```
#Import all necessary packages

import requests
import os
import bs4
from apiclient.discovery import build
import string
from youtube_transcript_api import YouTubeTranscriptApi
import csv
import json
import pandas as pd
from sentence_transformers import SentenceTransformer
from sklearn import preprocessing
from textblob import TextBlob
from scipy.spatial import distance
import pickle


#Load the text embedding model

vectorize = SentenceTransformer('paraphrase-multilingual-mpnet-base-v2')


#Initialize the parameters for the Youtube API

DEVELOPER_KEY = "AIzaSyDqlrLX0prKppP3eG02uqprQJk080XoKrc"
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"

#Build the Youtube API Object

youtube_object = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION,developerKey =
DEVELOPER_KEY)


#Function to classify a video as Clickbait or Not Clickbait
#vID is the Video ID of the Youtube Video

def isclickbait(vID):

  #Loading the model

  model = pickle.load(open('/content/isclickbait.sav','rb'))

  #Creating the object used for normalizing the values

  mms = preprocessing.MinMaxScaler()

  #Words often associated with clickbait stored as set to find intersection
```

```
  clickbait_words_corpus = {"fake", "clickbait", "click", "bait", "lie", "false",
                            "waste", "thumbnail", "title", "subscribe", "subscribed",
                            "wasted", "useless", "bullshit", "baited", "rat",
                            "misinformation", "credible", "crap", "tosh", "dumb",
                            "life", "time", "unreliable", "lack", "bogus", "fraud",
                            "phony", "scam", "sham", "trick",
"cheat","hoax","rumour","rumor",
                            "gossip"}

  #Range of data of the features used in training, required for normalizing the
data

  ranges = {'likes':[[-1],[10051603]],
            'views' :  [[367],[213578965]],
            'dislikes': [[-1],[386489]],
            'noOfComments' : [[0],[100]],
            'count_fake_comment' : [[0],[20]],
            'neg_comments' : [[0],[37]]
            }


  #Extract video related data / meta-data


video_stat=youtube_object.videos().list(part='snippet,statistics,contentDetails',id
=vID).execute()
  likes = video_stat['items'][0]['statistics']['likeCount'] if 'likeCount' in
video_stat['items'][0]['statistics'] else -1
  views = video_stat['items'][0]['statistics']['viewCount']
  dislikes = video_stat['items'][0]['statistics']['dislikeCount']
  ldRatio = (int(likes) - int(dislikes))/(int(likes) + int(dislikes))

  #Extract the comments of the video

  comment_data = []
  try:

video_response=youtube_object.commentThreads().list(part='snippet,replies',videoId=
vID,maxResults=100).execute()
    for item in video_response['items']:
                      comment =
item['snippet']['topLevelComment']['snippet']['textDisplay']
                      comment_data.append(comment)
  except:
    print("Disabled Comments, get search URLs")
    url = 'https://google.com/search?q='
    request_result=requests.get( url + meta_data[i]['title'] + " fake")
    soup = bs4.BeautifulSoup(request_result.text,
                             "html.parser")
    comment_data = [i.getText() for i in soup.find_all( 'h3' )]
```

```
  #Extract the audio transcript of the video

  try:
    audioTranscript = YouTubeTranscriptApi.get_transcript(vID)
    transcript = ""
    for i in audioTranscript:
      transcript += i['text'] + " "
  except Exception as e:
      transcript = " ".join(comment_data)

  noOfComments = len(comment_data)

  #Count the no of comments which claim the video to be 'fake' or 'Clickbait'

  count_fake_comment = 0
  neg_comments = 0
  for comment in comment_data:
      processed_comment = comment.translate(str.maketrans('', '',
string.punctuation))
      if TextBlob(comment).sentiment.polarity < 0:
        neg_comments += 1
      unique_words_comment = set(processed_comment.split(" "))
      if len(clickbait_words_corpus.intersection(unique_words_comment)) >= 1:
          count_fake_comment += 1
  try:
      #Calculate Fake Comment Ratio and Positive comment ratio

      fake_comment_ratio = count_fake_comment / len(comment_data)
      pos_ratio = (noOfComments - neg_comments) / len(comment_data)
  except ZeroDivisionError:
      fake_comment_ratio = -1
      pos_ratio = -1


  #Get the title of the video

  title = video_stat['items'][0]['snippet']['title']

  #Vectorize the title of the video

  title = vectorize.encode(title)

  #Vectorize the audio transcript of the video

  transcript = vectorize.encode(transcript)

  #Find the cosine similarity of the above vectors
```

```
  titleVtranscript = distance.cosine(title,transcript)

  #Normalizing the data using the initialized ranges

  ranges['likes'].insert(1,[likes])
  ranges['views'].insert(1,[views])
  ranges['dislikes'].insert(1,[dislikes])
  ranges['noOfComments'].insert(1,[noOfComments])
  ranges['count_fake_comment'].insert(1,[count_fake_comment])
  ranges['neg_comments'].insert(1,[neg_comments])
  likes = mms.fit_transform(ranges['likes'])[1][0]
  views = mms.fit_transform(ranges['views'])[1][0]
  dislikes = mms.fit_transform(ranges['dislikes'])[1][0]
  noOfComments = mms.fit_transform(ranges['noOfComments'])[1][0]
  count_fake_comment = mms.fit_transform(ranges['count_fake_comment'])[1][0]
  neg_comments = mms.fit_transform(ranges['neg_comments'])[1][0]

  #Storing all the data collected in a single list for prediction

  data =
[likes,views,dislikes,ldRatio,noOfComments,count_fake_comment,fake_comment_ratio,ti
tleVtranscript,neg_comments,pos_ratio]

  #Prediction

  res = (model.predict_proba([data])[::,1]*100)[0]

  #Print the results

  print("Our Model predicted that this video is {:.2f}% clickbait".format(res))
```