**Speech Emotion Recognition with Machine Learning -Divvela Vishnu Sai Kumar**

Speech emotion recognition, the best ever python mini project. The best example of it can be seen at call centers. If you ever noticed, call centers employees never talk in the same manner, their way of pitching/talking to the customers changes with customers. Now, this does happen with common people too, but how is this relevant to call centers? Here is your answer, the employees recognize customers’ emotions from speech, so they can improve their service and convert more people. In this way, they are using speech emotion recognition. So, let’s discuss this project in detail.

Speech emotion recognition is a simple Python mini-project, which you are going to practice with DataFlair. Before, I explain to you the terms related to this mini python project, make sure you bookmarked the complete list of Python Projects.

**What is Speech Emotion Recognition?**

Speech Emotion Recognition, abbreviated as SER, is the act of attempting to recognize human emotion and affective states from speech. This is capitalizing on the fact that voice often reflects underlying emotion through tone and pitch. This is also the phenomenon that animals like dogs and horses employ to be able to understand human emotion.

SER is tough because emotions are subjective and annotating audio is challenging.

**What is librosa?**

librosa is a Python library for analyzing audio and music. It has a flatter package layout, standardizes interfaces and names, backwards compatibility, modular functions, and readable code. Further, in this Python mini-project, we demonstrate how to install it (and a few other packages) with pip.

**What is JupyterLab?**

JupyterLab is an open-source, web-based UI for Project Jupyter and it has all basic functionalities of the Jupyter Notebook, like notebooks, terminals, text editors, file browsers, rich outputs, and more. However, it also provides improved support for third party extensions.

**Speech Emotion Recognition – Objective**

To build a model to recognize emotion from speech using the librosa and sklearn libraries and the RAVDESS dataset.

Speech Emotion Recognition – About the Python Mini Project

In this Python mini project, we will use the libraries librosa, soundfile, and sklearn (among others) to build a model using an MLPClassifier. This will be able to recognize emotion from sound files. We will load the data, extract features from it, then split the dataset into training and testing sets. Then, we’ll initialize an MLPClassifier and train the model. Finally, we’ll calculate the accuracy of our model.

**The Dataset**

For this Python mini project, we’ll use the RAVDESS dataset; this is the Ryerson Audio-Visual Database of Emotional Speech and Song dataset, and is free to download. This dataset has 7356 files rated by 247 individuals 10 times on emotional validity, intensity, and genuineness. The entire dataset is 24.8GB from 24 actors, but we’ve lowered the sample rate on all the files, and you can download it here.

**Prerequisites**

You’ll need to install the following libraries with pip:

pip install librosa soundfile numpy sklearn pyaudio

If you run into issues installing librosa with pip, you can try it with conda.

**Steps for speech emotion recognition python projects**

1. Make the necessary imports:

import librosa

import soundfile

import os, glob, pickle

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import accuracy\_score

Screenshot:

python project - import data

2. Define a function extract\_feature to extract the mfcc, chroma, and mel features from a sound file. This function takes 4 parameters- the file name and three Boolean parameters for the three features:

mfcc: Mel Frequency Cepstral Coefficient, represents the short-term power spectrum of a sound

chroma: Pertains to the 12 different pitch classes

mel: Mel Spectrogram Frequency

Learn more about Python Sets and Booleans

Open the sound file with soundfile.SoundFile using with-as so it’s automatically closed once we’re done. Read from it and call it X. Also, get the sample rate. If chroma is True, get the Short-Time Fourier Transform of X.

Let result be an empty numpy array. Now, for each feature of the three, if it exists, make a call to the corresponding function from librosa.feature (eg- librosa.feature.mfcc for mfcc), and get the mean value. Call the function hstack() from numpy with result and the feature value, and store this in result. hstack() stacks arrays in sequence horizontally (in a columnar fashion). Then, return the result.

#DataFlair - Extract features (mfcc, chroma, mel) from a sound file

def extract\_feature(file\_name, mfcc, chroma, mel):

with soundfile.SoundFile(file\_name) as sound\_file:

X = sound\_file.read(dtype="float32")

sample\_rate=sound\_file.samplerate

if chroma:

stft=np.abs(librosa.stft(X))

result=np.array([])

if mfcc:

mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample\_rate, n\_mfcc=40).T, axis=0)

result=np.hstack((result, mfccs))

if chroma:

chroma=np.mean(librosa.feature.chroma\_stft(S=stft, sr=sample\_rate).T,axis=0)

result=np.hstack((result, chroma))

if mel:

mel=np.mean(librosa.feature.melspectrogram(X, sr=sample\_rate).T,axis=0)

result=np.hstack((result, mel))

return result

-

**Python Speech Emotion recognition projection**

3. Now, let’s define a dictionary to hold numbers and the emotions available in the RAVDESS dataset, and a list to hold those we want- calm, happy, fearful, disgust.

