#insert data set to google colab

import pandas as pd

df=pd.read\_csv('/content/netflix\_titles.csv')

df

# Rearrange the columns (example: B, C, A order)

df = df[['listed\_in', 'country','type','duration','title', 'description']]

# Save the modified DataFrame to a new CSV file

df.to\_csv('modified\_dataset.csv', index=False)

df = df.dropna(subset=['description'])

df['description'].fillna('', inplace=True)

print(df)

#input and output

x=df.iloc[:,0:4].values

y=df.iloc[:,5:7].values

print(x)

print(y)

#train&test-75% of the data to trainig and 25% to testing

from sklearn.model\_selection import train\_test\_split

x\_Train,x\_Test,y\_Train,y\_Test=train\_test\_split(x,y,random\_state=0)

print(x.shape)

print(x\_Train.shape)

print(x\_Test.shape)

print(y.shape)

print(y\_Train.shape)

print(y\_Test.shape)

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import linear\_kernel

# User input

country\_input = input("Enter your preferred country: ").lower()

genre\_input = input("Enter your preferred movie genre: ").lower()

duration\_input = input("Enter your preferred movie duration: ").lower() # e.g., '90 min'

# Initialize and fit the TF-IDF vectorizer for movie descriptions

tfidf\_vectorizer = TfidfVectorizer(stop\_words='english')

tfidf\_vectorizer = TfidfVectorizer(stop\_words='english')

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(df['description'])

# Calculate the cosine similarity between movies

cosine\_sim = linear\_kernel(tfidf\_matrix, tfidf\_matrix)

# Function to get movie recommendations

def get\_recommendations(title):

idx = df.index[df['title'] == title].tolist()

if not idx:

return []

idx = idx[0]

sim\_scores = list(enumerate(cosine\_sim[idx]))

sim\_scores = sorted(sim\_scores, key=lambda x: x[1], reverse=True)

sim\_scores = sim\_scores[1:6] # Get the top 5 similar movies

movie\_indices = [i[0] for i in sim\_scores]

return df['title'].iloc[movie\_indices]

# Filter the dataset based on user input with partial matching

filtered\_movies = df[

(df['country'].str.lower().str.contains(country\_input)) &

(df['listed\_in'].str.lower().str.contains(genre\_input)) &

(df['duration'].str.lower().str.contains(duration\_input))

]

# If no matches were found, recommend similar movies based on genre

if filtered\_movies.empty:

# Filter based on genre only

filtered\_movies = df[df['listed\_in'].str.lower().str.contains(genre\_input)]

if not filtered\_movies.empty:

print("No exact matches found. Here are similar movies based on genre:")

# Get recommendations for each filtered movie

for title in filtered\_movies['title']:

recommendations = get\_recommendations(title)

print(f"Recommended movies for {title}:")

for rec\_title in recommendations:

print(f"- {rec\_title}: {df.loc[df['title'] == rec\_title, 'description'].values[0]}")

if filtered\_movies.empty:

print("No movies found matching your criteria.")