import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

from google.colab import files

# Upload your CSV file containing title, genres, and description

import pandas as pd

from google.colab import files

uploaded = files.upload()  # Upload your CSV file

movies\_md  = pd.read\_csv('credits.csv', encoding='latin-1') # Specify encoding as 'latin-1'

# If 'latin-1' doesn't work, try other encodings like 'ISO-8859-1', 'cp1252', etc.

movies\_md.head()

# Now you can use movies\_md

print(movies\_md.isnull().sum()) # Changed df to movies\_md

# Load the movie data. Make sure this file contains the columns 'genres' and 'description'.

# Replace 'movies.csv' with the actual name of your file.

# Specify encoding as 'latin-1' or another appropriate encoding if needed.

try:

    movies\_md  = pd.read\_csv('movies.csv', encoding='latin-1')

except FileNotFoundError:

    print("Error: 'movies.csv' not found. Please upload the correct file.")

    # You might want to exit or handle this error appropriately

    # exit() # Uncomment this line if you want to stop execution

# If 'latin-1' doesn't work, try other encodings like 'ISO-8859-1', 'cp1252', etc.

# If your file is 'credits.csv' and it contains the required columns, load that instead:

# movies\_md  = pd.read\_csv('credits.csv', encoding='latin-1')

# Check the columns in your DataFrame to ensure 'genres' and 'description' exist

# print(movies\_md.columns)

print(movies\_md.isnull().sum())

# Sample user profile (preferred genres and themes)

user\_profile = {

    'preferred\_genres': 'Action Sci-Fi',

    'preferred\_description\_keywords': 'technology hacker reality dream future'

}

# Ensure 'genres' and 'description' columns exist and handle potential NaNs

# Filling NaNs with empty strings to avoid errors during string concatenation

if 'genres' in movies\_md.columns and 'description' in movies\_md.columns:

    movies\_md['genres'] = movies\_md['genres'].fillna('')

    movies\_md['description'] = movies\_md['description'].fillna('')

    # Combine genre and description to create a single feature

    # Use the correct DataFrame name: movies\_md instead of movies

    movies\_md['features'] = movies\_md['genres'] + ' ' + movies\_md['description']

    # Add user's preferences as a "pseudo movie" to match against

    user\_features = user\_profile['preferred\_genres'] + ' ' + user\_profile['preferred\_description\_keywords']

    all\_features = movies\_md['features'].tolist() + [user\_features]

    # Vectorize using TF-IDF

    vectorizer = TfidfVectorizer(stop\_words='english')

    tfidf\_matrix = vectorizer.fit\_transform(all\_features)

    # Compute cosine similarity between user and all movies

    similarity\_scores = cosine\_similarity(tfidf\_matrix[-1], tfidf\_matrix[:-1]).flatten()

    # Add scores and rank

    movies\_md['similarity'] = similarity\_scores

    recommended = movies\_md.sort\_values(by='similarity', ascending=False)

    # Show top N recommendations

    top\_n = 3

    print(f"Top {top\_n} personalized movie recommendations:\n")

    # Ensure 'title' column exists in your DataFrame before trying to access it

    if 'title' in recommended.columns and 'genres' in recommended.columns:

        print(recommended[['title', 'genres', 'similarity']].head(top\_n))

    else:

        print("Warning: 'title' or 'genres' column not found in the DataFrame.")

        print(recommended.head(top\_n)) # Print available columns instead

else:

    print("Error: 'genres' or 'description' column not found in the DataFrame.")

    print("Please ensure your uploaded CSV file contains these columns.")

