Delivering Personal Movie Recommendation with an AI Driven Matchmaking System

**Student Name:** MADHAN KUMAR D

**Register Number:** 513523106027

**Institution:** ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

**Department:** ELECTRONICS AND COMMUNICATION ENGINEERING

**Date of Submission:** 3/05/2025

**Github Repository Link:**

**https://github.com/MAHESH-0007-ux/TEAM-6.git**

### **Problem Statement**

### *In today's digital era, users are overwhelmed with an ever-expanding library of movies across multiple streaming platforms. The sheer volume of content makes it increasingly difficult for individuals to discover movies that align with their personal preferences. Traditional movie recommendation systems often rely solely on past behavior or general popularity metrics, failing to provide truly personalized and meaningful suggestions. Simultaneously, shared movie experiences—such as watching with friends or like-minded individuals—can enhance user satisfaction, yet most systems do not account for the social aspect of content consumption.This project aims to address two interrelated challenges: (1) delivering highly personalized movie recommendations using advanced AI and data science techniques, and (2) incorporating a matchmaking system that pairs users with others who share similar movie tastes or complementary viewing interests. The goal is not only to improve content discovery but also to enhance user engagement by fostering a community-oriented viewing experience.*

### *Current recommendation engines lack a deep understanding of user preferences beyond explicit ratings or genre-based filtering. They also ignore the potential value of interpersonal compatibility in shared media consumption. By integrating machine learning techniques with user profiling and collaborative filtering, we propose a system that goes beyond recommending just movies—it recommends people to watch them with.*

### *This dual-personalization approach introduces unique challenges in data modeling, privacy considerations, and user experience design. However, it offers a powerful solution to enhance user satisfaction in content consumption and provide a novel layer of interaction through interest-based matchmaking.*

### *In summary, the core problem this project seeks to solve is the lack of intelligent, socially-aware recommendation systems that can simultaneously cater to personal taste and build meaningful user connections. Through the implementation of an AI-driven recommendation and matchmaking engine, we aim to redefine how users interact with entertainment platforms and with each other.*

### **Project Objectives**

### *As the project transitions from the planning stage to practical implementation, the objectives have become more refined and technically focused based on initial data exploration and feasibility assessments. The core aim remains to build a hybrid system that delivers personalized movie recommendations while enabling AI-driven matchmaking based on users' movie preferences and behavioral traits. However, after reviewing the structure and depth of available data, the goals have evolved to better reflect what is achievable and impactful.*

### ***Key Technical Objectives:Develop a Recommendation Engine*** *using a hybrid approach that combines collaborative filtering and content-based methods to generate personalized movie suggestions.****Build a User Similarity Model*** *that profiles users based on their ratings, genre preferences, and behavioral attributes to identify compatibility with other users.*

### ***Integrate a Matchmaking Component*** *that utilizes clustering or similarity scoring (e.g., cosine similarity or embeddings from a neural network) to suggest compatible users for shared movie experiences.*

### ***Design a Scalable Pipeline*** *for data ingestion, model training, and real-time inference using Python-based tools and libraries such as Scikit-learn, Pandas, and possibly TensorFlow or PyTorch.*

### ***Create a User Interface (optional)*** *that visualizes recommendations and suggested matches, possibly using Streamlit or Flask.*

### ***Model Goals:***

### ***Accuracy****: Optimize for precision and recall in both recommendations and user matches, using metrics like RMSE, precision@k, and NDCG.*

### ***Interpretability****: Ensure the system provides understandable outputs—why a movie or person was recommended—through explainable AI techniques.*

### ***Real-world Applicability****: Design the models to be deployable and practical for integration into a real entertainment platform or social app.*

### ***Evolving Goals:***

### *During data exploration, it became clear that the diversity of user preferences and the sparsity of rating data could pose challenges. As a result, the focus has shifted slightly from building a purely predictive model to one that emphasizes interpretability and user satisfaction. Additionally, the importance of social factors in matchmaking led to the inclusion of user clustering and potential personality mapping using quiz or metadata input.*

