Speech Emotion Recognition using Librosa

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.

Dataset on Google Drive: https://drive.google.com/file/d/1wWsrN2Ep7x6lWqOXfr4rpKGYrJhWc8z7/view

In [4]:
```
#Connect your Drive with Colab
from google.colab import drive
drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=Tr
ue).
```

In [5]:
```
#Check where your Dataset Zip File is
!ls '/content/drive/My Drive/Important Extras/Data Science Works/_Data Science Work/Speech Emotion Recognition'
```

```
'Speech Emotion Recognition Notebook.ipynb'
 speech-emotion-recognition-ravdess-data.zip
```

In [0]:
```
#Unzip the file contents
!unzip '/content/drive/My Drive/Important Extras/Data Science Works/_Data Science Work/Speech Emotion Recognition/speech-em
```

In [9]:
```
#You can see the zip folder has been extracted
!ls
```

```
Actor_01  Actor_05  Actor_09  Actor_13  Actor_17  Actor_21  drive
Actor_02 Actor_06 Actor_10 Actor_14 Actor_18 Actor_22 sample_data
Actor_03 Actor_07 Actor_11 Actor_15 Actor_19 Actor_23
Actor_04  Actor_08  Actor_12  Actor_16  Actor_20  Actor_24
```

In [10]:
```
#Install Librosa and SoundFile to your Machine
!pip install librosa soundfile
```

```
Requirement already satisfied: librosa in /usr/local/lib/python3.6/dist-packages (0.6.3)
Requirement already satisfied: soundfile in /usr/local/lib/python3.6/dist-packages (0.10.3.post1)
Requirement already satisfied: scikit-learn!=0.19.0,>=0.14.0 in /usr/local/lib/python3.6/dist-packages (from librosa) (0.22.2.
post1)
Requirement already satisfied: resampy>=0.2.0 in /usr/local/lib/python3.6/dist-packages (from librosa) (0.2.2)
Requirement already satisfied: six>=1.3 in /usr/local/lib/python3.6/dist-packages (from librosa) (1.12.0)
Requirement already satisfied: numpy>=1.8.0 in /usr/local/lib/python3.6/dist-packages (from librosa) (1.18.4)
Requirement already satisfied: decorator>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from librosa) (4.4.2)
Requirement already satisfied: numba>=0.38.0 in /usr/local/lib/python3.6/dist-packages (from librosa) (0.48.0)
Requirement already satisfied: audioread>=2.0.0 in /usr/local/lib/python3.6/dist-packages (from librosa) (2.1.8)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-packages (from librosa) (1.4.1)
Requirement already satisfied: joblib>=0.12 in /usr/local/lib/python3.6/dist-packages (from librosa) (0.15.1)
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.6/dist-packages (from soundfile) (1.14.0)
Requirement already satisfied: llvmlite<0.32.0,>=0.31.0dev0 in /usr/local/lib/python3.6/dist-packages (from numba>=0.38.0->lib
rosa) (0.31.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from numba>=0.38.0->librosa) (46.4.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.6/dist-packages (from cffi>=1.0->soundfile) (2.20)
```

In [0]:
```
#Import All Important Libraries
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, confusion_matrix
```

In [0]:
```
#function for extracting mfcc, chroma, and mel features from 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
```

In [0]:
```
#Define the motions dictionary
emotions = {
    '01':'neutral',
```

```
    '02':'calm',
    '03':'happy',
    '04':'sad',
    '05':'angry',
    '06':'fearful',
    '07':'disgust',
    '08':'surprised'
}

#Emotions we want to observe
observed_emotions = ['calm', 'happy', 'fearful', 'disgust']
```

In [0]:
```
#Load the data and extract features for each sound file
def load_data(test_size = 0.2):
  x, y = [], []
  for folder in glob.glob('/content/Actor_*'):
    print(folder)
    for file in glob.glob(folder + '/*.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)
```

In [15]:
```
x_train,x_test,y_train,y_test=load_data(test_size=0.2)
```

```
/content/Actor_01
/content/Actor_13
/content/Actor_03
/content/Actor_04
/content/Actor_05
/content/Actor_18
/content/Actor_07
/content/Actor_02
/content/Actor_14
/content/Actor_21
/content/Actor_08
/content/Actor_15
/content/Actor_17
/content/Actor_10
/content/Actor_16
/content/Actor_11
/content/Actor_09
/content/Actor_06
/content/Actor_24
/content/Actor_20
/content/Actor_23
/content/Actor_12
/content/Actor_19
/content/Actor_22
```

In [16]:
```
#Shape of train and test set and Number of features extracted
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
```

```
(614, 154)
Features extracted: 180
```

In [0]:
```
#Initialise Multi Layer Perceptron Classifier
model = MLPClassifier(alpha = 0.01, batch_size = 256, epsilon = 1e-08, hidden_layer_sizes = (300,), learning_rate = 'adapti
```

In [20]:
```
model.fit(x_train, y_train)
```

Out[20]:
```
MLPClassifier(activation='relu', alpha=0.01, batch_size=256, beta_1=0.9,beta_2=0.999, early_stopping=False, epsilon=1e-08,
      hidden_layer_sizes=(300,), learning_rate='adaptive',
      learning_rate_init=0.001, max_fun=15000, max_iter=500,
      momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
      power_t=0.5, random_state=None, shuffle=True, solver='adam',
      tol=0.0001, validation_fraction=0.1, verbose=False,
      warm_start=False)
```

In [0]:
```
#Predict for the test set
y_pred = model.predict(x_test)
```

In [26]:
```
#Calculate Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 75.97%
```