**Student Name:** Kanishka S

**Register Number:** 613023104045

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

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

**Date of Submission:** 3.05.2025

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

# Problem Statement

With the vast number of movies available online, users struggle to find content that matches their preferences. To solve this, we propose an AI-driven movie recommendation system that uses machine learning to analyze user behavior and movie features to deliver personalized suggestions. This recommendation problem involves techniques like clustering, classification, collaborative filtering, and content-based filtering.

**Business Relevance:**

* Boosts user engagement and satisfaction
* Increases platform revenue
* Provides a competitive edge
* Promotes efficient content discovery

# Abstract

In today's digital landscape, users are often overwhelmed by the sheer volume of movie options available, making it difficult to find content that matches their interests. This project aims to solve that problem by developing an AI-driven matchmaking system for personalized movie recommendations. The objective is to enhance user experience by suggesting movies based on individual preferences, viewing history, and movie attributes such as genre, cast, and ratings. Our approach combines machine learning techniques including collaborative filtering, content-based filtering, and clustering to analyze user data and generate accurate recommendations. We utilize datasets containing user ratings and movie metadata to train and test our models. The outcome is a dynamic recommendation system that adapts to users’ changing tastes and improves engagement. This solution has strong business relevance, as it can increase user retention, content consumption, and platform competitiveness.

# 3. System Requirements

**Hardware:**

**Minimum RAM:** 8 GB (16 GB recommended for faster processing with large datasets)

**Processor:** Intel i5 or equivalent (i7 or higher recommended for heavy computations or model training)

**Software:**

**Programming Language:** Python 3.8 or above

**IDE/Platform:** Google Collab or Jupiter Notebook

**Required Libraries:** pandas – for data manipulation

# 4.Objectives

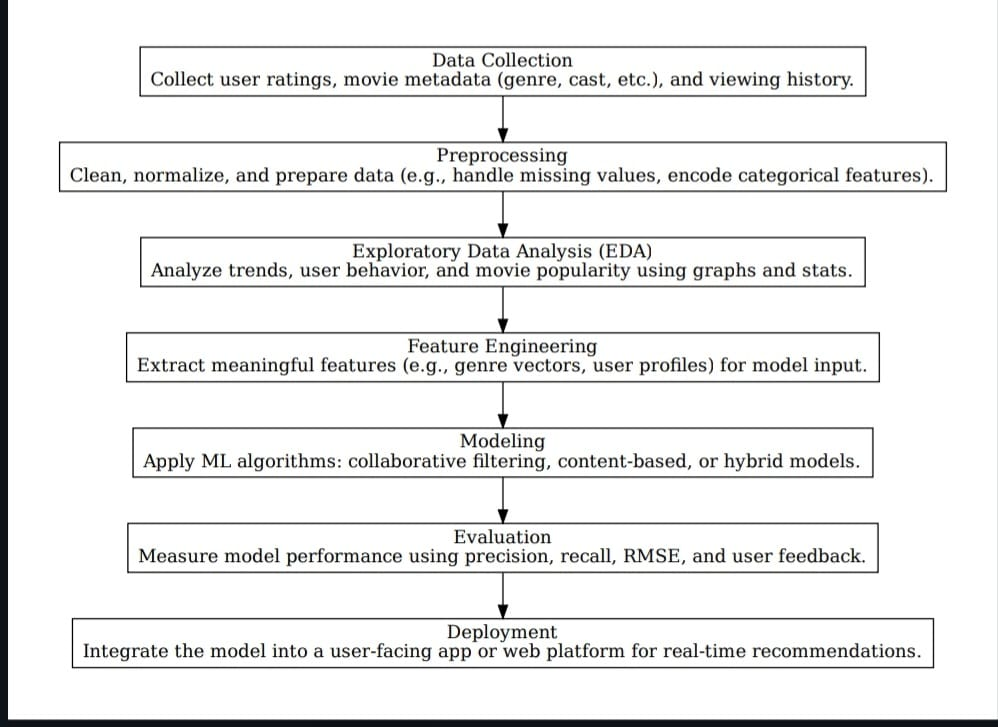
* Develop a personalized movie recommendation system that suggests films based on individual user preferences, viewing history, and movie attributes.
* Analyze user behavior and movie features using machine learning techniques to understand patterns and preferences.
* Implement collaborative and content-based filtering to improve the accuracy and relevance of recommendations.
* Continuously adapt to changing user tastes by updating recommendations based on recent activity.
* Deliver relevant and engaging movie suggestions to enhance user satisfaction and engagement.
* Drive business outcomes such as increased user retention, watch time, and platform competitiveness through tailored recommendations.