### **Flowchart of the Project Workflow**

### *The workflow of this AI-driven personalized movie recommendation and matchmaking system follows a structured, multi-phase process that begins with data collection and ends in real-world deployment and feedback optimization. The initial phase involves gathering relevant data from reliable sources. This includes movie metadata—such as genres, cast, and keywords—from public APIs like TMDB or datasets like MovieLens, along with user-specific data, including ratings, viewing history, and optionally demographic or questionnaire-based personality inputs. Once collected, the data is preprocessed to ensure quality and consistency. This step includes handling missing values, encoding categorical variables (e.g., genres and tags), normalizing rating values, and transforming the data into a format suitable for modeling.After preprocessing, the next step is exploratory data analysis (EDA), which helps uncover insights about user preferences, genre popularity, and rating patterns. Visualization tools and statistical analysis are used to understand data distributions, correlations, and potential outliers. These insights guide the model-building phase, where two primary components are developed: the recommendation engine and the matchmaking module. The recommendation engine uses a hybrid approach, combining collaborative filtering (e.g., matrix factorization) and content-based filtering to suggest personalized movies to each user. Meanwhile, the matchmaking system leverages clustering or similarity measures to identify users with shared or complementary viewing tastes, forming the basis for intelligent pairing.*

### *Once the models are built, they are rigorously evaluated. Recommendation performance is assessed using metrics such as RMSE, precision@k, and NDCG, while the matchmaking model is evaluated through clustering metrics like silhouette score or user satisfaction proxies. Following evaluation, the models are integrated into a user-facing application, potentially using tools like Streamlit or Flask to display movie suggestions and compatible user matches in an intuitive interface. Finally, a feedback loop is incorporated, enabling the system to collect ongoing user interactions and improve model accuracy over time. This end-to-end workflow ensures both personalized content delivery and an engaging, socially-aware user experience.*

### 

### **Data Description**

### *For this project, we utilized the* ***MovieLens 1M Dataset****, a widely recognized benchmark dataset for building and testing recommendation systems. This dataset was sourced from the* ***GroupLens research group*** *and is freely available through platforms like* ***Kaggle*** *and the* ***official GroupLens website****. Additionally, we supplemented this data with movie metadata from* ***The Movie Database (TMDB) API****, which provided enriched information such as movie genres, cast, crew, keywords, and plot overviews to enhance content-based recommendation modeling.The data used is primarily* ***structured****, consisting of numerical and categorical variables such as user IDs, movie IDs, rating values, timestamps, and genre labels. The core dataset includes approximately* ***1 million ratings*** *from* ***6,000 users*** *across* ***4,000 movies****, with each user having rated at least 20 movies. The TMDB data added further dimensionality with text-based features like movie descriptions, which we tokenized and embedded for advanced modeling.*

### *In total, the dataset comprises several key components:*

### ***Ratings data****: User ID, Movie ID, Rating (1–5 scale), Timestamp*

### *.****Movies metadata****: Title, Genre(s), Release Year, Cast/Crew, Keywords.*

### ***User data****: Limited demographic information (age, gender, occupation), with the option to simulate additional data like personality scores for matchmaking.*

### *The dataset is considered* ***static****, meaning it represents a snapshot in time and does not update dynamically. However, it can be extended with real-time feedback mechanisms during model deployment to simulate dynamic updates and improve personalization over time.*

### *Since the core recommendation task is* ***unsupervised*** *(no explicit target variable), the focus is on learning latent relationships between users and items. However, if we frame part of the system as a prediction task—for example, predicting a rating or match score—then the* ***target variable*** *would be the user’s rating or a compatibility score derived from user profiles.*

### *Overall, the richness and structure of the MovieLens + TMDB dataset combination make it well-suited for both recommendation modeling and user matchmaking, supporting a variety of machine learning approaches.*

