#importing necessary libraries

import os

import numpy as np

import scipy.io.wavfile as wav

#from python\_speech\_features import mfcc

#defining the necessary functions for creating a dataset for KNN matching

#defining a function to get the distance between feature vectors and find neighbours

def distance(instance1,instance2,k):

distance=0

mm1=instance1[0]

cm1=instance1[1]

mm2=instance2[0]

cm2=instance2[1]

#method to calculate distance between two instances

distance=np.trace(np.dot(np.linalg.inv(cm2),cm1))

distance+=(np.dot(np.dot((mm2-mm1).transpose() , np.linalg.inv(cm2)) ,mm2-mm1 ))

distance+=np.log(np.linalg.det(cm2))-np.log(np.linalg.det(cm1))

distance-=k

return distance

#to be checked within a given dataset(dataset of features.)

def getNeighbours(trainingset,instance,k):

distance=[]

for x in range(len(trainingSet)):

dist=distance(trainingSet[x],instance,k)+ distance(instance,trainingSet[x],k)

distances.append((trainingSet[x][2],dist))

distances.sort(key=operator.itemgetter(1))

neighbours=[]

for x in range(k):

neighbors.append(distances[x][0])

print("top",k,"neighbours are",neighbours)

return neighbours

#identify the nearest neighbours:

def nearestClass(neighbours):

classVote={}

for x in range(len(neighbours)):

response=neighbours[x]

if response in classVote:

classVote[response]+=1

else:

classVote[respose]=1

sorter=sorted(classVote.items(),key=operator.itemgetter(1),reverse=True)

return sorter[0][0]

@app.route ('/',methods=['Get'])

def index():

# Main page

return render\_template('music.html')

@app.route('/predict', methods=['GET','POST'])

def upload():

if request.method =='POST':

# get the file from post request

f = request.files['image']

# save the file to ./uploads

basepath ="D:/TSB Projects\Music Genre Detection\flask"

file\_path = os.path.join(basepath, 'uploads', secure\_filename(f.filename))

f.save(file\_path)

print(file\_path)

i=1

results + {1: 'blues' , 2: 'classical' ,3: 'country' , 4: 'disco' , 5: 'hiphop', 6: 'jazz' , 7: 'metal' , 8: 'pop' ,9: 'raggae' , 10: 'rack'}

(rate,sig)=wav.read(file\_path)

print(rate,sig)

mfcc\_feat=mfcc(sig,rate,winlen=0.020,appendEnergy=False)

covariance = np.cov(np.matrix.transpose(mfcc\_feat))

mean\_matrix = mfcc\_feat.mean(0)

feature=(mean\_matrix,covariance,0)

pred=nearestClass(getNeighbours(dataset ,feature ,8))

print("predicted genre = ",pred,"class = ",results[pred])

return "this song is classified a "+str(results[prend])

if \_\_name\_\_ == '\_\_main\_\_':

app.run(threaded = False)