Phase-2 Submission Template

Student Name : MATHIN.S

Register Number : 510623205039

Institution : C.ABDUL HAKEEM COLLEGE OF ENGINEERING AND TECHNOLOGY

Department : B.TECH-“IT”.

Date of Submission : 02/05/2025

Github Repository Link : [https://github.com/PRO123-LAB/MATHIN.git]

---

1. Problem Statement

Delivering personalized movie recommendations with an AI driven matchmaking system

- *Revisit and refine the problem based on additional understanding of the dataset.*
- *Clearly define the type of problem (classification, regression, clustering, etc.).*
- *Explain why solving this problem matters (impact, relevance, or application area).]*

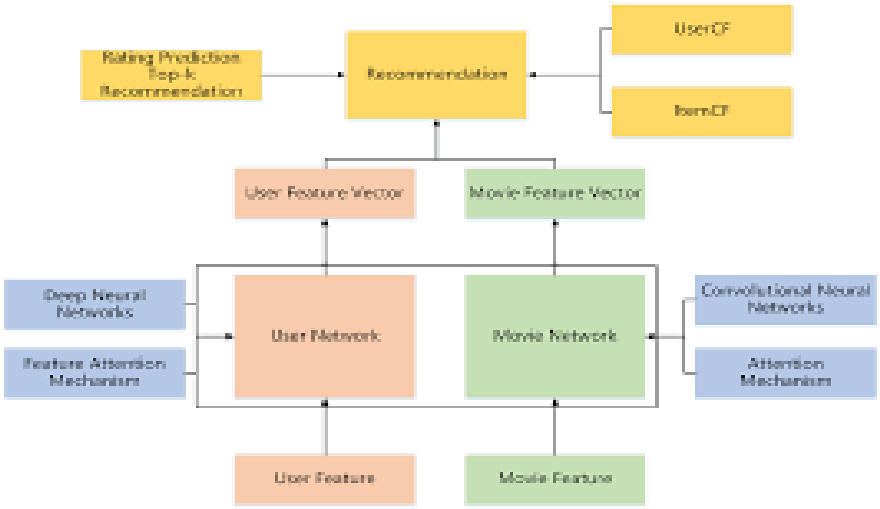
2. Project Objectives

→The primary goal is to offer movie suggestions that are tailored to the individual tastes, preferences, and viewing history of each user.

- **Develop an AI-Driven Matchmaking System:** To achieve personalization, an intelligent system needs to be built using AI techniques to analyze user data and movie data to find relevant matches.
- **Understand User Preferences:** The system must effectively learn and understand the explicit and implicit preferences of users based on their interactions and available data.

- **Leverage Movie Data:** The system should utilize comprehensive information about movies (metadata, content, etc.) to identify relevant titles for recommendations.

3. Flowchart of the Project Workflow



4. Data Description

I. User Data:

This category encompasses all information related to the users of the movie recommendation system.

- **User IDs:** Unique identifiers for each user in the system.
  - **Type:** Integer, String (Alphanumeric)

- **Description:** Used to uniquely identify and track each user's interactions and preferences.
- **Example:** 12345, user_abc, UUID-e45a-b789-c1d2

- **Explicit Preferences (Optional but valuable):** Data directly provided by the user.
  - **Movie Ratings:** Scores given by users to movies they have watched.
    - **Type:** Integer (e.g., 1-5 stars), Float (e.g., 0.5 increments)
    - **Description:** Reflects the user's explicit opinion on a movie.
    - **Example:** 5, 3.5, 1
  - **Genre Preferences:** Genres explicitly selected or indicated as favorites by the user.
    - **Type:** List of Strings (e.g., ["Action", "Comedy", "Sci-Fi"])
    - **Description:** User's stated interest in specific movie genres.
    - **Example:** ["Drama", "Thriller"], ["Animation"]
  - **Actor/Director Preferences:** Actors or directors explicitly followed or marked as favorites.
    - **Type:** List of Strings (e.g., ["Tom Hanks", "Christopher Nolan"])
    - **Description:** User's stated interest in specific individuals involved in filmmaking.
    - **Example:** ["Meryl Streep"], ["Quentin Tarantino"]
  - **Watchlist/Saved Movies:** List of movies a user has explicitly added to watch later.
    - **Type:** List of Movie IDs
    - **Description:** Indicates movies the user has shown interest in watching in the future.
    - **Example:** [MID_101, MID_256, MID_890]

II. Movie Data:

This category contains information about the movies available in the system.

