Movie Recommendation System Deployemnt Link

Step 1: Introduction

The objective of this project was to develop and deploy a Movie Recommendation System using the MovieLens dataset. Instead of traditional collaborative filtering or matrix factorization, this project implements a content-based recommendation approach with a similarity matrix. The system recommends movies that are most similar to a user-selected movie, and it is deployed on Hugging Face Spaces using Streamlit.

Step 2: Dataset

Dataset Used: MovieLens

Preprocessing:

- Extracted movie titles and corresponding IDs.
- Created a similarity matrix (stored as similarity.pkl).
- Stored movie metadata as movie_list.pkl for efficient retrieval.

The similarity matrix was computed offline (before deployment) and then loaded at runtime for fast recommendation.

Step 3: Methodology

Recommendation Approach (Content-Based Filtering)

For a selected movie, the system:

- Looks up its index in the movies DataFrame.
- Retrieves the most similar movies from the similarity matrix.
- Returns the Top-5 recommended movies along with their posters.

Fetching Posters

- Movie posters are fetched dynamically using the TMDB (The Movie Database) API.
- If a poster is missing, a placeholder image is used.

Application Workflow

- User selects a movie from the dropdown (st.selectbox).
- On clicking "Recommend":
  - Results are displayed in a 5-column Streamlit layout.
  - The recommend() function retrieves 5 similar movies.
  - Posters are fetched via the TMDB API.

Step 3: Implementation

Core Recommendation Function

```python
1 usage  MD Irfanul Kabir Hira
def recommend(movie):
    index = movies[movies['title'] == movie].index[0]
    movies_list = sorted(list(enumerate(similarity[index])),reverse=True, key=lambda x: x[1])[1:6]

    recommended_movies = []
    recommended_movies_posters = []
    for i in movies_list:
        movie_id = movies.iloc[i[0]].movie_id

        recommended_movies.append(movies.iloc[i[0]].title)
        # fetch Poster From API
        recommended_movies_posters.append(fetch_poster(movie_id))
    return recommended_movies, recommended_movies_posters

similarity = pickle.load(open('similarity.pkl', 'rb'))
```

Step 4: Requirements

- streamlit
- pandas
- requests

Step 5: Deployment

- The application was deployed on Hugging Face Spaces using Streamlit.
- Deployment Link : https://huggingface.co/spaces/erfanulkabirhira/DataSynthis_Job_task

Step 6: Results

- The app successfully generates Top-5 similar movie recommendations for any movie selected by the user.
- Posters and titles are displayed interactively in the Hugging Face app.
- Example: If the user selects The Dark Knight, recommendations may include Batman Begins, Inception, etc.

Step 7: Conclusion

- Best Approach Used: Content-based filtering with similarity matrix.
- Why: It is simple, fast, and works well for this project's deployment goal. Unlike matrix factorization, it does not require user-rating data during inference, making it lightweight and easy to deploy.
- Deployment Success: The Hugging Face app provides an easy-to-use interface, allowing users to test movie recommendations interactively.