🎬 Movie Recommendation System – Final Report

📌 Project Title:

Content-Based Movie Recommendation System Using Metadata

🧠 Objective:

The primary goal of this project is to develop a **movie recommendation system** that suggests similar movies based on descriptive metadata such as **cast, crew, genres, keywords, and overview**. Unlike collaborative filtering, which relies on user behavior, this project focuses solely on the **content of the movie** to generate recommendations.

🛠️ Tools and Technologies Used:

- **Programming Language:** Python
- **Libraries:**
  - `Pandas` and `NumPy` – for data manipulation and analysis
  - `Seaborn` and `Matplotlib` – for data visualization
  - `NLTK` – for text preprocessing (like stemming)
  - `Scikit-learn` – for vectorization and similarity measurement
- **Vectorization Technique:** CountVectorizer
- **Similarity Measure:** Cosine Similarity
- **Development Environment:** Jupyter Notebook

📁 Datasets:

- **`movies.csv`** – Contains metadata about movies such as title, overview, genres, and keywords.
- **`credits.csv`** – Contains additional metadata such as cast and crew (including the director).

These datasets are part of the **TMDb 5000 Movie Dataset**, which is a popular public dataset for movie-related machine learning tasks.

🔍 Methodology:

1. Data Loading and Merging:

- Loaded `movies.csv` and `credits.csv`.
- Merged them on the movie `title` or `id` for a complete metadata record per movie.

2. Data Cleaning:

- Removed duplicates and rows with missing essential data.
- Parsed JSON-like fields such as `cast`, `crew`, `genres`, and `keywords`.

3. Feature Engineering:

Extracted key components to describe each movie:

- **Director Name** from crew
- **Top 3 Cast Members**
- **Genres and Keywords**

4. Tag Construction:

- Combined overview, genres, director, keywords, and cast into a unified string field called `tags`.
- Preprocessed text (lowercasing, removing spaces, stemming) for uniformity.

5. Vectorization and Similarity:

- Applied `CountVectorizer` to convert the `tags` text into numerical feature vectors.
- Computed **cosine similarity** between movie vectors.

6. Recommendation Function:

- Defined a Python function `recommend(movie_title)` that:
  - Finds the movie in the dataset.
  - Retrieves the top 5 most similar movies based on cosine similarity scores.

📈 Results:

- The system can **recommend 5 similar movies** for any valid movie input.

- It works well for action, sci-fi, drama, and fantasy genres.
- Example: Inputting **"Avatar"** returns similar sci-fi and fantasy films based on crew, genre, and plot similarity.

💡 Advantages:

- Doesn't require user ratings or interactions.
- Works purely on metadata – ideal for new or unrated movies.
- Easy to extend with more metadata fields.

🚀 Future Improvements:

- **Hybrid Approach:** Combine content-based with collaborative filtering using user ratings.
- **TF-IDF Vectorization:** Replace CountVectorizer to weigh rare words higher.
- **Web Deployment:** Build a frontend using **Streamlit**, **Flask**, or **React**.
- **Search Bar:** Add dynamic search to input movie titles.
- **Visualization:** Graphically show recommendation relationships between movies.

✅ Conclusion:

This content-based recommendation system is a foundational machine learning project that can be scaled into a full-fledged movie recommender. By leveraging metadata, we have created a lightweight yet powerful tool to explore and suggest movies based on similarity in content.

***** END *****