**Student Name:** KANISHKA S

**Register Number:** 613023104045

**Institution:** Vivekananda college of Technology for Women

**Department:** B.E Computer science and Engineering

**Date of Submission:** 30.04.2025

**GitHub Repository Link:** https://github.com/Kanishka200511/Project.git

**1.Problem Statemen****t**

* Project aims to address this issue by delivering personalized movie recommendations through an AI-driven matchmaking system that leverages deep user insights and content characteristics. By analyzing user behavior, viewing history, preferences, and movie attributes, our system intelligently matches users with movies they are most likely to enjoy.
* **Type** **of** **Problem:** This is primarily a classification and ranking problem, with elements of clustering for user segmentation. The model classifies movies based on predicted user preferences and ranks them accordingly to recommend the most suitable content*.*
* **Impact** **and** **Relevance:** Solving this problem significantly enhances the user experience on digital streaming platforms by reducing search time, increasing user satisfaction, and promoting content.

### **2. Project Objectives**

**Technical Objectives**:

* Preprocess and structure movie and user data effectively.
* Build and compare collaborative filtering, content-based, and hybrid recommendation models.
* Optimize for high accuracy (target: 80%+), personalization, and real-world usability.
* Ensure model outputs are interpretable and scalable for practical applications.

**Evolved Goals:**

* Initial focus was on collaborative filtering, but after exploring the dataset, we now aim to use a hybrid approach to improve performance and handle cold-start problems.

### **3. Flowchart of the Project Workflow**

**1. Start:**

* Project Initialization

**2. User Registration & Profile Setup:**

* User Collect preferences, demographics, and interests

**3. Data Collection**

* Movie metadata (genre, cast, ratings)
* User behavior (watch history, ratings, likes)

**4. Data Preprocessing**

* Clean and normalize data
* Handle missing values
* Encode categorical variables

**5. Feature Engineering**

* Extract user and movie features
* Create user-item interaction matrix

**6. Model Selection & Training**

* Choose recommendation algorithms (e.g., collaborative filtering, content-based filtering, hybrid models)

**7. AI-Driven Matchmaking Engine**

* Match users with movies based on trained model
* Incorporate user-to-user similarity for enhanced suggestions

**8. Recommendation Generation**

* Generate top-N personalized recommendations

Adjust

**9. Feedback Loop**

* based on real-time feedback
* Gather user responses to improve model accuracy
* Update user profiles and model parameters

**10. Display & Interface**

* Deliver recommendations via user-friendly UI

**11. End**

**4. Data Description**

**Dataset Name & Origin:**

* + Movie Lens 100K dataset from Group Lens Research (Kaggle)
  + Tad API (The Movie Database) for additional metadata like poster images, cast, and genre details

**Type of Data:**

* + Structured data: ratings, user IDs, movie IDs, genres
  + Unstructured data: movie overviews and poster images (text & image)
  + Semi-structured data: API responses in JSON format

**Number of Records and Features:**

* Movie Lens 100K: 100,000+ ratings
* Users: ~900
* Movies: ~1,700
  + Features: user ID, movie ID, rating, timestamp, genre, title, overview
  + Static or Dynamic Dataset: Movie Lens is statute API is dynamic
  + Target Variable: In supervised tasks (e.g., predicting ratings), the target variable is the rating (scale 1–5)

### 

### **5. Data Preprocessing**

### **1. Missing Values:** No missing values in the core dataset. Future metadata will be cleaned using mean/mode imputation or row removal.

### **2. Duplicates:** No duplicate entries found.

### **3. Outliers**: Ratings are bounded (1–5). Metadata outliers (e.g., release year) were capped using IQR limits.

### **4. Data Types:** UserID and MovieID converted to integers, Rating to float, Timestamp to datetime.

### **5. Encoding:** Genres one-hot encoded; optional demographic fields encoded via label/one-hot encoding.

### **6. Scaling:** Metadata (e.g., year) standardized using Z-score normalization.

### **7. Code Sample:**

### ratings['timestamp'] = pd.to\_datetime(ratings['timestamp'], unit='s')

### genres = movies['genres'].str.get\_dummies(sep='|')

### movies = pd.concat([movies, genres], axis=1)

### df = pd.merge(ratings, movies, on='movieId')

### scaler = StandardScaler()

### df['year'] = scaler.fit\_transform(df[['year']])

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

**Univariate Analysis**

* **User Age:** Most users are aged 18–35, suggesting a youth-skewed audience
* **Gender**: Males slightly outnumber females; some users chose “Other” or preferred not to specify.
* **Genres:** 'Drama', 'Action', and 'Comedy' are the most frequent.
* **Release Years:** Most movies were released between 2000–2020.
* **Ratings:** Ratings are right-skewed, clustering between 3.0 and 4.5.

**Bivariate/Multivariate Analysis**

**Correlations:** Positive correlation between average\_rating and number\_of\_ratings. Weak link between release\_year and ratings.

**User Age vs. Ratings:** Younger users tend to give higher ratings.

**Genre vs. Popularity:** Action and comedy show wide rating variance.

**Gender Preferences:** Males favor action and sci-fi; females lean toward romance and drama.

**Insights Summary**

* Age, gender, and genre preferences are key influencers.
* Movie popularity and user rating patterns support collaborative filtering.
* Recent, well-rated movies with broad appeal should be prioritized in recommendation strategies.

### **7. Feature Engineering**

### **User Features**

### **Age Binning:** Categorized users into age groups (Teen, Adult, etc.).

### **Gender Encoding:** One-hot encoded for model use.

### **Movie Features**

### **Genre Vectorization:** Multi-hot encoded for model compatibility.

### **Decade of Release:** Added temporal grouping.

### **8. Model Building**

### **Models Use:**

### Logistic Regression

### Random Forest Classify

### **Why These Models Were Chosen:**

### Logistic Regression was used as a baseline model due to its simplicity

### Random forest it effectively manages diverse user behaviors

### **Data** **Split:** The dataset—comprising user profiles, movie metadata and interaction

### **Evaluation**:

### Accuracy – Overall correctness of predictions

### Precision – Correctness of positive (liked movie) predictions

### Logistic Regression: 0.78

### Random Forest: 0.8

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

### **Confusion** **Matrix:** Random Forest showed fewer false negatives and more true positives than Logistic Regression.

### **ROC Curve & AUC:** Logistic Regression (~0.84), Random Forest (~0.92).

### **Feature Importance (Random Forest):** genre preferences, past ratings, movie popularity.

### **Model Comparison:** Random Forest outperformed across all metrics, especially F1-Score (0.86 vs. 0.78)

### **Residual Analysis:** Logistic Regression struggled with edge cases; Random Forest was more robust.

### **10. Tools and Technologies Used**

* **Programming Language:** Python

### **IDE:** Google Collab

### **Libraries:** pandas, NumPy, scikit-learn, matplotlib, seaborn, Boost (optional)

### **Visualization:** Portly (interactive), Tableau (optional dashboards)

* **Purpose:** Used for data processing, model training, evaluation, and visual analysis in building the AI-driven movie recommendation system**.**

### 

### **11. Team Members and Contributions**

* **Akalya** **S**

Responsibilities: Team Lead, Documentation & Reporting

* **Dharshini V**

Responsibilities: Data Cleaning, Feature Engineering

* **Kaniga** **S**

Responsibilities: Exploratory Data Analysis (EDA), Visualization

* **Kanishka S**

Responsibilities: Model Development, Evaluation

* **Mahalakshmi D**

Responsibilities: Support in EDA and Model Tuning