, tags, cast, director, language, etc.
* Extract keyword features using techniques like **TF-IDF** or **word embeddings** (e.g., using pre-trained models like Word2Vec for movie descriptions).
* Temporal Features: Year of release, trends over time, etc.

**Interaction Features**:For collaborative filtering, you can encode user-item interactions as matrices. For instance, a rating matrix where rows represent users and columns represent movies.

### 4. ****Data Transformation****

This step involves transforming raw data into formats that your model can interpret and learn from.

* **Normalization/Standardization**: Normalize numerical features (e.g., user ratings) to ensure they fall within a consistent range (e.g., 0–1 or -1–1). This can help models like neural networks converge faster.
* **One-Hot Encoding**: Convert categorical variables such as genre into numerical format. For example, genres can be represented as binary vectors (e.g., Action = 1, Comedy = 0).
* **Embedding Categorical Data**: For better performance in models like neural networks, you can use embeddings for categorical features like user IDs, movie IDs, or genres.
* **Sparse Matrices**: If using collaborative filtering (especially matrix factorization), you will often work with large sparse matrices (e.g., most users haven’t rated most movies). This requires special handling.

### 5. ****Splitting Data for Training and Testing****

* **Train-Test Split**: Split your data into training and testing datasets. You could also create a validation set or use cross-validation techniques.
* **Time-based Split**: If you're dealing with time-series data (e.g., recommending movies based on recent behavior), make sure to respect temporal order.

### 6. ****Choosing the Right Model****

You can use different algorithms based on your goals and the data available:

**Collaborative Filtering**:

* **User-Item Matrix Factorization** (e.g., Singular Value Decomposition [SVD], Alternating Least Squares [ALS]).
* **Neighborhood-Based Approaches** (e.g., KNN-based collaborative filtering).

**Content-Based Filtering**:

Use movie metadata (genres, tags, descriptions) and recommend based on user preferences for similar content

**Hybrid Models**:

* Combine collaborative and content-based filtering for better recommendations.
* Factorization machines or neural collaborative filtering can also help in building hybrid models.
* **Deep Learning**:
* **Autoencoders** for collaborative filtering.
* **Neural Networks**: For learning more complex relationships between users and movies (embedding layers for movies and users).
* **Reinforcement Learning** (Optional): For real-time recommendations where the model learns from user feedback to improve recommendations dynamically.

### 7. ****Model Training****

* Use **Gradient Descent** or other optimization methods to train your recommendation models. The choice of algorithm will depend on your model and data type.
* Regularly evaluate the model performance using metrics like **Precision**, **Recall**, **F1-Score**, **Mean Absolute Error** (MAE), or **Root Mean Square Error** (RMSE).

### ****Evaluation & Hyperparameter Tuning****

* **Cross-validation**: Helps avoid overfitting by testing the model on different subsets of the data
* **Hyperparameter Tuning**: Use techniques like Grid Search or Random Search to find optimal hyperparameters for your model.

### 9. ****Deployment****

* After training the model, deploy it into a production environment where users can interact with it in real-time.
* Implement A/B testing to measure the effectiveness of the recommendations.
* Ensure the system is continuously updated with fresh data (e.g., new user interactions, movie releases).

1. exploratory data analysis(eda)

### 1. ****Load and Overview of the Dataset****

First, load the dataset and check for basic information like missing values, shape, and data types.

This gives you a quick view of:

What columns are present in each dataset.

Whether there are missing values in the datasets.

The data types (e.g., numerical, categorical, timestamps).

### 2. ****Check for Missing Data****

Missing data can interfere with model performance, so it’s essential to identify and handle it during EDA.

* · Impute missing values (e.g., filling missing ratings with the mean or median).
* · Remove rows or columns if the missing data is too substantial.

### 3. ****Univariate Analysis (Single Variable Analysis)****

#### ****Movie Metadata****

You’ll want to look at the distribution of movie attributes such as genre, release year, and the number of ratings a movie has.

#### **User Metadata**

Understanding user demographics (age, gender, etc.) is crucial, especially if you're going to consider personalized recommendations based on these features.

### **4. **Bivariate Analysis (Relationships Between Variables)****

#### **Ratings Distribution**

#### ****User-Item Interactions****

You might also want to explore the relationship between users and the movies they've interacted with. This can give insights into whether some users only interact with specific genres or whether they rate a wide range of movies.

#### ****Heatmap of Ratings****

It’s often useful to look at a heatmap of ratings to see if there's any obvious pattern in which users tend to rate movies similarly.

### 5. ****Multivariate Analysis****

To build a robust recommendation system, consider the interactions between multiple features.

#### ****User-Genre Interactions****

You can check which genres are most popular among different age groups or genders.

