#Features Extraction

import numpy as np

from sklearn import preprocessing

import python\_speech\_features as mfcc

def calculate\_delta(array):

"""Calculate and returns the delta of given feature vector matrix"""

rows,cols = array.shape

deltas = np.zeros((rows,20))

N = 2

for i in range(rows):

index = []

j = 1

while j <= N:

if i-j rows-1:

second = rows-1

else:

second = i+j

index.append((second,first))

j+=1

deltas[i] = ( array[index[0][0]]-array[index[0][1]] + (2 \* (array[index[1][0]]-array[index[1][1]])) ) / 10

return deltas

def extract\_features(audio,rate):

"""extract 20 dim mfcc features from an audio, performs CMS and combines

delta to make it 40 dim feature vector"""

mfcc\_feat = mfcc.mfcc(audio,rate, 0.025, 0.01,20,appendEnergy = True)

mfcc\_feat = preprocessing.scale(mfcc\_feat)

delta = calculate\_delta(mfcc\_feat)

combined = np.hstack((mfcc\_feat,delta))

return combined

#training Speakers Model

import cPickle

import numpy as np

from scipy.io.wavfile import read

from sklearn.mixture import GMM

from speakerfeatures import extract\_features

import warnings

warnings.filterwarnings("ignore")

#path to training data

source = "development\_set\\"

#path where training speakers will be saved

dest = "speaker\_models\\"

train\_file = "development\_set\_enroll.txt"

file\_paths = open(train\_file,'r')

count = 1

# Extracting features for each speaker (5 files per speakers)

features = np.asarray(())

for path in file\_paths:

path = path.strip()

print path

# read the audio

sr,audio = read(source + path)

# extract 40 dimensional MFCC & delta MFCC features

vector = extract\_features(audio,sr)

if features.size == 0:

features = vector

else:

features = np.vstack((features, vector))

# when features of 5 files of speaker are concatenated, then do model training

if count == 5:

gmm = GMM(n\_components = 16, n\_iter = 200, covariance\_type='diag',n\_init = 3)

gmm.fit(features)

# dumping the trained gaussian model

picklefile = path.split("-")[0]+".gmm"

cPickle.dump(gmm,open(dest + picklefile,'w'))

print '+ modeling completed for speaker:',picklefile," with data point = ",features.shape

features = np.asarray(())

count = 0

count = count + 1

#Evaluating Performance On Test Set

import os

import cPickle

import numpy as np

from scipy.io.wavfile import read

from speakerfeatures import extract\_features

import warnings

warnings.filterwarnings("ignore")

import time

#path to training data

source = "development\_set\\"

modelpath = "speaker\_models\\"

test\_file = "development\_set\_test.txt"

file\_paths = open(test\_file,'r')

gmm\_files = [os.path.join(modelpath,fname) for fname in

os.listdir(modelpath) if fname.endswith('.gmm')]

#Load the Gaussian gender Models

models = [cPickle.load(open(fname,'r')) for fname in gmm\_files]

speakers = [fname.split("\\")[-1].split(".gmm")[0] for fname

in gmm\_files]

# Read the test directory and get the list of test audio files

for path in file\_paths:

path = path.strip()

print path

sr,audio = read(source + path)

vector = extract\_features(audio,sr)

log\_likelihood = np.zeros(len(models))

for i in range(len(models)):

gmm = models[i] #checking with each model one by one

scores = np.array(gmm.score(vector))

log\_likelihood[i] = scores.sum()

winner = np.argmax(log\_likelihood)

print "\tdetected as - ", speakers[winner]

time.sleep(1.0)