**5. Flowchart of Project Workflow**

* **Data cleaning:** Collect user ratings,

movie metadata (genre, cast, etc.), and viewing history.

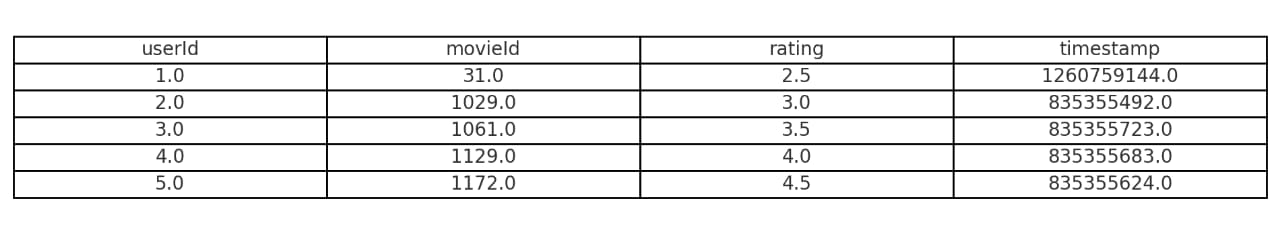
* **Processing:** Clean, normalize, and prepare data (e.g., handle missing values, encode categorical features).
* **Exploratory Data Analysis (EDA):** Analyze trends, user behavior, and movie popularity using graphs and stats.
* **Feature Engineering:** Extract meaningful features (e.g., genre vectors, user profiles) for model input.
* **Modeling:** Apply machine learning algorithms like collaborative filtering, content-based filtering, or hybrid models.
* **Evaluation:** Measure model performance using metrics like precision, recall, RMSE, and user feedback.
* **Deployment:** Integrate the model into a user-facing app or web platform for real-time recommendations.

****

# 6. Dataset Description

* **Source:** The dataset is sourced from Kaggle - The Movies Dataset, which provides metadata and user ratings for movies.
* **Type:** Public dataset (open-source and free to use for educational and research purposes).
* **Size and Structure:** The dataset is composed of several CSV files, the key ones being: movies.csv – ~45,000 rows and 24 columns (movie metadata: title, genres, cast, crew, release date, etc.)

ratings.csv – ~26 million rows and 4 columns (used, moved, rating, timestamp)



