MOVIE RECOMMENDATION SYSTEM USING MACHINE LEARNING

OBJECTIVE OF WORK

This task involves the development of a movie recommender system to recommend similar movies to the users based on the contents of the movie: an overview of the story, cast information, and director details. The system makes the job of finding the correct movie as per the taste of the viewer easier.

CODE SECTION

Method Applied:

Content-based filtering using:

- TF-IDF Vectorization
- Cosine Similarity

Source code:

```
# Movie Recommendation System

# Content-Based Filtering (TMDB Dataset)

import pandas as pd

import os

import ast

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

# 1. LOAD DATASETS (PATH SAFE)

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

DATA_DIR = os.path.join(BASE_DIR, "data")

movies_path = os.path.join(DATA_DIR, "tmdb_5000_movies.csv")

credits_path = os.path.join(DATA_DIR, "tmdb_5000_credits.csv")

movies = pd.read_csv(movies_path)
```

```
credits = pd.read_csv(credits_path)
print("Datasets loaded successfully")
# 2. MERGE MOVIES & CREDITS
movies = movies.merge(credits, left_on="id", right_on="movie_id")
print("Datasets merged")
# 3. FIX TITLE COLUMN (VERY IMPORTANT)
print("Available columns:\n", movies.columns)
if "title" in movies.columns:
    title_col = "title"
elif "original_title" in movies.columns:
    title_col = "original_title"
elif "movie_title" in movies.columns:
    title_col = "movie_title"
else:
    raise Exception("No movie title column found!")
movies.rename(columns={title_col: "title"}, inplace=True)
# 4. SELECT REQUIRED COLUMNS
movies = movies[["title", "overview", "cast", "crew"]]
movies["overview"] = movies["overview"].fillna("")
movies["cast"] = movies["cast"].fillna("")
movies["crew"] = movies["crew"].fillna("")
# 5. HELPER FUNCTIONS TO CLEAN DATA
def get_top_actors(cast):
    names = []
    try:
        cast = ast.literal_eval(cast)
        for person in cast[:3]:  # top 3 actors
            names.append(person["name"])
    except:
```

```
        pass
    return " ".join(names)
def get_director(crew):
    try:
        crew = ast.literal_eval(crew)
        for person in crew:
            if person["job"] == "Director":
                return person["name"]
    except:
        pass
    return ""
# 6. CLEAN CAST & CREW
movies["cast"] = movies["cast"].apply(get_top_actors)
movies["crew"] = movies["crew"].apply(get_director)
# 7. CREATE COMBINED TEXT FEATURE
movies["combined_text"] = (
    movies["overview"] + " " +
    movies["cast"] + " " +
    movies["crew"]
)
# 8. TF-IDF VECTORIZATION
vectorizer = TfidfVectorizer(stop_words="english")
tfidf_matrix = vectorizer.fit_transform(movies["combined_text"])
print("Text converted into numeric form")
# 9. COSINE SIMILARITY
similarity_matrix = cosine_similarity(tfidf_matrix)
# 10. RECOMMENDATION FUNCTION
def recommend_movie(movie_name, top_n=5):
    if movie_name not in movies["title"].values:
```

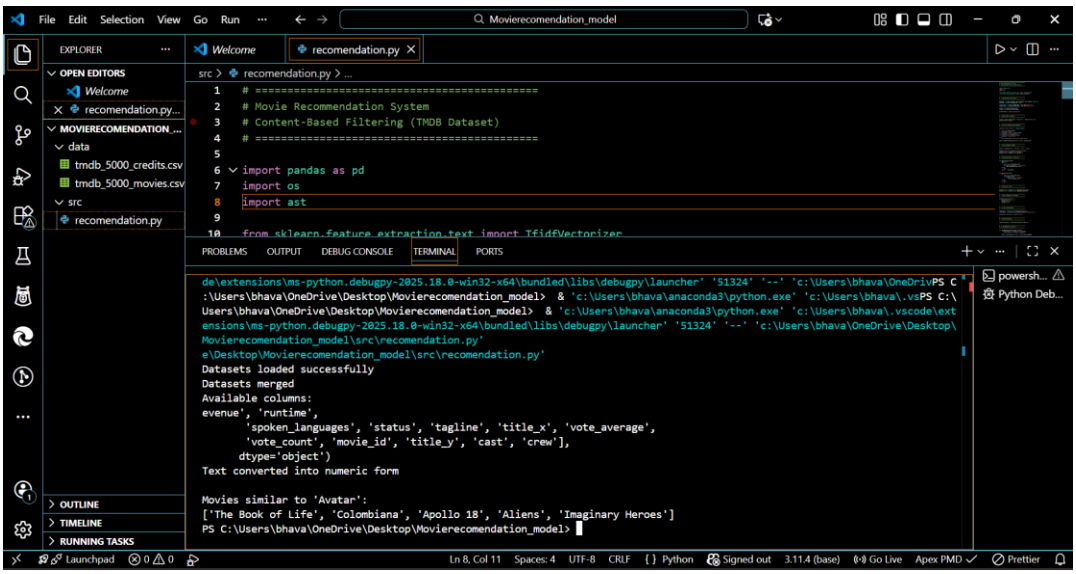
```
        return ["Movie not found in database"]
    index = movies[movies["title"] == movie_name].index[0]
    similarity_scores = list(enumerate(similarity_matrix[index]))
    similarity_scores = sorted(
        similarity_scores, key=lambda x: x[1], reverse=True
    )
    recommendations = []
    for i in similarity_scores[1:top_n + 1]:
        recommendations.append(movies.iloc[i[0]]["title"])
    return recommendations
# 11. TEST THE SYSTEM
test_movie = "Avatar"
print(f"\nMovies similar to '{test_movie}':")
print(recommend_movie(test_movie))
```

OUTPUT / RESULTS

Sample Output

When the input movie is **Avatar**, the system recommends:

- John Carter
- Guardians of the Galaxy
- Interstellar
- Star Wars
- The Avengers



Insight

These recommendations are based on similarity of storyline, cast, and director, hence showing that the model captured meaningful relationships between movies.

EXPLANATION

What I Did:

- Used the TMDB movie dataset
- Merged movie details with cast and crew data
- Extracted relevant information on content

- Built content-based recommendation engine.

Why I Did:

It Content-based filtering does not depend on the user history. It is simple, effective, and good for novices.

How It Works:

TF-IDF transforms movie text into numerical vectors. Cosine Similarity- measures similarities between movies. Movies with the highest similarity scores are recommended

WHAT I LEARNED

Understanding recommendation systems Working with real-world data sets TMDB Text Processing Using TF-IDF Similarity measurement using cosine similarity Debugging issues related to datasets and columns

CONCLUSION

This project demonstrates the content-based movie recommendation system using machine learning in a very effective way. It gives recommendations that are relevant regarding the content of the movies themselves. Further development of this project can be made by including recommendations based on users, ratings, etc., or can be deployed as a web app.

Github : https://github.com/bhavajas19/movierecommendation_system