SER Phase I Prerequisitive -> pip install librosa soundfile numpy sklearn pyaudio

```
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
```

```
#Extracting features (mfcc, chroma, mel) from audio files
def feature_extraction(file_name, mfcc, chroma, mel):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        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))
        return result
```

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

```
# Emotions to recognize
emotions_to_recognize=['angry','disgust','surprised','calm','neutral','happy','sad','fearful']
```

```
# Data Loading and feature extraction for each sound file
def load_data(test_size=0.2):
    x,y=[],[]
    for file in glob.glob('/content/drive/MyDrive/RAVDESS/**/*.wav'):
        file_name=os.path.basename(file)
        emotion=emotions[file_name.split("-")[2]]
        if emotion not in emotions_to_recognize:
            continue
        feature=feature_extraction(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)
```

```
# Dataset splitting for training and testing
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
```

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

```
(1152, 288)
```

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

```
Features extracted: 40
```

```
# Model Initialization -> 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)
```

```
# Model Training
model.fit(x_train,y_train)
```

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

```
# Model Prediction
y_pred=model.predict(x_test)
```

```
# Model Accuracy
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))

    Accuracy: 46.88%
```