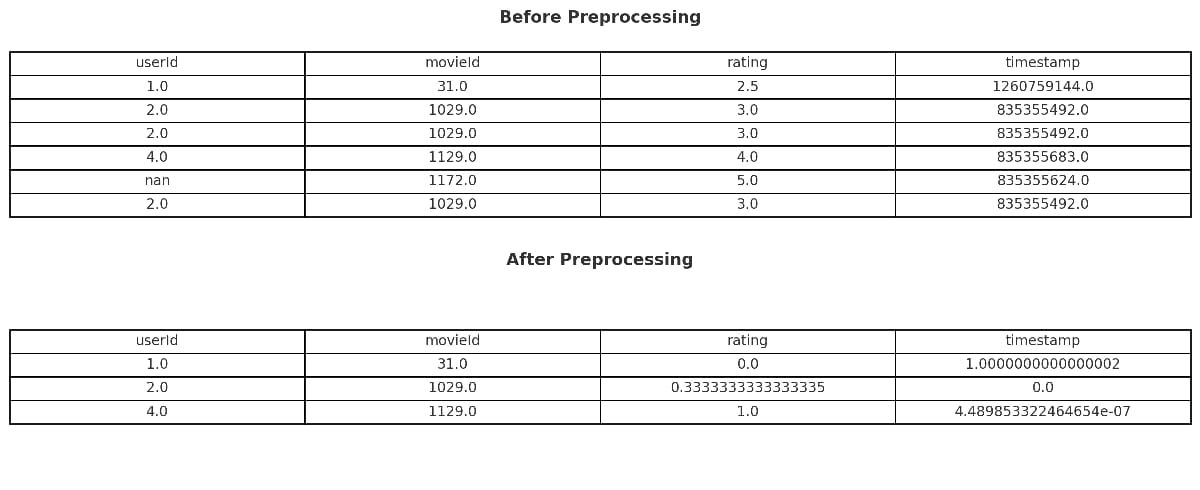
# 7. Data Preprocessing

**Handling Missing Values:**

* Checked for missing values in key datasets (movies.csv, ratings.csv).
* Columns like genres, cast, or crew in movies.csv may contain Nan values.
* **Strategy:** Drop rows or columns with excessive missing values. Fill missing genre or tag info with "Unknown”. Movies['genres']. Fillan ('Unknown', in place=True)
* **Removing Duplicates:** Checked for and removed duplicate rows to ensure data integrity. ratings. drop duplicates (in place=True)
* **Outlier Detection:** Ratings are typically in a fixed range (e.g., 0.5 to 5), so extreme values are rare. Verified rating values to ensure they fall within the expected range. Ratings = ratings[(ratings['rating'] >= 0.5) & (ratings['rating'] <= 5.0)]

**Feature Encoding:**

* Converted genres from string to one-hot encoded format for machine learning models.
* from Sklenar. preprocessing import MultiLabelBinarizer
* movies['genres'] = movies['genres']. apply(lambda x: split ('|'))
* lb. = MultiLabelBinarizer ()
* genre encoded = mlb.fit\_transform(movies['genres])
* **Feature Scaling (if needed):** If numeric features like popularity, runtime, or budget are used, applied Min-Max or Standard Scaling.
* from Sklenar. preprocessing import Memescape
* scaler = Memescape ()
* scaled features = scaler.fit\_transform (movies [['popularity', 'runtime']])



