```
#Import all necessary packages

import requests
import os
import bs4
from apiclient.discovery import build
import csv
import datetime
import string
from youtube_transcript_api import YouTubeTranscriptApi
import csv
import json
import pandas as pd
from sentence_transformers import SentenceTransformer

#Loading the pre-trained model for text embedding

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

#Parameters for YouTube API

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

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

#Main Dictionaries to store the retrieved values

MetaDataMain = {}
CommentsDataMain = {}
next_page_token = ""


#Function to get the video related data

def getdata(query, max_results, pageToken=""):
    meta_data = {}
    comments_data = {}

    #Get a list of Video IDs

    search_keyword = youtube_object.search().list(q = query, type = "video",
                                                  part = "id, snippet",  order =
'viewCount',
                                                  maxResults = max_results,
pageToken = pageToken).execute()
    nextPageToken = search_keyword["nextPageToken"]
    results = search_keyword.get("items", [])
```

```
    v_id = []
    for result in results:
         if result['id']['kind'] == "youtube#video":
                v_id.append(result["id"]["videoId"])
    flag = {}
    for i in v_id:
        flag[i] = True
        meta_data[i] = {}


    #Get Audio Transcript

    for i in v_id:
        try:
            transcript = YouTubeTranscriptApi.get_transcript(i)
            meta_data[i]["audioTranscript"] = [" ".join(t['text'] for t in
transcript)]
        except Exception as e:
            meta_data[i]["audioTranscript"] = "No Transcript Found"

    #Get meta-data of videos

    for i in v_id:
        print("Processing V-Id", i)
        if flag[i] == False:
            continue
        try:
            temp = {'videoId': i}

video_stat=youtube_object.videos().list(part='snippet,statistics,contentDetails',id
=i).execute()

channel_stat=youtube_object.channels().list(part='status,snippet,statistics',id=vid
eo_stat['items'][0]['snippet']['channelId']).execute()

category_stat=youtube_object.videoCategories().list(part='snippet',id=video_stat['i
tems'][0]['snippet']['categoryId']).execute()
            temp['title'] = video_stat['items'][0]['snippet']['title']
            temp['likes'] = video_stat['items'][0]['statistics']['likeCount'] if
'likeCount' in video_stat['items'][0]['statistics'] else -1
            temp['views'] = video_stat['items'][0]['statistics']['viewCount']
            temp['dislikes'] = video_stat['items'][0]['statistics']['dislikeCount']
if 'dislikeCount' in video_stat['items'][0]['statistics'] else -1
            temp['thumbnail'] =
video_stat['items'][0]['snippet']['thumbnails']['medium']['url']
            temp['category'] = category_stat['items'][0]['snippet']['title']
            temp['ldRatio'] = (int(temp['likes']) - int(temp['dislikes'])) /
(int(temp['likes']) + int(temp['dislikes'])) if 'likes' in temp and 'dislikes' in
temp else None
            temp['channelVideos'] =
```

```
channel_stat['items'][0]['statistics']['videoCount'] if 'videoCount' in
channel_stat['items'][0]['statistics'] else -1
            temp['channelSubscribers'] =
channel_stat['items'][0]['statistics']['subscriberCount'] if 'subscriberCount' in
channel_stat['items'][0]['statistics'] else -1
            temp['channelMadeForKids'] =
channel_stat['items'][0]['status']['madeForKids'] if 'madeForKids' in
channel_stat['items'][0]['status'] else False
            meta_data[i].update(temp)
        except Exception as e:
            print("Some error with keys, exit")
            print(e)
            continue

    #Get comments of the video

    for i in v_id:
        if flag[i] == False:
            continue
        temp = []
        try:
            try:

video_response=youtube_object.commentThreads().list(part='snippet,replies',videoId=
i,maxResults=100).execute()
                for item in video_response['items']:
                    comment =
item['snippet']['topLevelComment']['snippet']['textDisplay']
                    temp.append(comment)
                comments_data[i]= temp
            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")
                temp = [i.getText() for i in soup.find_all( 'h3' )]
                comments_data[i]= temp
                pass
        except Exception as e:
            print("Disabled Comments and error with urls, exit")
            print(e)
            continue

    return meta_data, comments_data, nextPageToken

#The data-fetch function is called repetetively until the required number of
vidoe-related data is obtained.

while len(MetaDataMain) < 175:
```

```
    meta_data, comments_data, next_page_token =
getdata('tiktok',25,next_page_token)
    MetaDataMain.update(meta_data)
    CommentsDataMain.update(comments_data)



#Calculate the number of comments which claim the video is fake

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"}
fake_comment_data = {}
temp = MetaDataMain
for v_id in CommentsDataMain:
    count_fake_comment = 0
    for comment in CommentsDataMain[v_id]:
        processed_comment = comment.translate(str.maketrans('', '',
string.punctuation))
        unique_words_comment = set(processed_comment.split(" "))
        if len(clickbait_words_corpus.intersection(unique_words_comment)) >= 1:
            count_fake_comment += 1
    try:
        fake_comment_ratio = count_fake_comment / len(CommentsDataMain[v_id])
    except ZeroDivisionError:
        fake_comment_ratio = -1
    temp[v_id].update({"noOfComments":
len(CommentsDataMain[v_id]),"noOfFakeComments":
count_fake_comment,"fakeCommentRatio": fake_comment_ratio})

MetaDataMain = temp

#Initialize the list of columns to vectorize

vecList = ['title','description','thumbnailCaption','commentsData']

#Vectorizing the columns

for i in vecList:
  df[i] = df[i].apply(lambda x: vectorize.encode(x))


#Drop the unnecssary columns

df = df.drop(columns=['videoId','thumbnail','Unnamed: 0'])
```

```
#Initialize the list of columns for the which the values need to be normalized
before training

normlist =
['likes','dislikes','views','publishedAt','channelAge','channelViews','channelSubsc
ribers','channelVideos']


#Normalizing the values

from sklearn import preprocessing
for i in normlist:
  x = df[[i]].values.astype(float)
  min_max_scaler = preprocessing.MinMaxScaler()
  x_scaled = min_max_scaler.fit_transform(x)
  df[i] = x_scaled


#Saving the processed data as a new CSV file

df.to_csv("VecDataSetCategorized.csv")
```