Dataset

```
import os
Root = "/content/drive/MyDrive/Colab_Notebooks/RAVDESS_Emotional_speech_audio"
os.chdir(Root)
```

```
ls
```

```
     modelForPrediction1.sav
     modelForPrediction.sav
     speech-emotion-recognition-ravdess-data/
     Speech_Emotion_Recognition_with_librosa.ipynb
     standardScalar.sav
```

```
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
```

```
#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
```

```
# 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=['calm', 'happy', 'fearful', 'disgust']
```

```
#Load the data and extract features for each sound file
def load_data(test_size=0.2):
    x,y=[],[]
    for file in glob.glob("/content/drive/MyDrive/Colab_Notebooks/RAVDESS_Emotional_speech_audio/speech-emotion-recognition-ravdess-data/Acto
        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)
```

```
#Split the dataset
x_train,x_test,y_train,y_test=load_data(test_size=0.25)
```

```
x_train
```

```
array([[-6.02389954e+02,  5.97717743e+01,  8.60734844e+00, ...,
         2.24425294e-05,  7.05290176e-06,  3.74911019e-06],
       [-6.64690369e+02,  6.82226181e+01,  6.91438007e+00, ...,
         1.92348180e-05,  1.16888250e-05,  1.09572538e-05],
       [-5.56770630e+02,  3.49958611e+01, -1.21606884e+01, ...,
         1.56850641e-04,  9.86818704e-05,  6.10335883e-05],
       ...,
       [-6.41358337e+02,  4.56047516e+01,  3.17263484e-01, ...,
         3.32857708e-05,  2.42486913e-05,  1.74304023e-05],
       [-6.41742493e+02,  3.81749878e+01, -8.41347885e+00, ...,
         3.26658337e-05,  2.97957540e-05,  2.17277611e-05],
       [-7.70246155e+02,  3.43720894e+01,  5.50091887e+00, ...,
         4.58828936e-06,  2.15270302e-06,  1.44739533e-06]])
```

```
#Get the shape of the training and testing datasets
print((x_train.shape[0], x_test.shape[0]))
```

```
(576, 192)
```

```
#Get the number of features extracted
print(f'Features extracted: {x_train.shape[1]}')
```

```
Features extracted: 180
```

```
#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)
```

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

```
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)
```

```
#Predict for the test set
y_pred=model.predict(x_test)
```

```
y_pred
```

```
array(['calm', 'disgust', 'calm', 'happy', 'calm', 'happy', 'disgust',
       'calm', 'happy', 'fearful', 'calm', 'fearful', 'disgust',
       'fearful', 'fearful', 'calm', 'happy', 'fearful', 'disgust',
       'calm', 'happy', 'happy', 'fearful', 'happy', 'happy', 'calm',
       'disgust', 'calm', 'happy', 'happy', 'happy', 'calm', 'happy',
       'happy', 'fearful', 'calm', 'disgust', 'calm', 'happy', 'disgust',
       'calm', 'disgust', 'happy', 'happy', 'calm', 'fearful', 'calm',
       'fearful', 'calm', 'happy', 'happy', 'disgust', 'fearful', 'calm',
       'fearful', 'happy', 'fearful', 'disgust', 'disgust', 'calm',
       'happy', 'fearful', 'disgust', 'fearful', 'fearful', 'calm',
       'happy', 'happy', 'calm', 'happy', 'fearful', 'calm', 'calm',
       'calm', 'disgust', 'calm', 'fearful', 'happy', 'happy', 'disgust',
       'happy', 'fearful', 'fearful', 'fearful', 'calm', 'calm', 'calm',
       'disgust', 'happy', 'calm', 'happy', 'calm', 'calm', 'fearful',
       'calm', 'calm', 'calm', 'fearful', 'happy', 'fearful', 'calm',
       'calm', 'calm', 'disgust', 'fearful', 'happy', 'disgust', 'happy',
       'fearful', 'calm', 'happy', 'fearful', 'fearful', 'calm', 'happy',
       'calm', 'disgust', 'calm', 'calm', 'disgust', 'calm', 'fearful',
       'calm', 'calm', 'calm', 'disgust', 'fearful', 'calm', 'happy',
       'fearful', 'calm', 'fearful', 'happy', 'fearful', 'calm',
       'fearful', 'happy', 'happy', 'happy', 'fearful', 'disgust',
       'fearful', 'disgust', 'calm', 'fearful', 'disgust', 'happy',
       'disgust', 'disgust', 'calm', 'calm', 'happy', 'fearful', 'calm',
       'fearful', 'calm', 'disgust', 'happy', 'happy', 'calm', 'disgust',
       'calm', 'fearful', 'disgust', 'happy', 'calm', 'calm', 'calm',
       'disgust', 'fearful', 'calm', 'happy', 'fearful', 'happy', 'calm',
       'calm', 'fearful', 'disgust', 'happy', 'disgust', 'calm', 'calm',
```

```
            'calm', 'disgust', 'disgust', 'calm', 'calm', 'fearful', 'happy',
            'disgust', 'fearful', 'happy'], dtype='<U7')
```

```
#Calculate the accuracy of our model
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

#Print the accuracy
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 73.44%
```

```
from sklearn.metrics import accuracy_score, f1_score
```

```
f1_score(y_test, y_pred,average=None)
```

```
array([0.8173913 , 0.65822785, 0.70967742, 0.72164948])
```

```
import pandas as pd
df=pd.DataFrame({'Actual': y_test, 'Predicted':y_pred})
df.head(20)
```

	Actual	Predicted
0	calm	calm
1	disgust	disgust
2	calm	calm
3	happy	happy
4	happy	calm
5	happy	happy
6	disgust	disgust
7	disgust	calm
8	happy	happy
9	fearful	fearful
10	calm	calm
11	disgust	fearful
12	disgust	disgust
13	fearful	fearful
14	disgust	fearful
15	calm	calm
16	happy	happy
17	fearful	fearful
18	disgust	disgust
19	calm	calm

```
import pickle
# Writing different model files to file
with open( 'modelForPrediction1.sav', 'wb') as f:
    pickle.dump(model,f)
```

```
filename = 'modelForPrediction1.sav'
loaded_model = pickle.load(open(filename, 'rb')) # loading the model file from the storage

feature=extract_feature("/content/drive/MyDrive/Colab_Notebooks/RAVDESS_Emotional_speech_audio/speech-emotion-recognition-ravdess-data/Actor_

feature=feature.reshape(1,-1)

prediction=loaded_model.predict(feature)
prediction
```

```
array(['disgust'], dtype='<U7')
```

```
feature
```