Github Link: https://github.com/praveenkumarselvaraj2/Project--Delivering-movie-recommandation-with-an-Ai-driven-matchmaking.git

Delivering movie recommandation with an Ai-driven matchmaking

PHASE-2

- **Problem Statement**

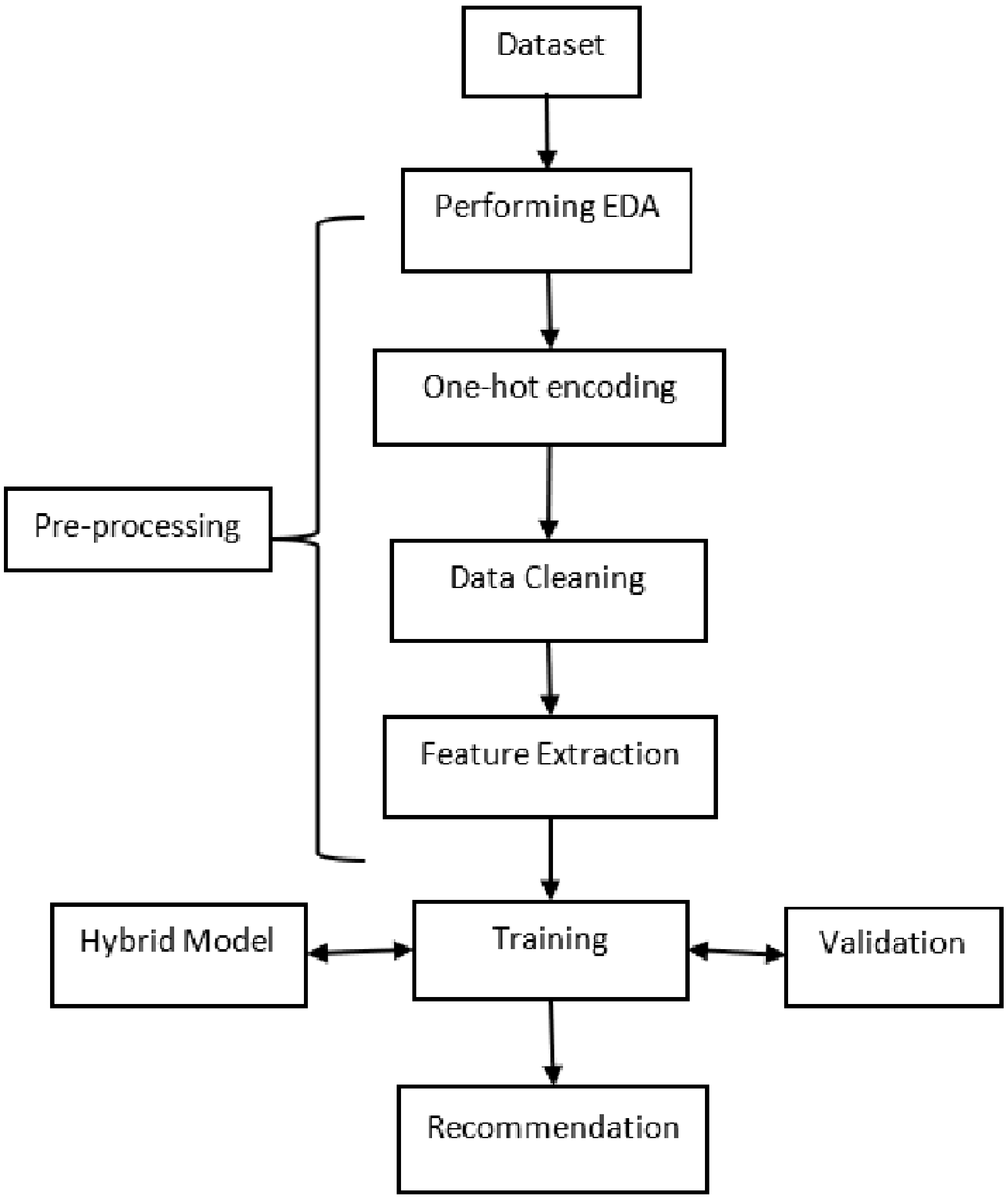
In today's digital age, viewers have access to thousands of movies across various streaming platforms, yet many still struggle to find content that truly resonates with their preferences. Existing recommendation engines typically rely on basic algorithms that consider user ratings or genre preferences. While useful, these methods often fall short in capturing deeper behavioral patterns, evolving tastes, or the social aspect of content consumption, leading to user dissatisfaction or content fatigue.

To address these limitations, we propose an AI-driven movie recommendation system that goes beyond traditional models. This solution will harness advanced machine learning techniques, including natural language processing and behavioral analytics, to better understand individual user profiles. By analyzing viewing history, watch duration, mood indicators, and contextual cues, the system will offer hyper-personalized recommendations that evolve over time.