- **Movie IDs:** Unique identifiers for each movie.
  - **Type:** Integer, String (Alphanumeric)
  - **Description:** Used to uniquely identify each movie in the database.
  - **Example:** 101, movie_xyz, MV-987

Basic Metadata:

- **Title:** The name of the movie.
  - **Type:** String
  - **Description:** The official title of the film.
  - **Example:** "The Shawshank Redemption"

Genre(s): Categorization of the movie into one or more genres.

- **Type:** List of Strings (e.g., ["Drama", "Crime"])
- **Description:** Helps in understanding the movie's thematic and stylistic elements.
- **Example:** ["Science Fiction", "Action", "Adventure"]

Release Date: The date when the movie was released.

- **Type:** Date
- **Description:** Can be relevant for users interested in new releases or movies from specific eras.
- **Example:** 1994-09-2

5. Data Preprocessing

I. Data Collection & Ingestion:

- **Gathering User Interaction Data:**
  - **Tracking User Activity:** Monitoring and logging user actions on the platform (views, searches, clicks, watchlist additions, ratings).
  - **Collecting Explicit Feedback:** Acquiring ratings, genre preferences, actor/director follows, and other directly provided information.
  - **Real-time vs. Batch Collection:** Determining whether data is ingested in real-time (for immediate updates) or in batches (e.g., daily processing).
- **Gathering Movie Data:**
  - **Sourcing Movie Metadata:** Obtaining information from internal databases or external APIs (e.g., IMDb, TMDB).
  - **Fetching Content Features:** Downloading and storing movie posters, trailers, and potentially full movie scripts (for advanced NLP).
  - **Regular Updates:** Implementing mechanisms to keep the movie database current with new releases and updated information.

II. Data Cleaning & Preprocessing:

- **Handling Missing Values:**
  - **Identification:** Identifying missing data points in both user and movie datasets (e.g., missing genre for a movie, no ratings for a new user).
  - **Imputation or Removal:** Deciding whether to fill in missing values (e.g., using the most frequent genre) or remove records with too many missing values.
- **Data Type Conversion:** Ensuring all data fields are in the appropriate format for analysis (e.g., converting timestamps to datetime objects, ensuring IDs are consistent).
- **Handling Inconsistent Data:**

- **Standardization:** Ensuring consistent formatting for text data (e.g., converting all text to lowercase).
- **Resolving Duplicates:** Identifying and removing duplicate user records or movie entries.
- **Addressing Outliers:** Detecting and handling unusual data points (e.g., extremely high or low ratings that might be erroneous).

- **Text Preprocessing (for plot summaries, reviews, etc.):**
  - **Tokenization:** Breaking down text into individual words or units.
  - **Stop Word Removal:** Eliminating common words (e.g., "the," "a," "is") that don't carry much meaning.
  - **Stemming/Lemmatization:** Reducing words to their root form to group similar meanings.
  - **TF-IDF or Word Embeddings:** Converting text data into numerical representations that machine learning models can understand.

III. Feature Engineering:

- **User Feature Engineering:**
  - **Recency, Frequency, Monetary (RFM) Features:** Analyzing how recently and frequently a user interacts and the types of movies they engage with.
  - **Genre Affinity Scores:** Calculating a score for each user-genre combination based on their past interactions.
  - **Actor/Director Preference Vectors:** Creating numerical vectors representing a user's affinity for specific actors and directors.
  - **Temporal Features:** Extracting features related to when users typically watch movies (e.g., day of the week, time of day).
- **Movie Feature Engineering:**
  - **Genre Vectors:** Creating binary vectors indicating the presence of different genres for each movie.
  - **Actor/Director Vectors:** Representing movies based on their cast and crew.
  - **Content Embeddings:** Using techniques like Doc2Vec or sentence embeddings to create numerical representations of movie plot summaries.
  - **Visual Feature Extraction:** Using Convolutional Neural Networks (CNNs) to extract features from movie posters.
- **Interaction Feature Engineering:**
  - Creating features based on user-movie interactions (e.g., how many times a user has watched a specific actor's movies).