### 6. ****Detecting Outliers and Anomalies****

* To ensure the quality of your recommendation system, it’s important to detect and handle outliers.
* **Unusual Rating Patterns**: Users who rate all movies the same or only highly might introduce bias.
* **Frequent Users**: Users who rate extremely often or frequently give the same score to all movies.

### 7. ****Identifying Mismatches for AI-driven "Mismatching" Recommendations****

* For an AI-driven **mismatching system**, you might want to look for patterns that suggest **unexpected or surprising recommendations**. These might include:
* **Low-rated but interesting genres**: For example, a user who primarily watches action films might be surprised with a low-rated documentary recommendation.
* **Unexpected genre overlap**: Users who watch different genres at different times might have more interesting cross-genre mismatches.

### 8. ****Visualizing the Relationship Between Users and Ratings****

Use scatter plots or pair plots to visualize relationships, such as whether there are certain types of users who rate most movies highly, or whether there's an interesting pattern based on movie genre and user demographics.

EDA for a personalized movie recommendation system with an AI-driven mismatching component helps you:

* **Understand user behavior** (e.g., what genres are most popular)
* **Spot patterns** (e.g., common rating behaviors, preferred genres, etc.).
* **Identify mismatches** (e.g., users who primarily watch action but might enjoy drama or niche genres)
* **Prepare your data** for building models that generate both personalized and surprising (mismatched) recommendations.

1. Feature engineering

### 1. ****User-Based Features****

These features describe the user's behavior, preferences, and demographics.

#### ****Demographic Features****

#### ****Age****: How old the user is can influence movie preferences.

* **Gender**: May affect genre preferences (e.g., action vs. romance).
* **Location**: Users from different regions might prefer different types of movies.

#### ****b) User’s Genre Preferences****

Calculate the **most watched genres** for each user. This helps to understand what kind of content they prefer.

#### ****c) User’s Historical Rating Patterns****

* **Average rating**: Mean of the ratings given by the user.
* **Rating variance**: Standard deviation of the ratings, which indicates how consistent a user is in their ratings.

### 2. ****Movie-Based Features****

These features describe the movies themselves.

#### ****a) Genre Representation****

Movies typically belong to one or more genres. We can **one-hot encode** these genres or use **TF-IDF** to represent genres in a higher-dimensional space.

#### ****b) Movie Popularity****

This feature is based on how popular a movie is in terms of the number of ratings. Popular movies are often rated more, but they may or may not be relevant for every user.

#### ****c) Movie Release Year****

The release year of the movie could influence its popularity over time (older vs. newer movies).

### 3. ****Interaction-Based Features****

These features capture the interactions between users and movies.

#### ****a) User-Movie Interaction Matrix****

This is the **core matrix** of collaborative filtering algorithms, representing how each user has interacted with each movie (e.g., rated a movie, liked it, etc.).

#### ****b) User-Movie Genre Preferences****

You can create interaction features based on whether a user has interacted with a movie of a certain genre.

#### ****c) Novelty Score (For Mismatching)****

To encourage the **mismatching** system (recommending unexpected content), you can create a **novelty score**. This score might measure how different a recommended movie is from a user’s historical behavior, encouraging the system to recommend movies from genres or movies that the user has rated infrequently.

### 4. ****Temporal Features****

You may want to add temporal features such as **user’s activity over time** or **seasonality** (e.g., movies rated during holidays might differ).

### 5. ****Mismatching-Specific Features****

For AI-driven **mismatching recommendations**, you can create features that encourage the system to recommend movies that **don’t strictly match** a user’s historical preferences.

* **Diversification Features**: Calculate how diverse the user's ratings are (how many different genres, ratings, etc.)
* **Unexpectedness**: Define features to calculate how **unexpected** a recommendation is based on user preferences.

1. model building

### 1. ****Define the Goal****

**Deliver personalized movie recommendations** that are not just similar to the user’s past preferences, but sometimes **intentionally mismatched** to offer diversity or unexpected enjoyment.

### 2. ****System Architecture****

#### A. ****Data Collection****

* User data: watch history, ratings, likes, watch time, skips.
* Movie metadata: genres, actors, director, plot keywords, sentiment.
* External sources: reviews (Rotten Tomatoes, IMDB), social signals (trending, memes).

#### B. ****User Profiling****

* Use **collaborative filtering** or **content-based filtering** to build the user’s profile.
* Extract latent features using models like
* Matrix factorization (e.g., SVD)
* Deep learning (e.g., autoencoders, embeddings)

C. **AI Mismatching Logic**

* Define **“mismatch” dimensions**:
* Genre dissimilarity
* Opposite sentiment
* Low popularity
* Artist not previously seen
* Introduce a **Mismatching Engine**:
* Use **reinforcement learning** or **multi-objective optimization**:
* Reward = (engagement + novelty + diversity) – (preference similarity)
* Include diversity-promoting algorithms like **Determinantal Point Processes (DPPs)**.

