```
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
```

```
#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']
```

Feature - Accuracy Tonnetz: 15.97%, Root Mean Square: 27.43%, chroma_spectral_centroid: 12.5%, chroma_spectral_bandwidth: 12.5%, chroma_spectral_contrast: 23.26%, chroma_spectral_flatness: 11.11%, chroma_spectral_rolloff: 12.5%, chroma_spectral_poly_features: 30.21% (Order: 2), chroma_CENS: 26.74%, CQT: 27.08 %, Zero Crossing Data: 20.14%, Mel spectrogram: 40.28%, MFCC: 48.61%, Chroma_STFT: 16.67%

Starting with MFCC, Mel Spectrogram and Chroma_spectral_poly_features (48.61%, 40.28% and 30.21%)

```
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Mel Spectogram
        mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
        result=np.hstack((result, mel))
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        return result
```

```
# 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]]
        feature=feature_extraction(file)
        x.append(feature)
        y.append(emotion)
    return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(1152, 288)
Features extracted: 171
Accuracy: 52.08%
```

Let's start trying different combinations

```
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
```

```
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Mel Spectogram
        mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
        result=np.hstack((result, mel))
        # Spectral Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # Root Mean Square
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        return result

x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(1152, 288)
Features extracted: 182
Accuracy: 49.65%
```

Trying another scenario

```
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Mel Spectogram
        mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
        result=np.hstack((result, mel))
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        return result


x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(1152, 288)
Features extracted: 193
Accuracy: 55.90%
```

182 and 193 features - accuracy - 49.65 and 55.90 % Not the curse of dimensionality issue therefore we go forward with these 4 features

```
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Mel Spectogram
```

```
        mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
        result=np.hstack((result, mel))
        # Chroma Spectral Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # Chroma CQT
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        # Chroma CENS
        chroma_cens = np.mean(librosa.feature.chroma_cens(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, chroma_cens))
        return result
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(1152, 288)
Features extracted: 205
Accuracy: 45.14%
```

Could be curse of dimensionality

Trying with Another Data Set - CREMA_D

```
def load_data(test_size=0.2):
    x,y=[],[]
    for file in glob.glob('/content/drive/MyDrive/SER/CREMA_D/*.wav'):
        file_name=os.path.basename(file)
        part=file.split('_')
        temp_emotion=part[2]
        if temp_emotion == 'SAD':
          y.append('sad')
        elif temp_emotion == 'ANG':
          y.append('angry')
        elif temp_emotion == 'DIS':
          y.append('disgust')
        elif temp_emotion == 'FEA':
          y.append('fear')
        elif temp_emotion == 'HAP':
          y.append('happy')
        elif temp_emotion == 'NEU':
          y.append('neutral')
        else:
          y.append('Unknown')
        feature=feature_extraction(file)
        x.append(feature)
        #y.append(emotion)
    return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
```

```
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/pitch.py:153: UserWarning: Trying to estimate tuning from empty frequency set.
  warnings.warn("Trying to estimate tuning from empty frequency set.")
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=512 is too small for input signal of length
  warnings.warn(
(5953, 1489)
Features extracted: 205
Accuracy: 100.00%
```

With CREMA, Accuracy is 100%, Therefore we try with SAVEE Data set

```python
def load_data(test_size=0.2):
    x,y=[],[]
    for file in glob.glob('/content/drive/MyDrive/SER/SAVEE/ALL/*.wav'):
        file_name=os.path.basename(file)
        part=file.split('_')[1]
        temp_emotion=part[:-6]
        if temp_emotion == 'sa':
          y.append('sad')
        elif temp_emotion == 'a':
          y.append('angry')
        elif temp_emotion == 'd':
          y.append('disgust')
        elif temp_emotion == 'f':
          y.append('fear')
        elif temp_emotion == 'h':
          y.append('happy')
        elif temp_emotion == 'n':
          y.append('neutral')
        else:
          y.append('surprise')
        feature=feature_extraction(file)
        x.append(feature)
        #y.append(emotion)
    return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
```

```python
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        return result
```

```python
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
```

```
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(384, 96)
Features extracted: 50
Accuracy: 67.71%
```

Experimenting Multi Feature Fusion to improve accuracy

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        return result
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(384, 96)
Features extracted: 53
Accuracy: 77.08%
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimiz
  warnings.warn(
```

Going forward adding another feature

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # Mel Spectogram
        mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
        result=np.hstack((result, mel))
        return result
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(384, 96)
Features extracted: 181
Accuracy: 66.67%
```

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Mel Spectogram
        mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
        result=np.hstack((result, mel))
        return result
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(384, 96)
Features extracted: 178
Accuracy: 62.50%
```

Now that we see that Mel spectrogram isn't useful in this scenario as the number of features extracted are a bit excess

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        return result
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(384, 96)
Features extracted: 53
Accuracy: 75.00%
```