### **5. Data Preprocessing**

*Effective data preprocessing was essential to ensure the quality and reliability of the recommendation and matchmaking models. The process began with importing the MovieLens 1M dataset along with the TMDB metadata, both of which were merged using movie IDs as a common key. The first step was handling* ***missing values****, particularly in the TMDB metadata where some entries (e.g., keywords or cast) were incomplete. For essential numerical features like ratings, no missing data was found, while for non-critical text-based fields, missing values were either* ***imputed using placeholders*** *like "Unknown" or dropped when not vital for modeling.*

*Next, we checked for and removed* ***duplicate records****, especially in the merged dataset where redundant movie entries could occur. Using Pandas, the .duplicated() function was applied, and duplicates were dropped after ensuring they did not contain distinct metadata. In terms of* ***outlier detection****, rating values were confined to a 1–5 scale, so no numerical anomalies were present. However, outliers in user behavior (e.g., users who rated thousands of movies or only a handful) were flagged. Users with fewer than 20 ratings were removed to ensure meaningful preference modeling.*

*Data types were carefully reviewed and corrected for consistency. Timestamps were converted to human-readable datetime formats for potential time-based analysis, while numeric columns were cast to appropriate types (e.g., int32, float64) to optimize memory usage. For* ***categorical features****, such as movie genres,* ***multi-label binarization*** *(a form of one-hot encoding) was applied, converting each genre into its own binary feature. User demographic features like gender and occupation were also encoded using* ***label encoding****.*

*To ensure feature comparability, especially for clustering and similarity models,* ***normalization*** *was applied to continuous features like average user rating and movie popularity using* ***min-max scaling****. Text features (movie overviews, keywords) were tokenized and converted into vector embeddings using TF-IDF or pretrained models like Word2Vec.*

*Each transformation was documented in code using clear comments and supported by markdown cells explaining the logic, ensuring that the entire preprocessing pipeline was both transparent and reproducible.*

### **6. Exploratory Data Analysis (EDA)**

*A comprehensive Exploratory Data Analysis (EDA) was conducted to understand the underlying structure and patterns within the MovieLens and TMDB datasets. The process began with* ***univariate analysis****, which explored the distribution of individual features. Using histograms, we examined the distribution of user ratings, revealing a strong skew toward higher ratings—particularly 4.0 and 5.0—indicating a tendency for users to rate movies they enjoyed more frequently.* ***Boxplots*** *were used to inspect the distribution of ratings by genre, showing, for example, that drama and thriller genres had more varied ratings, while comedy and action skewed toward higher averages.* ***Countplots*** *were used to visualize the frequency of movies across genres, confirming that drama and comedy were the most prevalent categories.*

*In the* ***bivariate and multivariate analysis****, a* ***correlation matrix*** *was generated for numeric features like average rating, number of ratings, release year, and movie popularity. While direct linear relationships were limited, we found that the number of ratings and average rating had a mild positive correlation, implying that popular movies often received higher scores.* ***Pairplots and scatterplots*** *helped visualize relationships between user age groups and their preferred genres, with younger users tending to favor action and sci-fi, while older users rated drama and documentary genres more frequently.* ***Grouped bar plots*** *revealed that male users rated sci-fi and action more frequently, while female users leaned toward romance and family genres.In terms of* ***relationship with the target variable****, we examined how features such as genre, release year, and user demographics influenced rating behavior. Older movies tended to have more polarized ratings, likely due to either classic status or outdated themes, while newer releases clustered around moderate-to-high ratings.*

*From these analyses, several key* ***insights*** *emerged. Genre, user demographics (especially age and gender), and popularity metrics significantly influence movie ratings. These features were flagged as highly relevant for the recommendation engine. For matchmaking, user rating patterns and genre preferences were identified as crucial indicators of compatibility. The EDA phase not only confirmed the richness of the data but also guided feature selection for model training.*

**7. Feature Engineering**

*Feature engineering played a critical role in improving the performance and interpretability of the recommendation and matchmaking models. Based on insights from exploratory data analysis (EDA), we created several new features to better capture user preferences, movie characteristics, and temporal aspects that could impact recommendations.*