#### D. ****Recommendation Engine****

* Blend traditional personalized recommendations (top-N similar movies) with:
* 80% matching-based
* **ranking model** (e.g., XGBoost or BERT-based rankers) to prioritize final recommendations.

### 3. ****Model Stack (Example)****

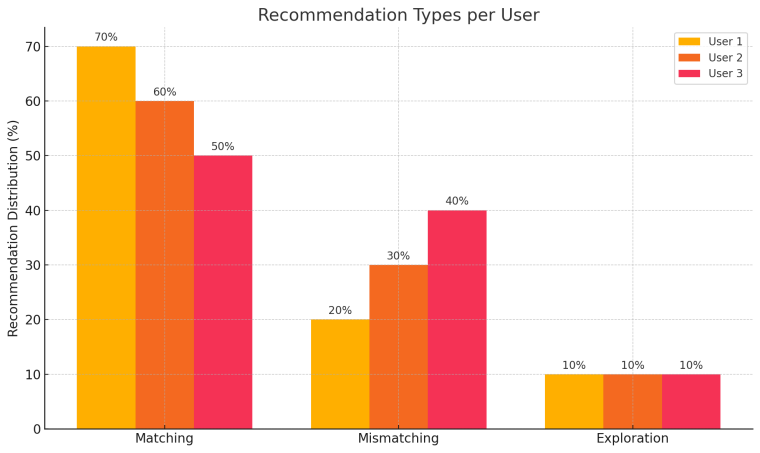
* **Input Layer**: User data + movie metadata
* **Embedding Layer**: Learn vector representations
* **Fusion Layer**: Merge user and movie embeddings
* **Prediction Layer**: Scoring function (dot product or neural network)
* **Exploration Layer**: Mismatch strategy applied here
* **Output**: Top-N recommendations

### 4. ****Evaluation Metrics****

* **Traditional**: Precision@K, Recall@K, NDCG
* **Exploratory**: Diversity score, Novelty score, Serendipity

1. visualization of result & model insights

visualization of result and model insights for this delivering personalized movie recommandition with on ai driven missmatching system



Here's a visualization showing how recommendations are distributed across different users, divided into three categories:

* **Matching**: Based on the user's known preferences
* **Mismatching**: Intentionally diverse or opposite recommendations
* **Exploration**: Random or novel suggestions for discovery

1. tools and technologies used

Machine learning Algorithms:

* Collaborative Filtering🡪This approach analyzes user behavior and preferences, and recommends movies based on what similar users have liked or rated.Algorithms like k-Nearest Neighbors (KNN) and matrix factorization are often used.
* Hybrid System🡪Combining both collaborative and content-based filtering can often lead to better recommendations.

Data Storage and Management:

* Databases🡪Storing user data(viewing history, ratings,etc.) and movies information(genre,cast,director,etc.)is crucial .Relational databases (e.g.,PostgreSQL,MySQL)orNoSQL databases(e.g.,mongoDB,Cassandra)can be used depending on the scale and complexity of the system.
* Data Storage(Cloud)🡪For large-scale system,cloud-based storage(e.g.,AmazonS3,Google cloud storage)offers scalability and cost – effectiveness.

Other Tools and Technologies:

* Programming Languages🡪Python is a popular choice for data science and machine learning tasks, including building recommendation system.
* Frameworks and Libraries🡪Libraries like Scikit-learn,TensorFlow, and PyTorch can be used for developing and deploying machine learning models.
* API Integration🡪APIs can be used to integrate the recommendation system with other services, such as streaming platforms or e-commerce websites.

AI-Driven Matchmaking System components:

* Data Collection and Preprocessing🡪Gathering data from various sources , cleaning and preparing it for model training.
* Model Training🡪Training machine learning models on the collected data to predict user preferences.
* Prediction and Ranking🡪Making predictions about which movies a user might like and ranking them based on their predicted ratings.
* User Interface🡪Presenting the recommended movies to the user in an engaging and user-friendly way.
* Feedback Loop🡪collecting user feedback on the recommendations to continuously improve the system.

1. team members and contributions

R.MOOGAMBIGAI :Documentation and Presentation

A..PRATHIPA: Data Collection and Integration

T.HEMAVARSHINI: Evaluation and Optimization

S.PRIYA: Data Cleaning and EDA

K.PAVITHRA: Feature Engineering and Modeling