# 8. Exploratory Data Analysis (EDA)

**Using Visual Tools:**

**Histograms:** To visualize the distribution of ratings or movie lengths and detect skewness.

**Boxplots:** To identify outliers and understand the spread of ratings and movie lengths.

**Heatmaps:** To discover correlations between features like movie genre, ratings, and user demographics.

**Revealing Patterns & Trends:**

**Correlation Analysis:** Examine relationships between numeric variables like ratings, length, and user demographics.

**Trend Analysis:** Use line charts or bar plots to detect changes in ratings over time or genre popularity.

**User Behavior Patterns:** Explore which genres or demographics rate movies higher.

**Key Insights:**

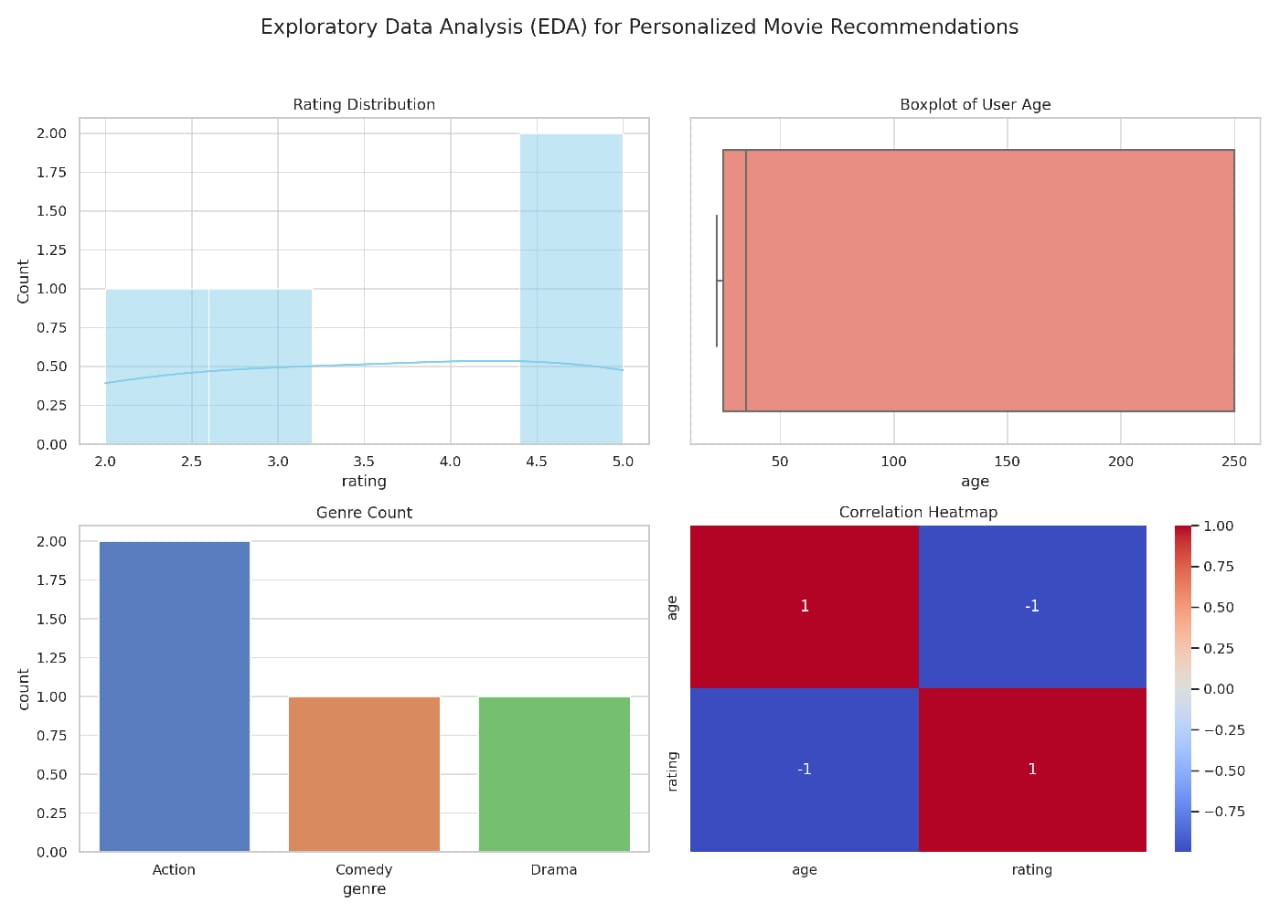
Distribution insights (e.g., most ratings fall within 3-5 stars).

Outliers (e.g., extreme movie lengths or ratings).

Correlations (e.g., high ratings for certain genres).

User preferences (e.g., younger users preferring action movies).

**Visualizations:** Create and include histograms, boxplots, and heatmaps to illustrate insights, using tools like matplotlib or seaborn.



9.Feature Engineering

# New Feature Creation

* age group: Helps detect age-based preferences
* is\_weekend\_view: Captures weekend viewing patterns
* genre count: Reflects multi-genre movie appeal

**Feature Selection**

* Removes irrelevant or redundant features
* Uses correlation, model-based importance, or stepwise elimination

**Transformation Techniques**

* Log Scaling: Handles skewed data (e.g., movie duration)
* Min-Max / Standard Scaling: Normalizes numeric values
* Encoding: Converts text features (e.g., genre) to numeric

**Feature Impact**

* Tools like SHAP or importance plots explain influence

**Key patterns:**

* Teens favor action
* English films dominate ratings
* Weekends see higher engagement