IV. Data Transformation & Scaling:

- **Normalization/Standardization:** Scaling numerical features to a similar range to prevent features with larger values from dominating the models. Techniques include Min-Max scaling and Z-score standardization.

- **Dimensionality Reduction (Optional):** If the number of features is very high, techniques like Principal Component Analysis (PCA) or t-SNE can be used to reduce the dimensionality while preserving important information.
- **Data Aggregation:** Combining data from different sources or time periods to create more informative features.

V. Data Partitioning:

- **Splitting Data:** Dividing the processed data into training, validation, and testing sets.
  - **Training Set:** Used to train the AI models.
  - **Validation Set:** Used to tune hyperparameters and evaluate model performance during training.
  - **Testing Set:** Used for a final, unbiased evaluation of the trained model's performance on unseen data.
- **Time-Based Splitting:** For recommendation systems, it's often important to split data chronologically to simulate real-world scenarios where models are trained on past data and evaluated on future interactions.

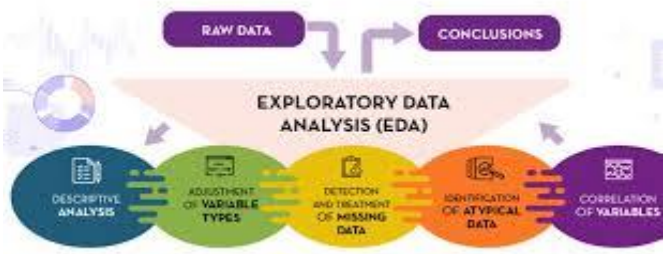
VI. Data Storage & Management:

- **Choosing Appropriate Databases:** Selecting suitable database technologies for storing processed user data, movie data, and interaction data (e.g., relational databases, NoSQL databases, graph databases).
- **Data Warehousing:** Potentially setting up a data warehouse for analytical purposes and long-term storage.
- **Data Governance & Privacy:** Implementing policies and procedures to ensure data quality, security, and compliance with privacy regulations.

6. Exploratory Data Analysis (EDA)

I. User Data EDA:

- **Distribution of User Interactions:**
  - **Number of Ratings per User:** Plot a histogram to see the distribution of how many movies each user has rated. This helps understand user activity levels. Are there a few super-raters and many with very few ratings?
  - **Number of Watched Movies per User:** Similar to ratings, analyze the distribution of the number of movies watched by each user.
  - **Distribution of Ratings:** Examine the overall distribution of ratings given by all users. Is there a bias towards higher or lower ratings?



-
    - **Temporal Analysis of User Activity:** Plot user activity (e.g., number of interactions) over time to identify trends or seasonality in movie watching and rating behavior.
- **User Preference Exploration:**
    - **Top Genres per User (if explicit preferences exist):** Analyze the explicitly preferred genres for each user to see common patterns.
    - **Implicit Genre Preferences:** Infer genre preferences from watch history. For each user, count the number of movies watched in each genre. Visualize the distribution of top genres per user.
    - **Actor/Director Affinity (if explicit or implicit data exists):** Identify frequently watched actors and directors per user.
    - **Distribution of Watchlist Sizes:** Analyze how many movies users typically add to their watchlists.
- **User Segmentation (Initial Exploration):**
    - **Based on Activity:** Group users based on the number of interactions (e.g., low, medium, high activity users).
    - **Based on Rating Behavior:** Identify user segments with different rating tendencies (e.g., generally positive, generally negative, more critical).
- **Handling Missing User Data:**
    - Quantify the amount of missing data for explicit preference fields (if any).