- **Project Objectives**

  - Deliver personalized movie recommendations using AI and machine learning based on user preferences, behavior, and contextual data.
  - Implement AI-driven user matchmaking to connect individuals with similar or complementary movie tastes for shared viewing experiences.
  - Enhance user engagement through features like shared watchlists, group recommendations, and interactive viewing options.
  - Incorporate mood and sentiment analysis to tailor suggestions based on user emotions and feedback.
  - Ensure a seamless, scalable, and privacy-conscious platform that can handle a growing user base while protecting personal d

- **Flowchart of the Project Workflow**



- **Data Description**
  - Dataset Name: MovieLens 100K
  - Source: GroupLens Research
  - Type of Data: User ratings (explicit)

- Records and Features: 100,000 ratings by 943 users on 1,682 movies
- Target Variable: Movie rating (1–5 stars)
- Static or Dynamic: Static
- Attributes Covered: User ID, Movie ID, Rating, Timestamp, Movie title, Genre
- Dataset Link: https://www.kaggle.com/datasets/dnyaneshyeole/10000-most-popular-english-movies-2023

- **Data Preprocessing**

  - Convert timestamps to readable dates for temporal analysis.
  - Normalize or scale ratings for consistency.
  - Encode categorical features like genre and occupation.
  - Create a user-item interaction matrix for collaborative filtering.
  - Split data into training and testing sets for model evaluation.

- **Exploratory Data Analysis (EDA)**

  - Analyze the distribution of ratings to understand user behavior (e.g., most ratings are 4 or 5 stars).
  - Identify the most and least rated movies to detect popularity trends.
  - Examine user activity levels to find highly active or inactive users.
  - Explore genre-wise rating patterns to see preferences by movie type.
  - Visualize rating trends over time to capture temporal shifts in user preferences.

- **Feature Engineering**

  - Create user-level features like average rating, rating count, or preferred genres.
  - Generate movie-level features such as average rating, total views, and genre encoding.
  - Extract temporal features from timestamps (e.g., rating time of day or day of week).
  - Build interaction features like user-movie rating deviation or user-genre affinity.
  - Encode categorical variables (e.g., genres, occupations) using one-hot or embedding methods.

- **Model Building**

  - Collaborative Filtering: Use techniques like Matrix Factorization (e.g., SVD) or k-Nearest Neighbors (k-NN) to recommend movies based on user-item interactions.
  - Content-Based Filtering: Develop models that recommend movies by analyzing movie metadata (e.g., genres, actors, and directors) and user preferences.
  - Hybrid Models: Combine collaborative and content-based methods to leverage the strengths of both approaches and improve recommendation accuracy.

- Deep Learning: Implement neural networks (e.g., autoencoders) for complex, non-linear relationships between users and movies.
- Evaluation Metrics: Use metrics like RMSE, MAE, Precision, and Recall to evaluate model performance

- **Visualization of Results & Model Insights**

  - Recommendation Accuracy: Visualize predicted vs. actual ratings using heatmaps or ROC/Precision-Recall curves to assess model performance.
  - Top Recommendations: Display top recommended movies using bar charts or word clouds based on user preferences.
  - User-Item Interaction Matrix: Show interactions between users and movies with a matrix to highlight similar user preferences for matchmaking.
  - Matchmaking Insights: Use a network graph to show connections between users with similar movie tastes.
  - Genre Preferences: Represent the distribution of preferred genres using pie charts or stacked bars.
  - Temporal Trends: Plot time series to observe how user preferences evolve over time.
  - Model Evaluation: Compare model performance with boxplots or violin plots of metrics like RMSE and precision.

- **Tools and Technologies Used**

  - Programming Languages: Python (for data manipulation, model building) and R (for statistical analysis, optional).
  - Libraries: Pandas and NumPy for data processing; Scikit-learn for machine learning algorithms.
  - Machine Learning: TensorFlow/Keras for deep learning models; Surprise and LightFM for collaborative and hybrid filtering.
  - Visualization: Matplotlib, Seaborn, and Plotly for data visualization; NetworkX for user-item networks.
  - Database: MySQL/PostgreSQL for structured data storage; MongoDB for unstructured data.
  - Cloud/Deployment: AWS, Google Cloud, or Azure for scalable infrastructure; Flask/Django for web API deployment.

- **Team Members and Contributions**

  1. K Prathishkumar – Data Collection & Preprocessing
  - Dataset Collection: Gather and preprocess the MovieLens dataset or another suitable dataset for recommendations.

- Data Cleaning: Handle missing values, normalize ratings, and clean user/item data.
- Feature Engineering: Extract meaningful features like user preferences, movie genres, and interaction data.
- Data Split: Split the data into training, validation, and test sets for model evaluation.

2. S Praveenkumar – Model Building & Machine Learning

- Collaborative Filtering: Implement collaborative filtering techniques (e.g., Matrix Factorization, SVD) for movie recommendations.
- Content-Based Filtering: Develop models based on movie metadata like genres, actors, and directors.
- Hybrid Model: Combine collaborative and content-based methods to improve recommendation accuracy.
- Evaluation: Evaluate models using metrics like RMSE, Precision, and Recall.

3. V Praveenraj – Matchmaking Algorithm & Social Features

- User Matchmaking: Develop algorithms to match users with similar movie preferences based on collaborative filtering.
- Social Features: Implement features that allow users to share recommendations, create shared watchlists, or interact with other users.
- Interaction Modeling: Explore user-item interaction patterns for better matchmaking suggestions.

4. A.M. Ranjith – Visualization & Deployment

- Result Visualization: Create visualizations for top recommendations, user-item interaction matrices, and matchmaking results using tools like Matplotlib and Plotly.
- Evaluation Insights: Visualize model evaluation results like precision-recall curves, confusion matrices, and ROC curves.
- Deployment: Use Flask/Django to deploy the recommendation system as a web application or API.
- Cloud Integration: Deploy the model on a cloud platform (AWS, Google Cloud) for scalability.