*One of the first steps in feature engineering was to* ***create new features*** *derived from existing data. For instance, from the movie release date, we* ***extracted the decade*** *(e.g., "2000s," "2010s") and* ***movie age*** *(current year minus release year) to identify trends in user preferences over time. This addition allowed the model to differentiate between new and classic films, which could influence how a user rates movies. Additionally, we* ***created a genre diversity score*** *by counting the number of genres associated with each movie. This metric helped capture the complexity of movies with multiple genres, which might appeal to different types of users.*

*We also employed* ***binning*** *techniques to group user ratings into broader categories. Instead of treating ratings as continuous numerical values, we categorized them into bins such as "Low", "Medium", and "High" ratings (e.g., 1-2, 3-4, 5). This transformation simplified the model’s learning task and reduced noise in the rating scale.*

*Another transformation involved* ***splitting columns****, specifically the "timestamp" field, which we broke down into* ***day of the week****,* ***hour of the day****, and* ***month*** *to assess temporal patterns in user activity. Users may be more active at specific times, which could be valuable for recommending movies at optimal times.*

*To improve the performance of the machine learning models, we also considered* ***dimensionality reduction*** *techniques, such as* ***Principal Component Analysis (PCA)****. This was applied to high-dimensional features like movie metadata (e.g., keywords and cast) to reduce the number of features while retaining important information. After performing PCA, we retained the top components that explained the most variance in the data.*

*Each of these engineered features—whether they provided new insights or reduced dimensionality—was carefully justified based on how it aligned with the goals of enhancing recommendation accuracy and matchmaking quality. These features not only captured domain-specific knowledge but also helped streamline the modeling process, improving overall model performance.*

### **8. Model Building**

### *In the model-building phase, we focused on developing two key models:* ***Collaborative Filtering*** *and* ***Random Forest Regressor****, to solve the problem of personalized movie recommendations and user matchmaking. These models were selected based on the nature of the data (ratings and user preferences) and the type of task (recommendation and regression).The* ***Collaborative Filtering*** *model, specifically* ***Matrix Factorization****, was chosen because of its effectiveness in handling sparse user-item rating matrices and capturing latent user-item relationships. This model leverages the* ***User-Item interaction matrix*** *and learns to predict missing values (ratings) based on similar users or items. Given that movie ratings are often sparse and user preferences are implicit (e.g., users might not rate all the movies they watch), collaborative filtering is an ideal approach to recommend items based on similarity in user behaviors. For implementation, we used* ***SVD (Singular Value Decomposition)****, a matrix factorization technique, to decompose the user-item matrix into lower-dimensional matrices that capture the latent features of users and items.The second model we implemented was the* ***Random Forest Regressor****, a robust ensemble learning method. This model was selected because it handles non-linear relationships well and is resistant to overfitting, making it suitable for a diverse set of features such as genre, release year, and movie popularity. The* ***Random Forest*** *is an ensemble of decision trees that provides high accuracy through averaging multiple predictions, making it ideal for regression tasks where we're predicting continuous values, such as movie ratings. It allows us to capture complex patterns in the data, like interactions between movie attributes and user demographics.*

### *To evaluate the models, the data was split into* ***training (80%)*** *and* ***testing (20%)*** *sets, with stratification applied to ensure a balanced distribution of ratings across users. The* ***Collaborative Filtering*** *model was evaluated using* ***Root Mean Squared Error (RMSE)*** *and* ***Mean Absolute Error (MAE)****, as it predicted continuous ratings. The* ***Random Forest*** *model was assessed using* ***R² score*** *and* ***RMSE*** *for its ability to predict rating values effectively.*

### *Both models were trained and fine-tuned, and initial performance metrics showed that* ***Collaborative Filtering*** *performed well in terms of capturing latent preferences, while the* ***Random Forest*** *model helped refine predictions based on a richer set of features.*

### **9. Visualization of Results & Model Insights**

*[Visualization plays a crucial role in interpreting the performance and behavior of machine learning models. In this project, several plots and charts were used to analyze both the* ***Collaborative Filtering*** *and* ***Random Forest*** *models and to draw meaningful insights about their predictive capabilities.*

