

G H Patel College of Engineering & Technology

**(The Charutar Vidya Mandal (CVM) University)**

**New V. V. Nagar**

# DEPARTMENT OF COMPUTER ENGINEERING

**AI/ML Report**

**on**

***Movie Recommendation System***

# Submitted By

**Name of Student :** Hitanshi Kansara

**Enrolment Number :** 12302130503001

**Name of Student :** Rushabh Lathiya

**Enrolment Number :** 12302130503002

# Guided By

**Prof. Priyang Bhatt**

# Artificial Intelligence & Machine Learning (202046702)

**A.Y. 2024-25 EVEN TERM**

**Title of the Project**

**Movie Recommendation System Using Content-Based Filtering**

**Objective**

The goal of this project is to build a recommendation system that suggests movies similar to a user’s selected movie. The system uses **content-based filtering**, analyzing features like genres, keywords, cast, crew, and overviews to compute similarity between movies. The final output is a Streamlit web app that displays the top 5 recommended movies along with their posters fetched from the TMDB API.

**Dataset Used**

* **Source**: The Movie Database (TMDB)
* **Files**:
  1. tmdb\_5000\_movies.csv: Contains metadata for 5,000 movies (budget, genres, keywords, etc.).
  2. tmdb\_5000\_credits.csv: Includes cast and crew details for the same movies.
* **Key Features**:
  1. genres: Movie categories (e.g., Action, Drama).
  2. keywords: Tags describing the movie’s theme.
  3. cast: Top 5 actors.
  4. crew: Director(s).
  5. overview: Short movie description.

**Model Chosen**

**Content-Based Filtering with Cosine Similarity**

1. **Data Preprocessing**:
   * Merged movies and credits datasets on the title column.
   * Extracted and cleaned features (genres, keywords, cast, director).
   * Combined features into a single tags column for vectorization.
   * Applied **Porter Stemming** to reduce words to their root form (e.g., "running" → "run").
2. **Vectorization**:
   * Used CountVectorizer with max\_features=5000 to convert text data into a bag-of-words matrix.
3. **Similarity Calculation**:
   * Computed pairwise cosine similarity between movies using cosine\_similarity from scikit-learn.

**Performance Metrics**

* **Qualitative Evaluation**:  
  Recommendations were validated by checking if the output movies share meaningful similarities (e.g., genres, directors, or themes) with the input movie.
* **Example**:  
  Input: *"The Dark Knight"* → Outputs include *"Batman Begins"* (same director and genre).

**Challenges & Learnings**

**Challenges:**

1. **Data Cleaning**:
   * Handling nested JSON structures in genres, keywords, and cast.
   * Removing spaces between words (e.g., "Science Fiction" → "ScienceFiction") to improve vectorization accuracy.
2. **API Integration**:
   * Fetching posters dynamically using TMDB API required handling API rate limits and invalid movie IDs.
3. **Memory Constraints**:
   * Storing and processing large similarity matrices (5000x5000).

**Learnings:**

1. **Text Processing**:
   * Stemming and vectorization techniques for NLP pipelines.
2. **Similarity Metrics**:
   * Practical implementation of cosine similarity for recommendation systems.
3. **Deployment**:
   * Building interactive UIs with Streamlit and integrating external APIs.

**Tools & Libraries**

* **Python**: Primary programming language.
* **Libraries**: Pandas, NumPy, Streamlit, scikit-learn, NLTK, Requests.
* **API**: TMDB for movie posters.