REPORT

1. Introduction

The recommendation system uses **content-based filtering** to analyze movie metadata and recommend movies similar to those selected by the user. This approach relies on textual information from the movies, which includes descriptions, keywords, genres, cast, and crew.

2. Libraries Used

To implement this recommendation system, the following libraries were used:

- **Numpy** and **Pandas**: For efficient data manipulation and numerical operations.
- **Matplotlib**: To visualize data and gain insights.
- **JSON**: For parsing JSON-formatted data fields.
- **NLTK (Natural Language Toolkit)**: To process and stem text data.
- **Scikit-Learn**: For vectorizing the text data and calculating similarity metrics.

3. Datasets

The **movies** and **credits** datasets contain essential metadata required to create the recommendation system:

- **Movies Dataset**: Consists of information such as budget, genres, release date, popularity, etc.
- **Credits Dataset**: Contains cast and crew information.

Both datasets contain 4,803 rows, corresponding to individual movies. The data was merged to allow for feature extraction and modeling.

4. Preprocessing and Feature Engineering

- **Data Merging**: The movies and credits datasets were merged using the movie_id column, creating a single dataframe that contains all necessary features.
- **Feature Extraction**: Selected features include:

- **Genres**: Genre information was extracted from JSON objects in the genres column.
- **Keywords**: Keywords were parsed to identify significant terms associated with each movie.
- **Cast and Crew**: Key individuals in the cast and crew (such as lead actors and directors) were extracted to represent movie-specific characteristics.

5. Text Vectorization

- **Tag Creation**: A new feature called **tags** was created by combining selected textual information from cast, crew, genres, and keywords. This allowed for a more comprehensive representation of each movie's characteristics in a single textual format.
- **Text Stemming**: Using `nltk.PorterStemmer`, words within the tags were stemmed to ensure that similar terms (e.g., "running" and "run") were treated as identical, enhancing the effectiveness of vectorization.
- **Vectorization Using CountVectorizer**: The tags were transformed into vectors using the `CountVectorizer` from scikit-learn, which converts text into a sparse matrix of token counts. This step facilitated numerical representation for subsequent similarity computations.

6. Similarity Computation

- **Cosine Similarity**: The `cosine_similarity` function was applied to the vectorized tags, creating a similarity matrix. This matrix represents the similarity between each pair of movies, with higher values indicating greater similarity.
- **Sorting Similar Movies**: An index-based enumeration list was created, allowing for identification of the 10 most similar movies to any selected movie. The movies were sorted in descending order based on cosine similarity scores, excluding the selected movie itself.

7. Recommendation System

The final recommendation system works as follows:

- A movie selected by the user is compared to all other movies based on cosine similarity.
- The system identifies the 10 most similar movies, providing a ranked list of recommendations.

8. Findings and Observations

- **High Accuracy in Similar Genres**: The system accurately recommends movies of similar genres and tones.
- **Impact of Keywords and Cast**: Including cast and keywords as features improves recommendations by aligning movies with similar themes and key contributors.
- **Vectorization and Similarity**: Text vectorization and cosine similarity effectively capture movie characteristics for recommendation purposes.

9. Future Improvements

Potential improvements for this recommendation system include:

- **Incorporating User Ratings**: Blending content-based filtering with collaborative filtering could personalize recommendations further.
- **Advanced NLP**: Using deep learning-based NLP models like Word2Vec or Transformers could capture more nuanced relationships between movies.

10. Conclusion

This project successfully created a content-based recommendation system that provides relevant movie suggestions. The combination of feature extraction, text vectorization, and cosine similarity yielded accurate recommendations that align closely with user interests.