```
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
from sklearn.metrics import confusion_matrix

from google.colab import drive
drive.mount('/content/drive')

#AUDIO_FILEPATH = '/Users/kunal/OneDrive/Documents/1. Stanford University/Sophomore Year/Pers
AUDIO_FILEPATH = '/content/drive/MyDrive/Colab Notebooks/ravdess-data/Actor_*/*.wav'

#Emotions in the RAVDESS dataset
emotions={
  '01':'neutral',
  '02':'calm',
  '03':'happy',
  '04':'sad',
  '05':'angry',
  '06':'fearful',
  '07':'disgust',
  '08':'surprised'
}

#Emotions to observe
observed_emotions = ['happy', 'sad', 'angry', 'fearful', 'disgust', 'surprised']

#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

#Load the data from the audio files and extract their features
```

```
def load_data(test_size):
    features_list, emotions_list = [], []
    for file in glob.glob(AUDIO_FILEPATH):
        file_name = os.path.basename(file)
        emotion = emotions[file_name.split('-')[2]]
        if emotion in observed_emotions:
            feature = extract_feature(file, mfcc=True, chroma=True, mel=True)
            features_list.append(feature)
            emotions_list.append(emotion)
        ##print(train_test_split(np.array(features_list), np.asarray(emotions_list), test_siz
    return train_test_split(np.array(features_list), np.asarray(emotions_list), test_size=tes

#Split the dataset into training data and test data
x_train, x_test, y_train, y_test=load_data(test_size=0.25)
print('Shape of the training data =', x_train.shape[0])
print('Shape of the test data =', x_test.shape[0])
print('Number of features extracted =', x_train.shape[1])

#Initialize the MLPClassifier model
hidden_layer_sizes = [200 for i in range(5)]
model = MLPClassifier(alpha=0.09, batch_size=32, epsilon=1e-08, hidden_layer_sizes=hidden_lay

#Train the model
model.fit(x_train, y_train)

#Test on training data
y_pred = model.predict(x_train)
accuracy = accuracy_score(y_true=y_train, y_pred=y_pred)
print("Accuracy on training data: {:.2f}%".format(accuracy*100))

#Test on test data
y_pred = model.predict(x_test)
accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy on test data: {:.2f}%".format(accuracy*100))

#Generate a confusion matrix
confusion = confusion_matrix(y_test, y_pred, labels=observed_emotions)
print(confusion)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
Shape of the training data = 864
Shape of the test data = 288
Number of features extracted = 180
Accuracy on training data: 95.49%
Accuracy on test data: 61.81%
[[27  2  7 10  2  1]
 [ 4 28  0 11  4  1]
 [ 3  0 36  0  6  1]
 [ 4  8  0 37  0  1]
 [ 8  4  3  3 30  1]
 [11  1  3  4  7 20]]
```