Experimenting other features

Doing an assessment of number of features extracted across different features in RAVDESS Data Set MFCC - 50 - We have control over it Chroma STFT - 12 Mel Spectrogram - 128 Zero Crossing Rate - 1 Chroma CQT - 12 Chroma CENS - 12 Chroma Poly Features - 3 (Order 2) Chroma Spectral Centroid - 1 Chroma Spectral Bandwidth - 1 Chroma Spectral Contrast - 7 Chroma Spectral Flatness - 1 Chroma Spectral Rolloff - 1 Root Mean Square - 1 Harmonics, Tonnetz - 6

Can we include those feature sets which extracts less features - how do we select it - Let's consider accuracy table we have -> Feature - Accuracy Tonnetz: 15.97%, Root Mean Square: 27.43%, chroma_spectral_centroid: 12.5%, chroma_spectral_bandwidth: 12.5%, chroma_spectral_contrast: 23.26%, chroma_spectral_flatness: 11.11%, chroma_spectral_rolloff: 12.5%, chroma_spectral_poly_features: 30.21% (Order: 2), chroma_CENS: 26.74%, CQT: 27.08 %, Zero Crossing Data: 20.14%, Mel spectrogram: 40.28%, MFCC: 48.61%, Chroma_STFT: 16.67%

```
#Extracting features from audio files
def feature_extraction(file_name):
```

```
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # RMS
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        return result

x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
(384, 96)
Features extracted: 54
Accuracy: 76.04%
```

Let's continue with further experimentation

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # RMS
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        # CQT
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        return result


x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
(384, 96)
Features extracted: 66
```

Let's try to further stretch

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # RMS
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        # CQT
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        # Zero Crossing Rate
        zcr = np.mean(librosa.feature.zero_crossing_rate(y=X).T, axis=0)
        result=np.hstack((result, zcr))
        return result


x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
(384, 96)
Features extracted: 67
```

Let's revert back to old set

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # RMS
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        # CQT
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        # CENS
        chroma_cens = np.mean(librosa.feature.chroma_cens(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, chroma_cens))
        return result
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
(384, 96)
Features extracted: 78
Accuracy: 70.83%
```

Further Try

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # RMS
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        # CQT
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        # Spectral Contrast
        chroma_spectral_contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=sample_rate).T, axis=0)
        result = np.hstack((result, chroma_spectral_contrast))
        return result
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
(384, 96)
Features extracted: 73
Accuracy: 72.92%
```

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # RMS
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        # CQT
```

```
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        # Chroma STFT
        chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
        result=np.hstack((result, chroma))
        return result

x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
(384, 96)
Features extracted: 78
Accuracy: 69.79%
```

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
```

```
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # RMS
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        # CQT
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        ## Tonnetz
        harmonic = np.abs(librosa.effects.harmonic(X))
        tonnetz = np.mean(librosa.feature.tonnetz(y=harmonic, sr=sample_rate).T, axis=0)
        result = np.hstack((result, tonnetz))
        return result


x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
(384, 96)
Features extracted: 72
Accuracy: 73.96%
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Opti
  warnings.warn(
```

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
```

```
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # RMS
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        # CQT
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        # Spectral RollOff
        chroma_spectral_rolloff = np.mean(librosa.feature.spectral_rolloff(S=stft, sr=sample_rate).T, axis=0)
        result = np.hstack((result, chroma_spectral_rolloff))
        return result

x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

Now with all these studies, we can conclude that MFCC + Chroma Spectral Poly Features + RMS + CQT Fusion would be the best option for SAVEE Data Set

```
#Extracting features from audio files
def feature_extraction(file_name):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        result=np.array([])
        # STFT
        stft=np.abs(librosa.stft(X))
        # MFCC
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=50).T, axis=0)
        result=np.hstack((result, mfccs))
        # Chroma Poly Features
        chroma_spectral_poly_features = np.mean(librosa.feature.poly_features(S=stft, order=2).T, axis=0)
        result = np.hstack((result, chroma_spectral_poly_features))
        # RMS
        rms = np.mean(librosa.feature.rms(y=X).T, axis=0)
        result = np.hstack((result, rms))
        # CQT
        cqt = np.mean(librosa.feature.chroma_cqt(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, cqt))
        return result
```

```
x_train,x_test,y_train,y_test= load_data(test_size=0.2)
print((x_train.shape[0], x_test.shape[0]))
print(f'Features extracted: {x_train.shape[1]}')
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/librosa/core/spectrum.py:222: UserWarning: n_fft=1024 is too small for input signal of le
  warnings.warn(
(384, 96)
Features extracted: 66
```

With Savee, The Max accuracy we could achieve is 78.12%