#DataFlair - Emotions in the RAVDESS dataset

emotions={

'01':'neutral',

'02':'calm',

'03':'happy',

'04':'sad',

'05':'angry',

'06':'fearful',

'07':'disgust',

'08':'surprised'

}

#DataFlair - Emotions to observe

observed\_emotions=['calm', 'happy', 'fearful', 'disgust']

4. Now, let’s load the data with a function load\_data() – this takes in the relative size of the test set as parameter. x and y are empty lists; we’ll use the glob() function from the glob module to get all the pathnames for the sound files in our dataset. The pattern we use for this is: “D:\\DataFlair\\ravdess data\\Actor\_\*\\\*.wav”. This is because our dataset looks like this:

Using our emotions dictionary, this number is turned into an emotion, and our function checks whether this emotion is in our list of observed\_emotions; if not, it continues to the next file. It makes a call to extract\_feature and stores what is returned in ‘feature’. Then, it appends the feature to x and the emotion to y. So, the list x holds the features and y holds the emotions. We call the function train\_test\_split with these, the test size, and a random state value, and return that.

#DataFlair - Load the data and extract features for each sound file

def load\_data(test\_size=0.2):

x,y=[],[]

for file in glob.glob("D:\\DataFlair\\ravdess data\\Actor\_\*\\\*.wav"):

file\_name=os.path.basename(file)

emotion=emotions[file\_name.split("-")[2]]

if emotion not in observed\_emotions:

continue

feature=extract\_feature(file, mfcc=True, chroma=True, mel=True)

x.append(feature)

y.append(emotion)

return train\_test\_split(np.array(x), y, test\_size=test\_size, random\_state=9)

Screenshot:

python data science project

5. Time to split the dataset into training and testing sets! Let’s keep the test set 25% of everything and use the load\_data function for this.

#DataFlair - Split the dataset

x\_train,x\_test,y\_train,y\_test=load\_data(test\_size=0.25)

Screenshot:

split speech image recognition dataset

6. Observe the shape of the training and testing datasets:

#DataFlair - Get the shape of the training and testing datasets

print((x\_train.shape[0], x\_test.shape[0]))

Screenshot:

get shape - python project

7. And get the number of features extracted.

#DataFlair - Get the number of features extracted

print(f'Features extracted: {x\_train.shape[1]}')

Output Screenshot:

Python project

8. Now, let’s initialize an MLPClassifier. This is a Multi-layer Perceptron Classifier; it optimizes the log-loss function using LBFGS or stochastic gradient descent. Unlike SVM or Naive Bayes, the MLPClassifier has an internal neural network for the purpose of classification. This is a feedforward ANN model.

#DataFlair - Initialize the Multi Layer Perceptron Classifier

model=MLPClassifier(alpha=0.01, batch\_size=256, epsilon=1e-08, hidden\_layer\_sizes=(300,), learning\_rate='adaptive', max\_iter=500)

Screenshot:

initialize MLP in python projects

9. Fit/train the model.

#DataFlair - Train the model

model.fit(x\_train,y\_train)

Output Screenshot:

train model simple python project

10. Let’s predict the values for the test set. This gives us y\_pred (the predicted emotions for the features in the test set).

#DataFlair - Predict for the test set

y\_pred=model.predict(x\_test)

Screenshot:

predict- simple python project

11. To calculate the accuracy of our model, we’ll call up the accuracy\_score() function we imported from sklearn. Finally, we’ll round the accuracy to 2 decimal places and print it out.

#DataFlair - Calculate the accuracy of our model

accuracy=accuracy\_score(y\_true=y\_test, y\_pred=y\_pred)

#DataFlair - Print the accuracy

print("Accuracy: {:.2f}%".format(accuracy\*100))

**Summary**

In this Python mini project, we learned to recognize emotions from speech. We used an MLPClassifier for this and made use of the soundfile library to read the sound file, and the librosa library to extract features from it. As you’ll see, the model delivered an accuracy of 72.4%. That’s good enough for us yet.