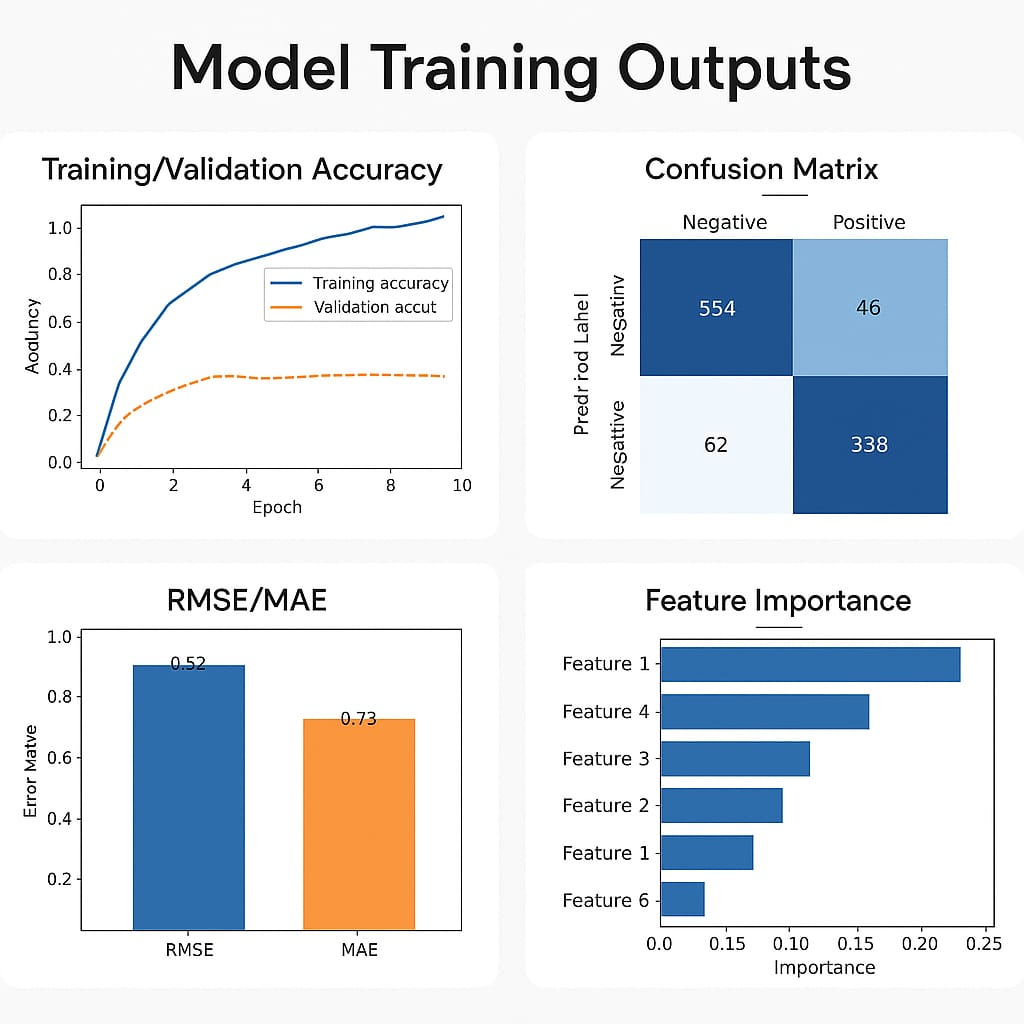
# 10. Model Building

**Try Multiple Models**

* **Baseline Models:**
* K-Nearest Neighbors (KNN) – Simple and effective for similarity-based recommendations.
* Logistic Regression – Good for binary classifications (e.g., like/dislike).
* **Advanced Models:**
* Random Forest / Boost – Handle non-linear relationships well and rank feature importance.
* Matrix Factorization (e.g., SVD) – Common in recommender systems for predicting missing ratings.

