Project Title: Movie Recommendation System using Data Analytics and Machine Learning

Name: Vignesh A
Batch: 10 Feb 2025 to 10 May 2025
UNID: UMIP277374

Abstract

This project presents a movie recommender system that leverages collaborative filtering techniques to suggest personalized movie recommendations to users. Using the MovieLens dataset, combined with metadata from TMDB, the system identifies similar users based on their rating behavior and recommends movies they have not yet seen. The model is designed to run efficiently within a Google Colab environment, making it accessible and easy to use for academic and prototype purposes.

Introduction

The objective of this project is to build a movie recommendation system using collaborative filtering techniques. The model leverages the MovieLens dataset, enriched with metadata from TMDB, to suggest relevant movies to users based on their rating behavior and similarities with other users.

Dataset Description

We utilized the following files from the MovieLens and TMDB metadata:

- **ratings_small.csv**: Contains 100,000 ratings from 700 users on 9,000 movies.
- **movies_metadata.csv**: Metadata for 45,000 movies including title, genre, release date, etc.

Key preprocessing steps:

- Converted the `id` field in `movies_metadata` to `movieId` for compatibility.
- Removed or filled missing data as required.

Tools and Technologies

- Python
- Pandas, NumPy
- Scikit-learn
- Matplotlib, Seaborn (for EDA)

Exploratory Data Analysis (EDA)

We explored the distribution of movie ratings and the number of ratings per movie.

- Most ratings were clustered around 3 to 4 stars.

- A small number of movies received a disproportionately high number of ratings.

Visualizations included:

- Histogram of rating distribution
- Histogram of ratings count per movie

Recommendation System

Sample Code

```
# Import necessary libraries
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import StandardScaler

# Load data
ratings = pd.read_csv('ratings_small.csv')
movies = pd.read_csv('movies_metadata.csv')
movies.rename(columns={'id': 'movieId'}, inplace=True)
movies['movieId'] = pd.to_numeric(movies['movieId'], errors='coerce')

# Create user-movie matrix
user_movie_matrix = ratings.pivot_table(index='userId', columns='movieId',
values='rating').fillna(0)

# Normalize ratings
scaler = StandardScaler()
user_movie_matrix_scaled = scaler.fit_transform(user_movie_matrix)

# Compute user similarity matrix
user_similarity_df = pd.DataFrame(
    cosine_similarity(user_movie_matrix_scaled),
    index=user_movie_matrix.index,
    columns=user_movie_matrix.index
)

# Get similar users
def get_similar_users(user_id, n=5):
    return
user_similarity_df[user_id].sort_values(ascending=False)[1:n+1].index

# Recommend movies
def recommend_movies(user_id, num_recommendations=5):
    similar_users = get_similar_users(user_id)
    similar_users_ratings = ratings[ratings['userId'].isin(similar_users)]
    user_movies = ratings[ratings['userId'] == user_id]['movieId'].tolist()
    recommended_movies = (

similar_users_ratings[~similar_users_ratings['movieId'].isin(user_movies)]
        .groupby('movieId')
        .agg({'rating': 'mean'})
        .sort_values(by='rating', ascending=False)
        .head(num_recommendations)
        .reset_index()
    )
    return pd.merge(recommended_movies, movies, on='movieId')[['title',
'rating']]
```

```
# Example: Get recommendations for user ID 1
recommendations = recommend_movies(user_id=1, num_recommendations=10)
print(recommendations)
```

We implemented a **user-based collaborative filtering** model using cosine similarity.

Steps:

1. **User-Movie Matrix**: Created a matrix with users as rows and movies as columns, filling in ratings where available.
2. **Standardization**: Scaled ratings using StandardScaler.
3. **Similarity Calculation**: Used cosine similarity to determine user-user similarity.
4. **Recommendation Logic**: Identified the top N similar users, aggregated their ratings, and recommended the highest-rated movies not yet seen by the target user.

Functionality:

- `get_similar_users(user_id, n=5)`: Fetches n users most similar to the given user.
- `recommend_movies(user_id, num_recommendations=5)`: Recommends movies by aggregating ratings from similar users.

Sample Output:

For user ID 1, the system recommended top 10 movies with highest predicted ratings based on similar users.

Optional Deployment

A sample Flask API was provided to demonstrate deployment:

```
@app.route('/recommend', methods=['POST'])
def recommend():
    data = request.get_json(force=True)
    user_id = data['user_id']
    num_recommendations = data.get('num_recommendations', 10)
    recommendations = recommend_movies(user_id, num_recommendations)
    return jsonify(recommendations.to_dict(orient='records'))
```

Conclusion

This project successfully demonstrates a basic collaborative filtering-based movie recommender system that works in Google Colab without third-party dependencies like Surprise. The results are customizable and can be enhanced with content-based filtering or hybrid approaches for better performance.

Future Enhancements

- Add genre filtering and user preferences
- Combine collaborative and content-based methods
- Deploy using Streamlit or Flask on Render/Heroku
- Enable real-time user input via web interface

9. Project Link

https://github.com/vignesh-a-09/Unified-Mentor-Internship-2025/blob/main/Movie%20recommendation%20system/Movie_recommendation_system%20(1).ipynb

---