*For the* ***Collaborative Filtering*** *model, we used a* ***residual plot*** *to examine the difference between predicted and actual ratings. The residuals were relatively evenly distributed, suggesting that the model was neither overfitting nor underfitting the data. We also plotted a* ***learning curve*** *showing the model's training and validation errors as a function of the number of iterations. This helped assess the convergence of the model and ensured it was learning efficiently.*

*For the* ***Random Forest Regressor****, a* ***feature importance plot*** *was created to identify which features contributed most to the model's predictions. This plot revealed that* ***user demographics (age and gender)*** *and* ***movie genres*** *were the most influential in predicting movie ratings. Genre, in particular, had a high importance score, reflecting how user preferences in specific genres could drive movie choices. The* ***R² score plot*** *provided a visual comparison of how well the model fit the data, and residual plots revealed that the model's errors were relatively consistent across different ratings, indicating good model performance.*

*Additionally,* ***performance comparison plots*** *were generated to contrast the two models’ effectiveness. The* ***RMSE*** *for the Collaborative Filtering model was lower, indicating that it outperformed the Random Forest Regressor in terms of predicting ratings with fewer errors. However, the Random Forest model, with its broader set of features, provided more nuanced insights into user preferences based on demographic and movie metadata.*

*A* ***confusion matrix*** *was not applicable in this regression problem, but for future classification tasks (e.g., movie categorization or user recommendation), this matrix could be useful to evaluate the true positives, false positives, true negatives, and false negatives.*

*In summary, these visualizations not only helped in assessing model accuracy but also provided valuable insights into the features influencing predictions, such as the importance of user demographics and genre preferences in the recommendation system.*

**10. Tools and Technologies Used**

*For this project, several tools and technologies were leveraged to efficiently handle data preprocessing, modeling, evaluation, and visualization tasks.*

*The primary* ***programming language*** *used for the entire project was* ***Python****, due to its versatility, extensive libraries, and support for machine learning. Python is widely used in the data science community and offers a broad range of tools that are well-suited for tasks like data manipulation, modeling, and visualization.*

*The project was primarily developed and executed using the* ***Google Colab*** *notebook environment, which provided a cloud-based platform for Python coding and real-time collaboration. Google Colab offered easy access to GPUs for model training, which was particularly useful when dealing with larger datasets and computationally intensive models. Additionally,* ***Jupyter Notebook*** *was used for documentation and exploratory analysis, allowing for a seamless integration of code, markdown explanations, and visualizations in one document.*

*In terms of* ***libraries****, a range of Python packages were employed:*

* ***Pandas*** *and* ***NumPy*** *were used for data manipulation and numerical computations. These libraries enabled easy handling of the MovieLens and TMDB datasets, including data cleaning, transformation, and feature engineering.*
* ***Scikit-learn*** *was the go-to library for implementing machine learning models, such as* ***Collaborative Filtering*** *(via SVD) and* ***Random Forest Regressor****. Scikit-learn also provided tools for data splitting, evaluation metrics, and model validation.*
* ***Matplotlib*** *and* ***Seaborn*** *were used extensively for* ***visualizations****, enabling the creation of plots like histograms, boxplots, and scatterplots to explore the data and evaluate model performance.*
* ***XGBoost*** *was explored for potential improvements in the Random Forest model, though it was not ultimately used in the final model due to its higher complexity in this context.*

*For* ***visualizations*** *of results and comparisons,* ***Matplotlib*** *and* ***Seaborn*** *were the primary tools. However,* ***Plotly*** *was also used to create interactive visualizations of performance metrics and feature importance, allowing for more dynamic presentations of the data and model insights.*

*Overall, these tools formed the backbone of the project, ensuring a smooth workflow from data exploration and model building to final evaluations and visual insights.*

**11. Team Members and Contributions**

*1.MADHAN KUMAR D – DATA CLEANING/TEAM LEADER*

*2.MOHANAPRIYA B.K – FEATURE ENGINEERING*

*3.MOHANAPRIYA S – MODEL DEVELOPMENT*

*4.LOKESHWARI M – EDA*

*5.MAHESH KUMAR V – DOCUMENTATION AND REPORTING*