**Student Name :** Maga M

**Register Number :** 732323106029

**Institution :** SSM College Of Engineering

**Department :** Electronics And Communication Engineering

**Date of Submission :** 29.04.2025

**Github Repository Link :** [<https://github.com/ma-ha2005/Delivering-personalize-movie-recommendations-with-an-AI-driven-match-making-system-.git>]

### **1. Problem Statement**

### Develop an advanced movie recommendation system

### Purpose:

### Define the challenges users face on streaming platforms and demonstrate the significance of incorporating social matchmaking.

### Description:

### Users experience choice overload due to vast movie libraries.

### Although initial recommendation systems improve accuracy using hybrid filtering, it does not address the social aspect.

### The matchmaking feature is designed to increase satisfaction by connecting users with similar tastes, thus enhancing engagement

### **2. Project Objectives**

* Goals:
* Build a robust hybrid recommendation model that merges collaborative and content-based approaches.
* Implement user similarity scoring to reliably quantify taste similarities.
* Develop a matchmaking engine to suggest compatible “movie buddies.”
* Visualize trends and insights through dashboards and charts.
* Deploy a user-friendly web interface (using Streamlit, Flask, or Gradio) to demonstrate the system.

**3. Flowchart of the Project Workflow**

The workflow proceeds as follows:

1. Data Collection

2. Data Preprocessing

3. Exploratory Data Analysis (EDA)

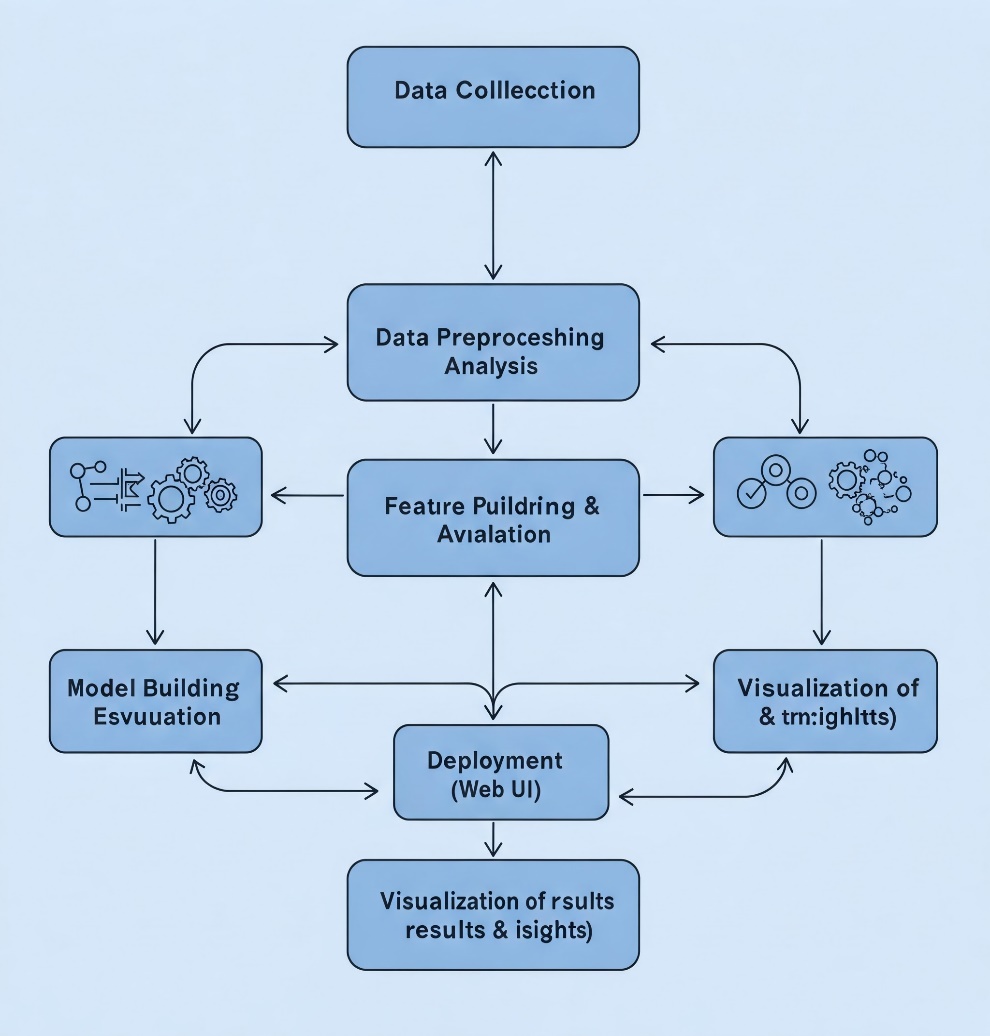
4. Feature Engineering

5. Model Building & Evaluation

6. Visualization of Results & Insights

7. Deployment of the Web Interface

A simple flowchart visualization:



### **4. Data Description**

### Sources & Datasets:

### MovieLens 100K for user ratings on over 1,000 movies.

### TMDB API for movie metadata (genres, descriptions, images).

### Synthetic user data to capture nuanced preferences (such as affinity for specific genres).

### Nature of Data:

### Structured tabular data, combining user-movie interactions, movie details, and additional tags for matchmaking.

### **5. Data Preprocessing**

* Key Steps:
  + Clean the data by removing duplicates and handling any missing values.
  + Encode categorical variables such as movie genres using one-hot encoding and process user attributes as required.
  + Normalize numerical features using methods like Standard Scaler.
  + Detect and manage outliers with boxplots or z-score analysis.
* Data set: [<https://datasets.imdbws.com/title.basics.tsv.gz>]
* Example (in Python):
  + Load ratings and metadata.
  + One-hot encode genres.
  + Scale numerical columns like rating and popularity.

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

### Techniques:

### Univariate analysis: Generate histograms and boxplots to understand the distribution of ratings and genres.

### Bivariate/multivariate analysis: Create scatter plots and correlation matrices to study relationships (e.g., between average rating and popularity).

### Draw insights such as identifying popular genres or trends in user behavior.

### Example Visualization:

### Plot a histogram of movie ratings to see the frequency distribution and overlay a kernel density estimate.

### **7. Feature Engineering**

### Guidelines:

### Construct user profiles based on individual rating histories to represent preferences.

### Calculate similarity metrics (using methods like cosine similarity) between user profiles.

### Generate interaction features (e.g., weighted scores for preferred genres) to enrich model inputs.

### Example:

* Use cosine similarity on user profile vectors to create a similarity matrix.

### **8. Model Building**

* Approaches:
  + Use collaborative filtering methods (such as kNN) to predict movie preferences.
  + Leverage content-based filtering using movie metadata.
  + Combine these into a hybrid model for greater accuracy.
* Evaluation Metrics:
  + MAE (Mean Absolute Error)
  + RMSE (Root Mean Squared Error)
  + Precision/Recall (if ranking recommendations)
  + Example for Model Optimization:
  + Apply GridSearchCV with an algorithm (like Random Forest Regressor) by tuning n\_estimators,max\_depth, etc., then evaluate the model on a test set using MAE, RMSE, and R² metrics..

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

Key Visualizations:

* Feature importance plots (bar charts) to highlight influencing factors.
* User similarity heatmaps to display clusters of similar users.
* Trend plots to examine the distribution of recommendations across genres.
* Example:
* Plot a bar chart of feature importances collected from the optimal Random Forest model.

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

* Programming Language: Python 3.x
* Libraries: pandas, numpy, scikit-learn, matplotlib, seaborn
* Deployment: Streamlit, Flask, or Gradio
* Version Control: Git/GitHub

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

* Outline each team member’s role:
  + Menaka : Project Planning & System Design, Documentation and Reporting
  + Madhanraj : Data Collection & Preprocessing, Testing
  + Maga : Model Building & Evaluation
  + Meganathan : Web Interface & Deployment