Mount The Drive

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

```
Mounted at /content/drive
```

Load The Datasets

```
import os
Root = "/content/drive/MyDrive/Project Code Space/speech-emotion-recognition-ravdess-data"
os.chdir(Root)
```

List All The Datasets

```
ls
```

```
Actor_01/  Actor_06/  Actor_11/  Actor_16/  Actor_21/
Actor_02/  Actor_07/  Actor_12/  Actor_17/  Actor_22/
Actor_03/  Actor_08/  Actor_13/  Actor_18/  Actor_23/
Actor_04/  Actor_09/  Actor_14/  Actor_19/  Actor_24/
Actor_05/  Actor_10/  Actor_15/  Actor_20/  modelForPrediction1.sav
```

Import The Necessary 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
```

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/Project Code Space/speech-emotion-recognition-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)
```

Split the dataset

```
x_train,x_test,y_train,y_test=load_data(test_size=0.25)
```

```
x_train
```

```
array([[-5.07093414e+02,  4.41557541e+01, -2.49317608e+01, ...,
         1.93066167e-04,  1.33097477e-04,  7.82514253e-05],
       [-4.70447571e+02,  1.76569138e+01, -2.86761608e+01, ...,
         5.40096022e-04,  4.01184778e-04,  4.11149638e-04],
       [-6.47872559e+02,  4.41566811e+01, -1.16800661e+01, ...,
         2.45946485e-05,  1.61880325e-05,  3.83465203e-06],
       ...,
       [-6.64179810e+02,  4.10878143e+01, -1.48883247e+01, ...,
         2.27178152e-05,  1.70372368e-05,  1.20926697e-05],
       [-5.88531982e+02,  6.03003578e+01,  1.75049515e+01, ...,
         1.34000598e-04,  6.59308062e-05,  1.95767807e-05],
       [-7.09077148e+02,  6.64060516e+01, -6.71996927e+00, ...,
         1.07113351e-06,  6.70256213e-07,  4.15542047e-07]])
```

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(alpha=0.01, batch_size=256, hidden_layer_sizes=(300,),
              learning_rate='adaptive', max_iter=500)
```

Predict for the test set

```
y_pred=model.predict(x_test)
```

```
y_pred
```

```
array(['disgust', 'happy', 'disgust', 'disgust', 'disgust', 'happy',
       'disgust', 'disgust', 'fearful', 'disgust', 'happy', 'fearful',
       'happy', 'fearful', 'disgust', 'happy', 'disgust', 'calm',
       'disgust', 'fearful', 'disgust', 'happy', 'happy', 'disgust',
       'disgust', 'calm', 'disgust', 'disgust', 'happy', 'calm',
       'disgust', 'happy', 'happy', 'disgust', 'disgust', 'calm', 'calm',
       'disgust', 'disgust', 'fearful', 'disgust', 'happy', 'happy',
       'happy', 'disgust', 'calm', 'fearful', 'calm', 'calm', 'happy',
       'happy', 'happy', 'disgust', 'fearful', 'happy', 'fearful',
       'disgust', 'disgust', 'happy', 'disgust', 'disgust', 'calm',
       'disgust', 'happy', 'disgust', 'disgust', 'happy', 'disgust',
       'disgust', 'fearful', 'disgust', 'disgust', 'disgust', 'fearful',
       'disgust', 'fearful', 'fearful', 'disgust', 'calm', 'disgust',
       'fearful', 'calm', 'fearful', 'disgust', 'fearful', 'disgust',
       'disgust', 'happy', 'disgust', 'disgust', 'fearful', 'disgust',
       'disgust', 'disgust', 'disgust', 'disgust', 'calm', 'happy',
       'disgust', 'disgust', 'disgust', 'fearful', 'disgust', 'disgust',
       'disgust', 'disgust', 'disgust', 'disgust', 'happy', 'happy',
       'happy', 'happy', 'calm', 'disgust', 'disgust', 'calm', 'happy',
       'disgust', 'disgust', 'disgust', 'disgust', 'calm', 'disgust',
       'disgust', 'disgust', 'fearful', 'disgust', 'fearful', 'fearful',
       'disgust', 'fearful', 'disgust', 'calm', 'disgust', 'happy',
       'happy', 'disgust', 'fearful', 'fearful', 'disgust', 'calm',
       'happy', 'disgust', 'disgust', 'happy', 'disgust', 'disgust',
       'disgust', 'happy', 'happy', 'calm', 'disgust', 'calm', 'disgust',
       'disgust', 'happy', 'happy', 'happy', 'calm', 'happy', 'fearful',
       'disgust', 'fearful', 'calm', 'happy', 'happy', 'disgust',
       'disgust', 'disgust', 'calm', 'fearful', 'disgust', 'disgust',
       'happy', 'happy', 'disgust', 'happy', 'fearful', 'happy',
       'disgust', 'calm', 'disgust', 'calm', 'calm', 'fearful', 'happy',
       'calm', 'fearful', 'happy', 'fearful', 'disgust', 'calm'],
      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: 61.46%
```

To Find The Proper Value Of Prediction

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

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

```
array([0.61176471, 0.60869565, 0.60869565, 0.63043478])
```

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

	Actual	Predicted
0	calm	disgust
1	happy	happy
2	happy	disgust
3	calm	disgust
4	disgust	disgust
5	calm	happy
6	disgust	disgust
7	happy	disgust
8	fearful	fearful
11	fearful	fearful

Dumping The Model

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

Finding Output

17	calm	calm

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

feature=extract_feature("/content/drive/MyDrive/Project Code Space/speech-emotion-recognition-ravdess-data/Actor_05/03-01-02-01-01-01-05.

feature=feature.reshape(1,-1)

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

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