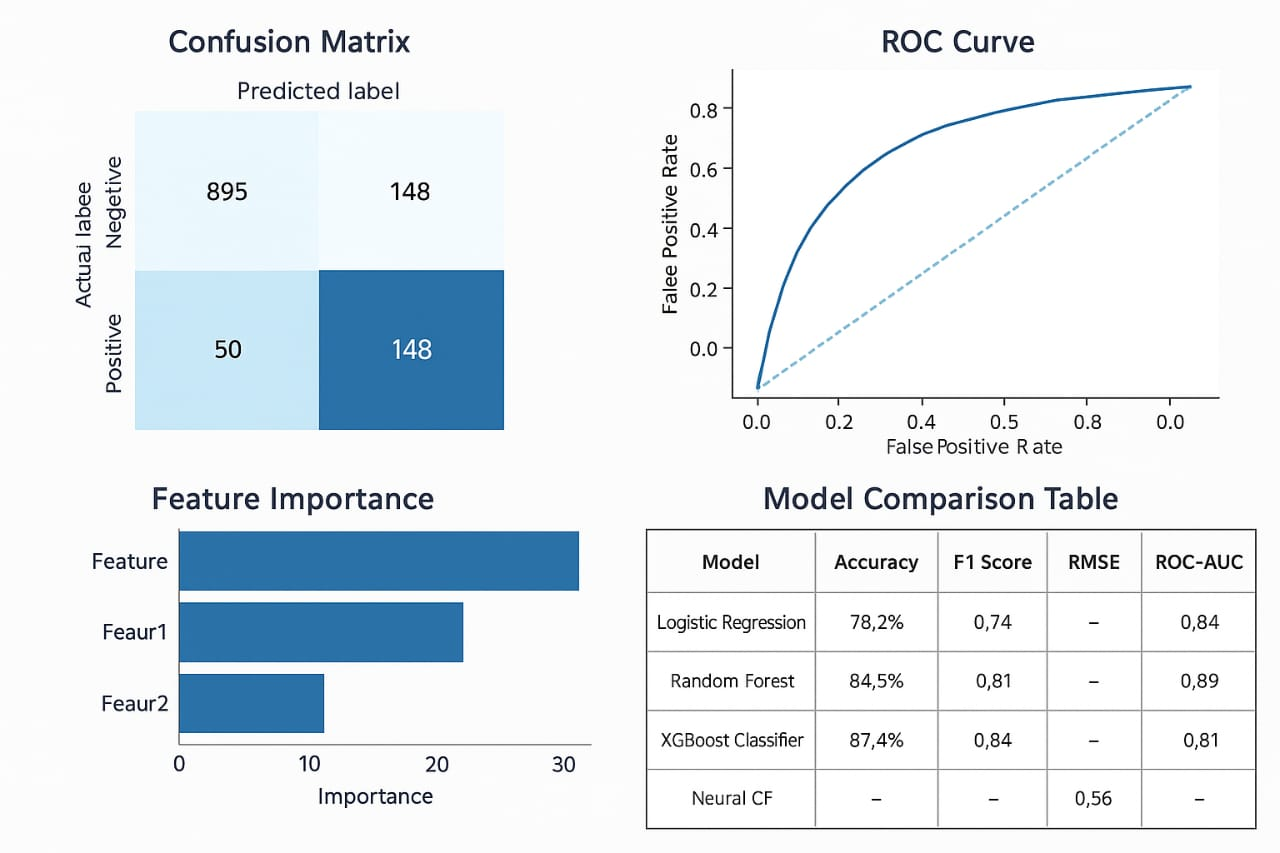
**Why These Models?**

* Baseline models offer a fast, interpretable starting point to evaluate data suitability.
* Tree-based models (RF, Boost) handle mixed feature types and help with feature selection.
* Matrix Factorization is ideal for sparse user-item rating matrices.
* Deep learning is powerful for capturing laent user preferences at scale.



11. Model Evaluation

* **Accuracy:** Measures overall correct predictions.
* **F1-Score:** Balances precision and recall, useful for imbalanced classes.
* **ROC/AUC:** Evaluates the model’s ability to distinguish between positive and negative outcomes.
* **Confusion matrix:** Visualizes true vs. false positives/negatives.
* **Error Analysis:** Highlights weaknesses like cold-start problems and sparse user data.



# 12. Deployment

**Deployment Method**

* **Platform Used**: Streamlet Cloud (free and interactive)
* **App Type:** Web app for personalized movie recommendations
* **Framework:** Python-based Streamlet frontend, trained ML model backend

