B. Deepak kumar

AIE21028

LAB-7

A1. Use HMM for classification of your speech signal using STFT features.

In [2]:
```
1 !pip install hmmlearn
```

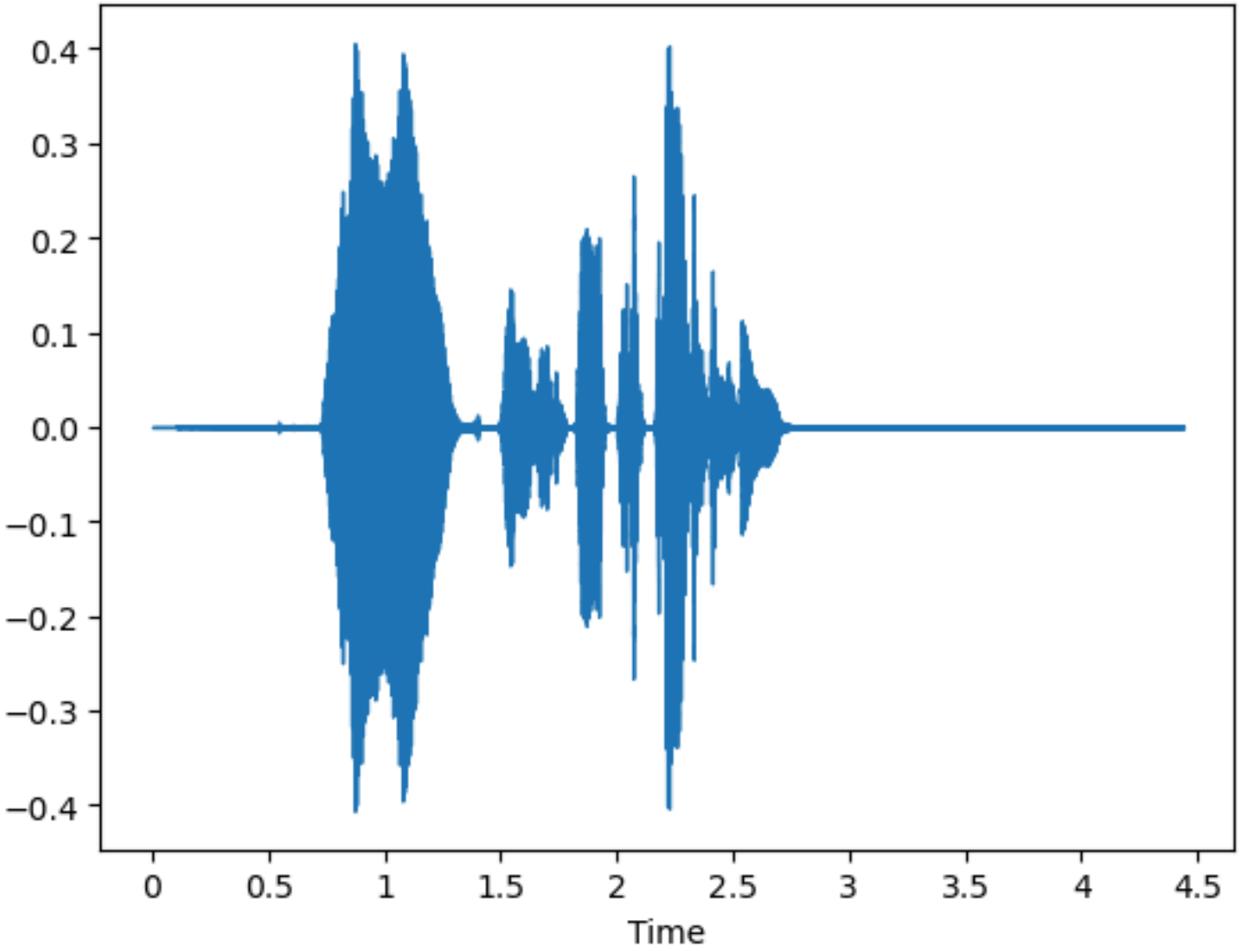
```
Collecting hmmlearn
  Downloading hmmlearn-0.3.2-cp39-cp39-win_amd64.whl (124 kB)
     ------------------------------------ 124.5/124.5 kB 48.1 kB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in c:\users\saide\anaconda3\lib\site-pack
ages (from hmmlearn) (1.21.6)
Requirement already satisfied: scipy>=0.19 in c:\users\saide\anaconda3\lib\site-pack
ages (from hmmlearn) (1.9.1)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in c:\users\saide\anacond
a3\lib\site-packages (from hmmlearn) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\saide\anaconda3\lib
\site-packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\saide\anaconda3\lib\site-pac
kages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.2
```

In [16]:

```
import numpy as np
import librosa
import matplotlib.pyplot as plt
from hmmlearn import hmm
import IPython.display as ipd
import scipy.signal as signal
import scipy.io.wavfile as wavfile
from glob import glob
import seaborn as sns
from scipy.signal import spectrogram
```

In [10]:
```
y, sr = librosa.load('AISPS.wav')
librosa.display.waveshow(y)
```

Out[10]: <librosa.display.AdaptiveWaveplot at 0x1e071b4aa00>



In [11]:
```
a = glob('AISPS.wav')
ipd.Audio(a[0])
```

Out[11]:

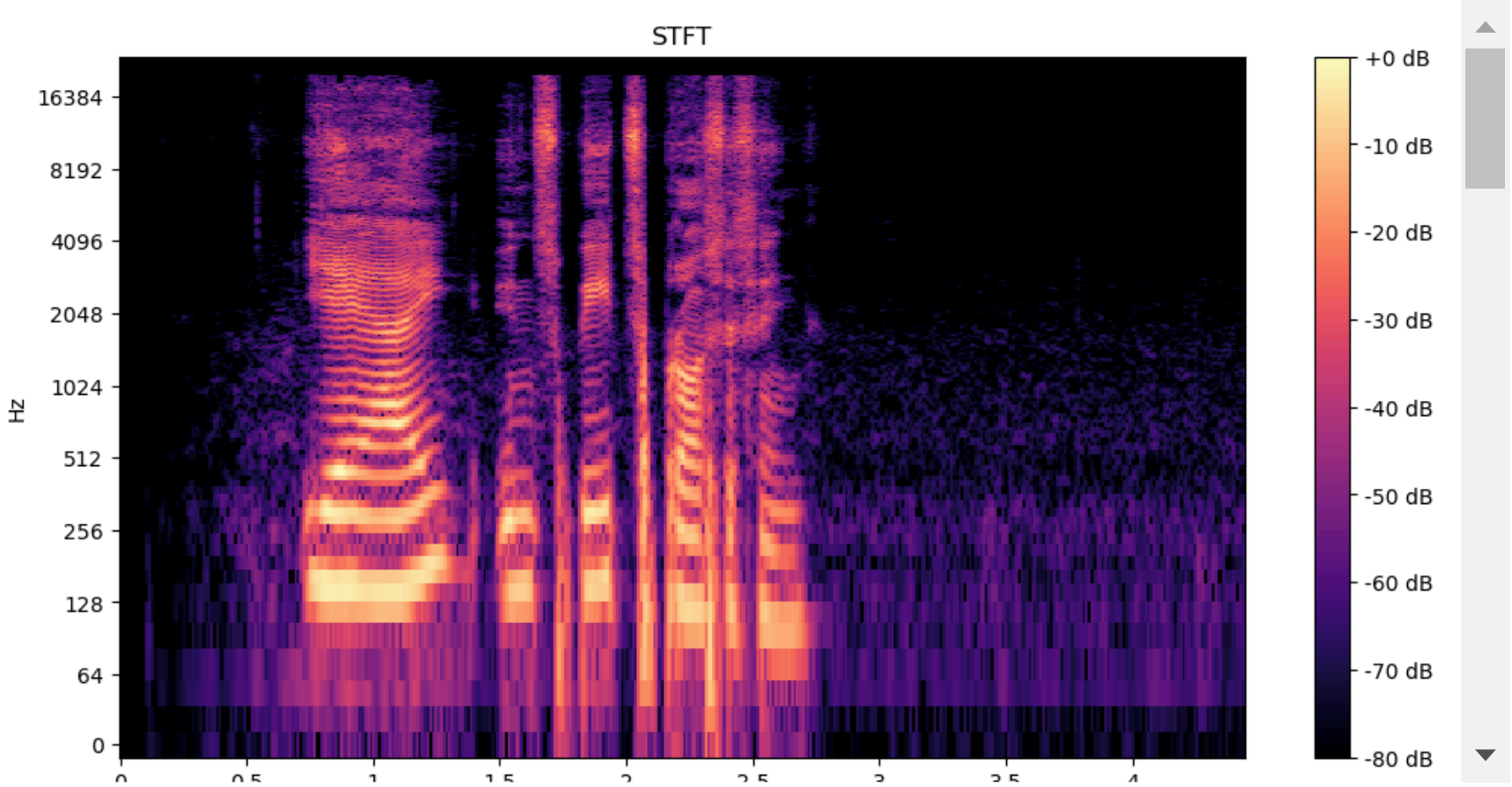
In [22]:

```
def load_audio(file_path):
    y, sr = librosa.load(file_path, sr=None)
    return y, sr

def stft_features(y, sr):
    stft = np.abs(librosa.stft(y))
    return stft

def train_hmm(features, n_components=3, n_iter=100):
    model = hmm.GaussianHMM(n_components=n_components, covariance_type="diag", n_
    model.fit(features)
    return model

def plot_stft(stft, sr):
    plt.figure(figsize=(12, 6))
    librosa.display.specshow(librosa.amplitude_to_db(stft, ref=np.max), sr=sr, x_
    plt.colorbar(format='%+2.0f dB')
    plt.title('STFT')
    plt.show()

def classify_signal(model, features):
    # Predict using the trained HMM model
    labels = model.predict(features.T)  # Transpose features to fit HMM's require
    return labels

```

In [30]:

```python
def main():
    audio_file_path = "AISPS.wav"

    # Load audio
    y, sr = load_audio(audio_file_path)

    # Extract STFT features
    stft = stft_features(y, sr)

    # Plot STFT
    plot_stft(stft, sr)

    # Train HMM
    model = train_hmm(stft.T)  # Transpose stft to fit HMM's requirement

    # Classify signal using trained HMM
    labels = classify_signal(model, stft)

    # Plot the classification result
    plt.figure(figsize=(12, 6))
    plt.plot(np.arange(len(labels)), labels, label='Classified State')
    plt.xlabel('Time')
    plt.ylabel('State')
    plt.title('HMM Classification Result')
    plt.legend()
    plt.show()

    # Print trained model parameters
    print("HMM Model Parameters:")
    print("Transition Matrix:")
    print(model.transmat_)
```

```
    print("Means:")
    print(model.means_)
    print("Covariances:")
    print(model.covars_)


if __name__ == "__main__":
    main()
```



In [ ]:

```
1
```