Title of Project

Movie Recommendation System Using Content-Based Filtering

Objective

To build a movie recommendation system that suggests movies to users based on the content features of movies such as genre, keywords, tagline, cast, and director.

Data Source

The dataset used for this project is from the YBIFoundation GitHub repository, containing various features related to movies.

Import Library

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import difflib
```

Import Data

```
DF = pd.read_csv("https://github.com/YBIFoundation/Dataset/raw/main/Movies%20Recommendation.csv")
```

Describe Data

```
print(DF.head())
print(DF.info())
print(DF.shape)
print(DF.columns)
```

Data Visualization

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Distribution of Movie Genres
genres = DF['Movie_Genre'].str.split('|', expand=True).stack().value_counts()
sns.barplot(y=genres.index, x=genres.values, palette='viridis')
plt.title('Distribution of Movie Genres')
plt.xlabel('Count')
plt.ylabel('Genre')
plt.show()
```

- **Data Preprocessing**

```
DF_features =
DF[['Movie_Genre','Movie_Keywords','Movie_Tagline','Movie_Cast','Movie_Director']].fillna('')
```

- **Define Target Variable (y) and Feature Variables (X)**

In this content-based filtering approach, the feature variables (X) are the combined text features of movies, while there is no explicit target variable (y).

Feature Engineering

Combine the features into a single text for each movie:

```
x = DF_features['Movie_Genre'] + ' ' + DF_features['Movie_Keywords'] + ' ' +
DF_features['Movie_Tagline'] + ' ' + DF_features['Movie_Cast'] + ' ' +
DF_features['Movie_Director']
```

Convert Features Text to Tokens

```
tfidf = TfidfVectorizer()
x = tfidf.fit_transform(x)
print(x.shape)
```

Compute Similarity Score using Cosine Similarity

```
Similarity_Score = cosine_similarity(x)
print(Similarity_Score.shape)
```

User Input and Validation for Closest Spelling

```
Favorite_Movie_Name = input('Enter your favourite movie name: ')
All_Movies_Title_List = DF['Movie_Title'].tolist()
```

```
Movie_Recommendation = difflib.get_close_matches(Favorite_Movie_Name,
All_Movies_Title_List)
print(Movie_Recommendation)

Close_Match = Movie_Recommendation[0]
print(Close_Match)

Index_of_Close_Match_movie = DF[DF.Movie_Title ==
Close_Match]['Movie_ID'].values[0]
print(Index_of_Close_Match_movie)
```

Get Recommendation Scores

```
Recommendation_Score =
list(enumerate(Similarity_Score[Index_of_Close_Match_movie]))
print(len(Recommendation_Score))
```

Sort Movies Based on Recommendation Scores

```
Sorted_Similar_Movies = sorted(Recommendation_Score, key=lambda x: x[1],
reverse=True)
print('Top 30 Movies Suggested for You: \n')

i = 1
for movie in Sorted_Similar_Movies:
  index = movie[0]
  title_from_index = DF[DF.index == index]['Movie_Title'].values[0]
  if i < 31:
    print(i, '.', title_from_index)
    i += 1
```

- **Modeling**

In this case, the model is the similarity measure computed using TF-IDF and cosine similarity. No explicit training phase is required as it is a content-based filtering approach.

- **Model Evaluation**

Evaluate the quality of recommendations based on user feedback or by measuring precision and recall on a test set, if available.

- **Prediction**

```
Movie_Name = input('Enter your favourite movie name: ')
list_of_all_titles = DF['Movie_Title'].tolist()

Find_Close_Match = difflib.get_close_matches(Movie_Name, list_of_all_titles)
Close_Match = Find_Close_Match[0]
Index_of_Movie = DF[DF.Movie_Title ==
Close_Match]['Movie_ID'].values[0]

Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Movie]))
Sorted_Similar_Movies = sorted(Recommendation_Score, key=lambda x: x[1],
reverse=True)

print('Top 10 Movies Suggested for You: \n')

i = 1
for movie in Sorted_Similar_Movies:
   index = movie[0]
   title_from_index = DF[DF.index == index]['Movie_Title'].values[0]
   if i < 11:
      print(i, '.', title_from_index)
      i += 1
```

- **Explanation**

This movie recommendation system uses content-based filtering. The features of each movie (genre, keywords, tagline, cast, and director) are combined into a single text string.

TF-IDF is used to convert these text features into numerical values, and cosine similarity is used to compute the similarity between movies. When a user inputs a favorite movie, the system finds the closest match and recommends movies that are most similar to the input movie based on content features.