Movie Recommendation System with Sentiment Filtering

🔹 Introduction

In the modern digital landscape, users are overwhelmed with vast choices of movies. Recommender systems help personalize user experience by suggesting movies based on preferences. This project builds a **hybrid recommendation system** using **collaborative filtering**, **genre similarity**, and **sentiment analysis** of user reviews to suggest the top 5 movies similar to a selected one.

🔹 Abstract

The aim of this project is to develop a movie recommendation engine that not only relies on user ratings and genre similarities but also integrates the **emotional polarity of movie reviews**. By combining collaborative filtering, content-based filtering, and natural language sentiment analysis, the system provides more human-centric, relevant suggestions. The recommendation pipeline is wrapped in an interactive **Streamlit** UI for ease of access.

🔹 Tools Used

- **Python** – Core programming language
- **Pandas, NumPy** – Data handling and transformation
- **Scikit-learn** – Cosine similarity computation
- **TextBlob** – Sentiment analysis on textual reviews
- **CountVectorizer** – Genre feature extraction
- **Streamlit** – Frontend UI for interaction

🔹 Steps Involved in Building the Project

1. **Data Collection & Cleaning**
    - Imported datasets: `movies.csv`, `ratings.csv`, `reviews.csv`
    - Removed duplicate titles and missing data.
2. **Collaborative Filtering**
    - Built a user-movie rating matrix.
    - Calculated cosine similarity between movies based on user ratings.
3. **Content-Based Filtering (Genre)**
    - Used `CountVectorizer` to process movie genres.
    - Generated cosine similarity matrix for content-based matching.

4. **Sentiment Analysis**
    - Applied `TextBlob` on review texts to compute polarity scores.
    - Calculated average sentiment per movie and normalized it.
5. **Hybrid Recommendation Engine**
    - Combined collaborative and genre scores (70%) with sentiment scores (30%).
    - Ranked movies based on final weighted scores.
6. **UI Development with Streamlit**
    - Built an interactive dropdown for movie selection.
    - Displayed top 5 recommendations with sentiment scores.

🔹 Conclusion

This hybrid movie recommendation system enhances the traditional approach by incorporating **real user sentiments**, ensuring more **emotionally aligned and accurate suggestions**. The project demonstrates the potential of combining collaborative and content filtering with NLP techniques, offering a robust, user-friendly experience through a Streamlit-based interface.

Files Created in This Project

1. **movie_recommendation_without_sentiment_score.py** – Basic recommendation using collaborative + genre filtering.
2. **movie_recommendation_with_sentiment_score.py** – Hybrid recommendation with added sentiment analysis *(runs slower than the model without sentiment scoring)*.
    - ⬤ **Reason**: The sentiment model performs additional computations by analyzing textual review data using sentiment analysis (TextBlob). Even though the sentiment scores are precomputed, combining them with other metrics and managing more data slightly increases the runtime compared to the simpler model.
3. **generate_reviews_from_ratings.ipynb** – Converts numeric ratings into synthetic review texts.
4. **movie_recommendation_without_sentiment_score.ipynb** – Jupyter notebook version of the basic recommender with detailed code explanations using markdown cells.

How to Run the App

To launch any `.py` Streamlit app, open terminal and run: streamlit run file_name.py