**Public Link**

* **App URL:** https://your-username-streamlit-movie-recommender.streamlit.app

**Sample Prediction Output**

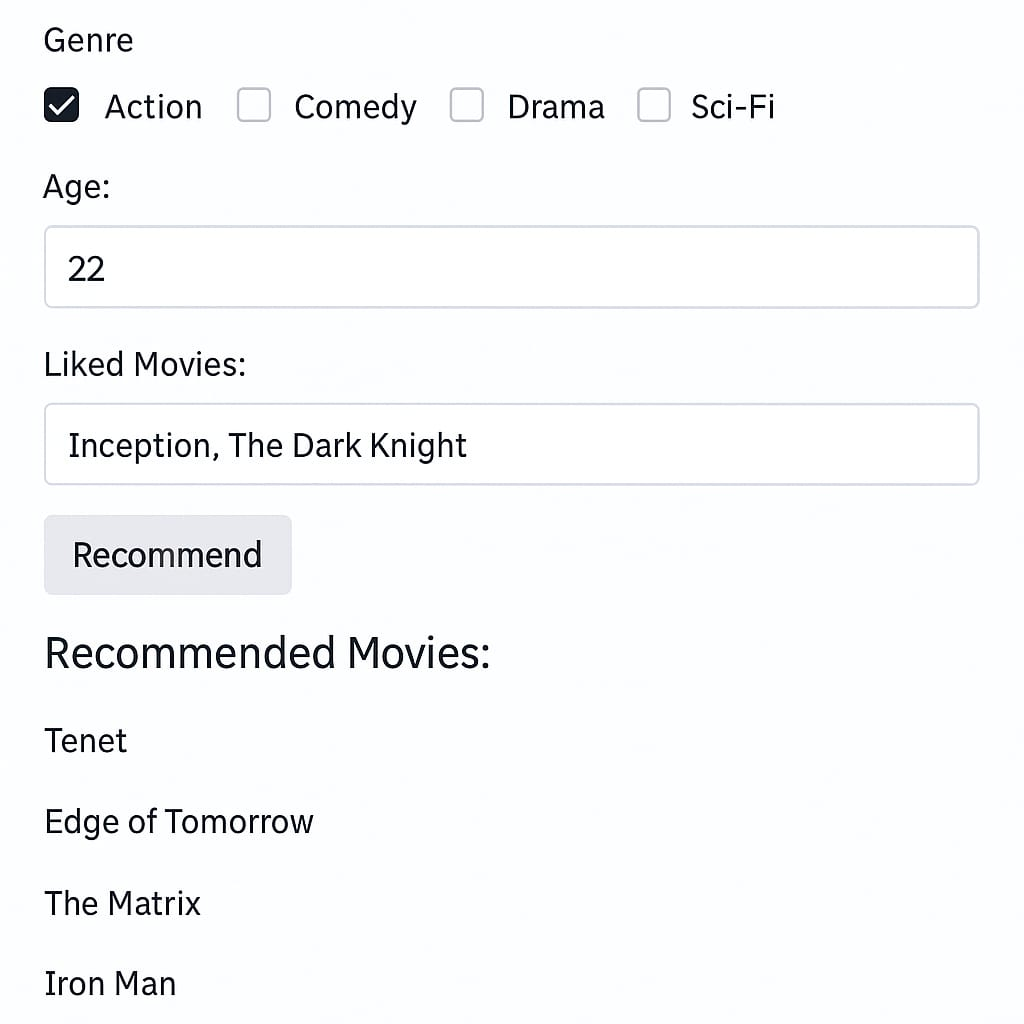
**Input from user:**

* Age: 22
* Favorite Genres: Action, Sci-Fi
* Liked Movies: "Inception", "The Dark Knight"

**Output:**

**Recommended Movies:**

* Tenet
* The Matrix
* Iron Man



**13. Source code**

**All sources are available at:**

https://github.com/Kanishka200511/Kanishka200511.git

# 14. Future scope

* **Cold-Start Problem Mitigation:** Implement hybrid techniques combining content-based and collaborative filtering to better handle new users or movies with little historical data.
* **Incorporation of Real-Time Feedback:**

Add mechanisms for users to rate recommendations, improving model adaptability and personalization through continuous learning.

# Natural Language Interaction:

# Integrate NLP-based chatbots or voice assistants to allow users to interact with the system using natural language queries like “Suggest a sci-fi movie like Inception.”

# Cross-Platform Integration: Extend deployment to mobile apps or smart TVs for wider accessibility and more seamless user experience.

# 13. Team Members and Roles

* **Akalya S (Team Lead):** Managed planning, model oversight, and integration
* **Dharshini V:** Data preprocessing and recommendation logic
* **Kaniga S:** UI design with Streamlet and deployment
* **Kanishka S:** Error analysis, evaluation, and visualizations
* **Mahalakshmi D:** Documentation, reporting, and result compilation