Movie Recommender System - Project Steps

1. Data Collection

- Datasets used: `tmdb_5000_movies.csv` and `tmdb_5000_credits.csv`- Used

  The Movie Database (TMDB) API optionally movie posters.

2. Data Preprocessing

- Merged movies and credits datasets on `movie_id`.
- Extracted and cleaned important fields like title, genres, overview, cast, crew.
- Converted text data (JSON-like) into usable lists using `ast.literal_eval`.

3. Feature Engineering

- Combined relevant features into a single `tags` field.
- Applied CountVectorizer or TF-IDF to convert text to vectors.
- Used cosine similarity to compute similarity between all movies.
- Saved the similarity matrix using `pickle` for reuse.

4. Efficient Data Handling

- Hosted large `similarity.pkl` file externally on Google Drive.
- Downloaded it at runtime using `gdown` to bypass GitHub's 100MB limit.
- Ensured file isn't tracked in Git using `.gitignore`.

5. Web App Interface

- Built interactive UI using Streamlit.
- User inputs movie name, gets top N similar movie recommendations.
- Retrieved posters dynamically using TMDB API and displayed in the app.

6. Deployment

- Deployed on Streamlit Cloud.

- Used external file hosting to manage large models efficiently.

- Streamlined requirements using `requirements.txt`.

7. Project Structure

movie_recommender_app/

app.py

similarity.pkl

movies_dict.pkl

datasets(.csv)

.gitignore

requirements.txt

8. Tech Stack

- Python Libraries: pandas, numpy, sklearn, pickle, ast, gdown, requests, Streamlit

- API: TMDB API

- Deployment: GitHub + Streamlit Cloud

🔍 So... Is It "Machine Learning"?

Yes — even though it's **not model training or supervised ML**, it **uses ML tools and concepts**:

Component	ML Category	Your Usage
Text Vectorization	NLP / Feature Engineering	TF-IDF / CountVectorizer
Similarity Calculation	Unsupervised ML concept	Cosine similarity on metadata vectors
Data Pipeline Handling	ML Workflow	Using Pandas and Pickle for processing