II. Movie Data EDA:

- **Distribution of Movie Properties:**
    - **Genre Distribution:** Plot a bar chart showing the number of movies in each genre. Are some genres significantly more represented than others? Are there movies with multiple genres?
    - **Release Year Distribution:** Analyze the distribution of movie release years. Is the dataset skewed towards older or newer movies?
    - **Runtime Distribution:** Examine the distribution of movie runtimes. Are there typical runtime ranges?
    - **Language and Country of Origin Distribution:** Explore the distribution of movies by language and country of origin.
- **Content Feature Exploration:**

    - **Keyword Analysis (from plot summaries or explicit keywords):** Identify the most frequent keywords associated with movies. This can give insights into common themes and topics. Techniques like word clouds can be useful.
    - **Co-occurrence of Genres and Keywords:** Explore which genres are frequently associated with specific keywords.
- **Community/Collaborative Data Exploration:**
    - **Distribution of Average Movie Ratings:** Plot a histogram of the average ratings for each movie. How are movies generally rated?
    - **Distribution of Number of Ratings per Movie:** Analyze how many ratings each movie has received. Popular movies will likely have more ratings.
    - **Relationship between Average Rating and Number of Ratings:** Investigate if there's a correlation between a movie's average rating and the number of ratings it has received.
- **Handling Missing Movie Data:**
    - Quantify the amount of missing data for different movie attributes (e.g., missing genre, director information).

III. Interaction Data EDA (if available as a separate entity):

- **Distribution of Interactions Over Time:** Analyze the overall number of interactions (watches, ratings, etc.) over time to identify platform usage trends.
- **User-Movie Interaction Matrix Sparsity:** Calculate the sparsity of the user-movie interaction matrix (the percentage of possible interactions that actually occurred). This is a crucial factor for collaborative filtering techniques. A very sparse matrix can pose challenges.

Tools and Techniques for EDA:

- **Histograms:** To visualize the distribution of numerical data.
- **Bar Charts:** To compare the frequencies of categorical data.
- **Scatter Plots:** To explore the relationship between two numerical variables.
- **Box Plots:** To show the distribution and identify outliers in numerical data.
- **Word Clouds:** To visualize the frequency of words in text data.
- **Correlation Matrices:** To understand the linear relationships between numerical features.
- **Summary Statistics:** Calculating mean, median, standard deviation, quartiles, etc., to get a numerical overview of the data.
- **Data Visualization Libraries (Python):** Matplotlib, Seaborn, Plotly.
- **Data Manipulation Libraries (Python):** Pandas.

7. Feature Engineering :

I. User Features:

These features aim to capture the past behavior and preferences of individual users.

- **Recency, Frequency, Monetary (RFM) Features for Movies:**
  - **Movie Recency:** How recently did the user watch a movie? (e.g., days since last watch).
  - **Movie Frequency:** How many movies has the user watched in a given period (e.g., last month, last year)?
  - **Genre Monetary:** How much time has the user spent watching movies of specific genres? (This is a proxy for "monetary" value in this context).

II. Movie Features:

These features describe the characteristics and popularity of individual movies.

- **Basic Metadata Features:**
  - **Genre Vectors:** Binary vectors indicating the presence of each genre.
  - **Release Year:** The year the movie was released (can be used as a numerical feature or binned into categories).
  - **Movie Age:** The number of years since the movie was released.
  - **Runtime:** The duration of the movie.
  - **Language and Country of Origin (as categorical features).**

8. Model Building :

I. Collaborative Filtering Models:

These models leverage the past interactions of users to find similarities and make recommendations.

- **Memory-Based Methods:**
  - **User-User Collaborative Filtering:** Find users who have similar viewing and rating patterns to the target user and recommend movies that those similar users

liked but the target user hasn't seen. Similarity can be calculated using metrics like Pearson correlation or cosine similarity.

- **Item-Item Collaborative Filtering:** Identify movies that are similar based on the ratings they have received from users. Recommend movies similar to those the target user has liked in the past. Similarity can be calculated using adjusted cosine similarity.

II. Content-Based Filtering Models:

These models recommend movies similar to those a user has liked in the past, based on the features of the movies themselves.

- **Profile Building:** Create a profile for each user based on the features of the movies they have liked (e.g., weighted average of genre vectors, TF-IDF vectors of plot summaries).
- **Similarity Calculation:** Calculate the similarity between the user's profile and the features of unseen movies (e.g., using cosine similarity).
- **Recommendation:** Recommend the most similar unseen movies.
- **Machine Learning Classifiers/Regressors:** Train models (e.g., Naive Bayes, Support Vector Machines, Decision Trees, Gradient Boosting) to predict whether a user will like a movie based on its content features and the user's content preferences learned from their past interactions.

