#!/usr/bin/env python

# coding: utf-8

# ## I create content base recommendation system for movies

#

#

# ### Import our necessary library

# In[1]:

import numpy as np

import pandas as pd

import json

import difflib

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

import streamlit as st

# ### Load our data

# In[65]:

movies\_data=pd.read\_csv('m.csv')

movies\_data.head(3)

# ### Select our features

# In[3]:

selected\_features = ['genres','keywords','tagline','cast','director']

print(selected\_features)

# ### Preprocessing data

# In[4]:

for feature in selected\_features:

movies\_data[feature] = movies\_data[feature].fillna('')

# In[5]:

combined\_features = movies\_data['genres']+' '+movies\_data['keywords']+' '+movies\_data['tagline']+' '+movies\_data['cast']+' '+movies\_data['director']

# ### Convert text to numerical vector

# In[6]:

vectorizer = TfidfVectorizer()

# In[7]:

feature\_vectors = vectorizer.fit\_transform(combined\_features)

# In[8]:

print(feature\_vectors )

# ### Calculate cosine\_similarity between all vectors

# In[9]:

similarity = cosine\_similarity(feature\_vectors)

# In[10]:

similarity = cosine\_similarity(feature\_vectors)

# ### Run a function to recommend movies

# #### The get\_close\_matches() function returns a list of close matched strings that satisfy the cutoff.

# In[44]:

list\_of\_all\_titles = movies\_data['title'].tolist()

def get\_recommendations(movie\_name):

find\_close\_match = difflib.get\_close\_matches(movie\_name, list\_of\_all\_titles)

close\_match = find\_close\_match[0]

index\_of\_the\_movie = movies\_data[movies\_data.title == close\_match]['index'].values[0]

sim\_scores = list(enumerate(similarity[index\_of\_the\_movie]))

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

sim\_scores = sim\_scores[:70]

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

a = set(movies\_data[['title']].iloc[movie\_indices]['title'])

return a

def get\_recommendations2(movie\_name):

find\_close\_match = difflib.get\_close\_matches(movie\_name, list\_of\_all\_titles)

close\_match = find\_close\_match[0]

index\_of\_the\_movie = movies\_data[movies\_data.title == close\_match]['index'].values[0]

sim\_scores = list(enumerate(similarity[index\_of\_the\_movie]))

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

sim\_scores = sim\_scores[:70]

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

a = set(movies\_data[['title']].iloc[movie\_indices]['title'])

return a

def get\_recommendations\_intersection(movie\_name1, movie\_name2):

rec1 = get\_recommendations(movie\_name1)

rec2 = get\_recommendations2(movie\_name2)

r\_final= pd.DataFrame({'title':list(rec1.intersection(rec2))})

final\_suggestion=movies\_data[movies\_data['title'].isin(r\_final['title'].tolist())].drop(['index','budget','homepage','id',

'keywords',

'original\_language',

'overview',

'popularity',

'production\_companies',

'release\_date',

'revenue',

'runtime','production\_countries','original\_title',

'spoken\_languages',

'status',

'tagline',

'vote\_average',

'vote\_count',

'crew'],1).reset\_index(drop=True)

if len(final\_suggestion)==0:

print('sorry, you should select two similar movie')

else:

return final\_suggestion

# In[45]:

get\_recommendations\_intersection('inception','interstaller')

# # use streamlit library to creat web app

#

# In[ ]:

import streamlit as st

import requests

from streamlit\_lottie import st\_lottie

# In[14]:

st.subheader('Hi Im Ali Nadi Khorasgani' )

st.title('This is a movie recommendation system Web App')

# In[15]:

def load\_lottieurl(url):

r=requests.get(url)

if r.status\_code !=200:

return None

return r.json()

lottie\_coding= load\_lottieurl('https://assets5.lottiefiles.com/packages/lf20\_khzniaya.json')

st\_lottie(lottie\_coding,height=300,key='coding')

# In[ ]:

# In[16]:

st.image('abcd.jpeg')

st.markdown("---")

# In[17]:

movies\_list = movies\_data['title'].values

selected\_movie1 = st.selectbox( " select your first favorite movie", movies\_list,key=1 )

selected\_movie2 = st.selectbox( "select your first favorite movie", movies\_list,key=2 )

st.write('we suggest you these movies ')

if st.button('Show Recommendation'):

recommended\_movie\_names = get\_recommendations\_intersection(selected\_movie1,selected\_movie2)

st.write(recommended\_movie\_names)

st.markdown("---")

st.markdown('[download\_dataset](https://drive.google.com/file/d/1cCkwiVv4mgfl20ntgY3n4yApcWqqZQe6/view)')