III. Hybrid Recommendation Systems:

Combining collaborative and content-based approaches can often lead to better performance and address the limitations of individual methods (e.g., cold start problem).

- **Weighted Averaging:** Combine the scores or rankings from different recommendation engines.
- **Switching:** Use different recommendation engines in different situations (e.g., content-based for new users, collaborative filtering for users with sufficient interaction history).
- **Feature Augmentation:** Use features from one approach as input to the other (e.g., using latent factors from collaborative filtering as features in a content-based model).
- **Meta-Learning:** Train a meta-model to learn how to best combine the outputs of different recommendation engines.

9. Visualization of Results & Model Insights

:

- **Example Recommendations for Specific Users:**

  - Display a user's profile (e.g., top genres, liked actors) alongside the top recommendations generated for them. This helps in understanding if the recommendations align with the user's known preferences.
  - Show the movies the user has already watched and rated, along with the recommendations. This provides context and allows for a qualitative assessment of relevance.
  - Include the predicted rating (if the model predicts ratings) alongside the recommendations.
- **Distribution of Recommended Movies:**
  - Visualize the genres of the movies being recommended overall. Is the system recommending a diverse set of genres, or is it heavily skewed towards a few?
  - Show the distribution of the age or release year of the recommended movies.
  - Analyze the popularity (e.g., number of ratings) of the recommended movies. Is the system primarily recommending popular titles, or is it also suggesting niche content?
- **Comparison of Different Recommendation Strategies:**
  - If we've implemented multiple recommendation approaches (e.g., collaborative filtering, content-based, hybrid), visualize the types of recommendations each strategy generates for the same user. This can highlight the strengths and weaknesses of each approach.
- **User Interface Mockups:**
  - Create mockups of how the recommendations would be presented to the user within the movie platform (e.g., carousels, lists, personalized landing pages). This helps in evaluating the user experience of the recommendation delivery.

II. Visualization of Model Performance:

- **Evaluation Metrics Over Time (if applicable):**
  - If the model is trained incrementally or evaluated over different time periods, plot the trends of key evaluation metrics (e.g., precision@k, recall@k, NDCG@k, RMSE) to see how performance evolves.
- **Distribution of Prediction Errors (for rating prediction):**
  - Plot a histogram of the differences between the predicted ratings and the actual ratings (residuals). This helps in understanding the model's accuracy and identifying any systematic biases in its predictions (e.g., consistently over- or under-predicting).
- **Precision-Recall Curves:**
  - For binary relevance tasks (e.g., whether a user will click on a recommendation), plot precision against recall at different threshold levels. This helps in understanding the trade-off between precision and recall.
- **ROC Curves and AUC (for binary relevance):**
  - Plot the True Positive Rate against the False Positive Rate at different threshold levels and calculate the Area Under the Curve (AUC). This is another way to evaluate the performance of binary classification model

10. Tools and Technologies Used

I. Programming Languages:

- **Python:** The dominant language for data science, machine learning, and AI due to its rich ecosystem of libraries and frameworks. It's highly likely to be the primary language.
- **Java/Scala:** Could be used for building scalable backend systems and data processing pipelines, especially if integrating with existing enterprise infrastructure.

II. Data Storage and Management:

- **Relational Databases (SQL):**
  - **PostgreSQL:** A powerful, open-source relational database known for its reliability and extensibility. Suitable for storing structured data like user profiles, movie metadata, and interaction logs.
  - **MySQL:** Another popular open-source relational database, widely used for web applications.

III. Data Processing and Feature Engineering:

- **Pandas (Python):** For data manipulation and analysis, including cleaning, transforming, and feature engineering.
- **NumPy (Python):** For numerical computations and array operations, essential for working with machine learning models.
- **Scikit-learn (Python):** A comprehensive library for machine learning tasks, including feature scaling, dimensionality reduction, and various model implementations.
- **Spark (with PySpark for Python):** A distributed computing framework for processing large datasets, especially useful for batch processing of user interactions and movie data.

11. Team Members and Contributions

Data cleaning	*T.MASOOD NAWAZ.*
EDA	*R.KUMARAN.*
Feature engineering	*A.BHARATH*
Model development	*S.MATHIN*
Documentation and reporting	*MD.AFNAN SAQUIB*