Importing Modules

In [1]:
```python
import sys
sys.executable
```

Out[1]:
```
'C:\\Users\\hardi\\anaconda3\\python.exe'
```

In [2]:
```python
import librosa
import librosa.display
```

In [3]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import Audio
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```python
import os
paths = []
labels = []
dataset_location = r'C:\Users\hardi\Downloads\archive\TESS Toronto emotional speech set data'

for dirname, _, filenames in os.walk(dataset_location):
    for filename in filenames:
        file_path = os.path.join(dirname, filename)
        paths.append(file_path)
        label = filename.split('_')[-1]
        label = label.split('.')[0]
        labels.append(label.lower())
print('Dataset is loaded')
```

```
Dataset is loaded
```

In [5]:
```python
paths[:5]
```

Out[5]:
```
['C:\\Users\\hardi\\Downloads\\archive\\TESS Toronto emotional speech set data\\OAF_angry\\OAF_back_angry.wav',
 'C:\\Users\\hardi\\Downloads\\archive\\TESS Toronto emotional speech set data\\OAF_angry\\OAF_bar_angry.wav',
 'C:\\Users\\hardi\\Downloads\\archive\\TESS Toronto emotional speech set data\\OAF_angry\\OAF_base_angry.wav',
 'C:\\Users\\hardi\\Downloads\\archive\\TESS Toronto emotional speech set data\\OAF_angry\\OAF_bath_angry.wav',
 'C:\\Users\\hardi\\Downloads\\archive\\TESS Toronto emotional speech set data\\OAF_angry\\OAF_bean_angry.wav']
```

In [6]:
```python
labels[:5]
```

Out[6]:
```
['angry', 'angry', 'angry', 'angry', 'angry']
```

In [7]:
```python
df = pd.DataFrame()
df['speech'] = paths
df['label'] = labels
df.head()
```

Out[7]:

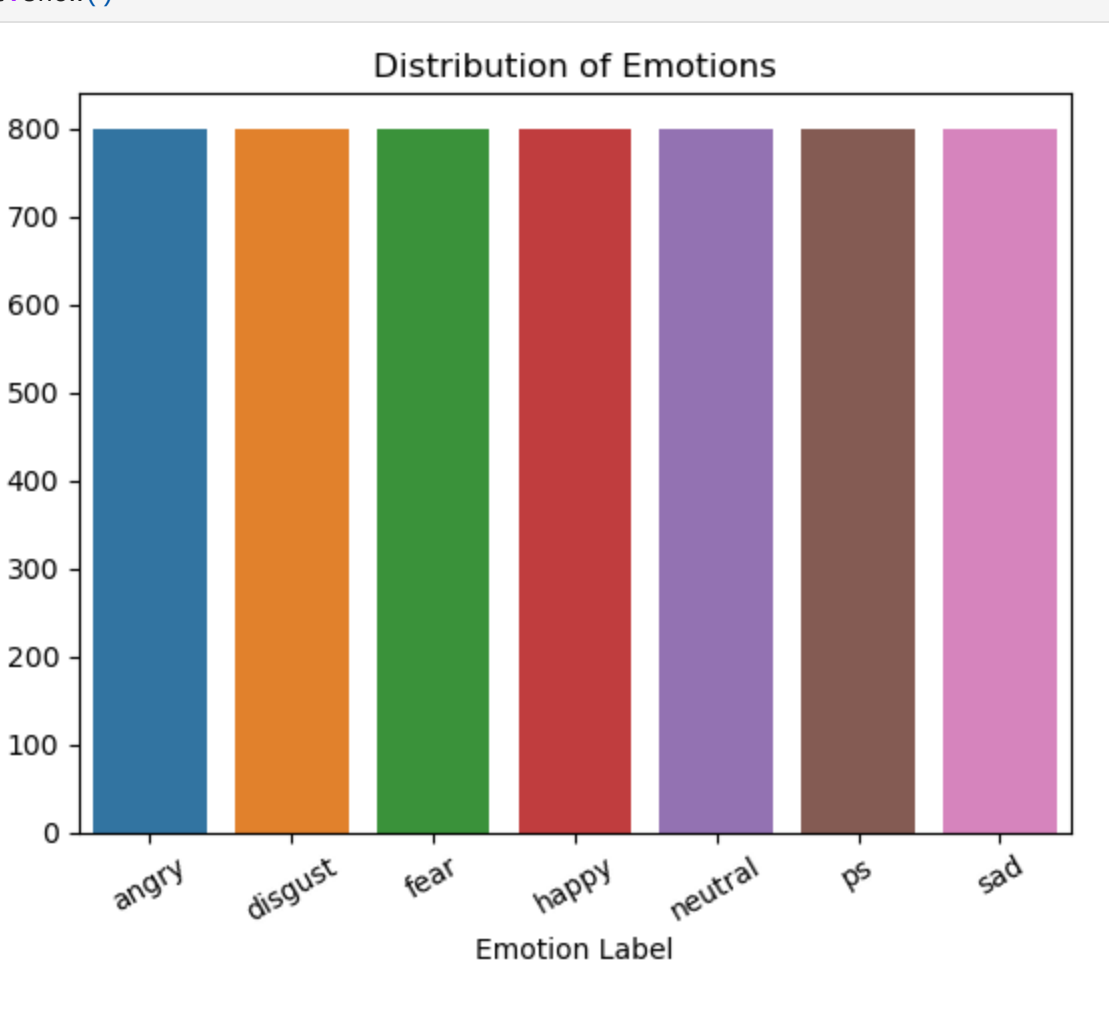
	speech	label
0	C:\Users\hardi\Downloads\archive\TESS Toronto ...	angry
1	C:\Users\hardi\Downloads\archive\TESS Toronto ...	angry
2	C:\Users\hardi\Downloads\archive\TESS Toronto ...	angry
3	C:\Users\hardi\Downloads\archive\TESS Toronto ...	angry
4	C:\Users\hardi\Downloads\archive\TESS Toronto ...	angry

In [8]:
```python
df['label'].value_counts()
```

Out[8]:
```
angry      800
disgust    800
fear       800
happy      800
neutral    800
ps         800
sad        800
Name: label, dtype: int64
```

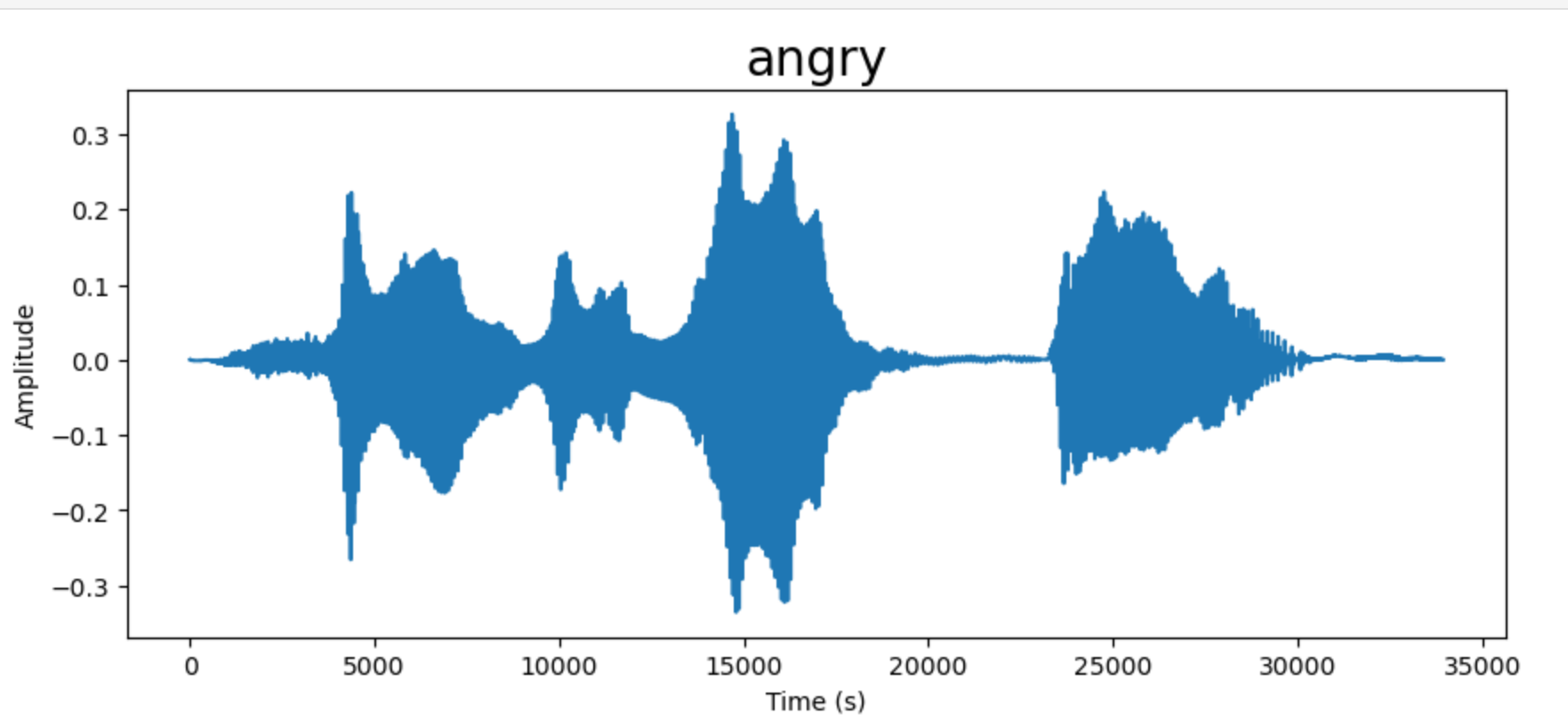
Exploratory Data Analysis

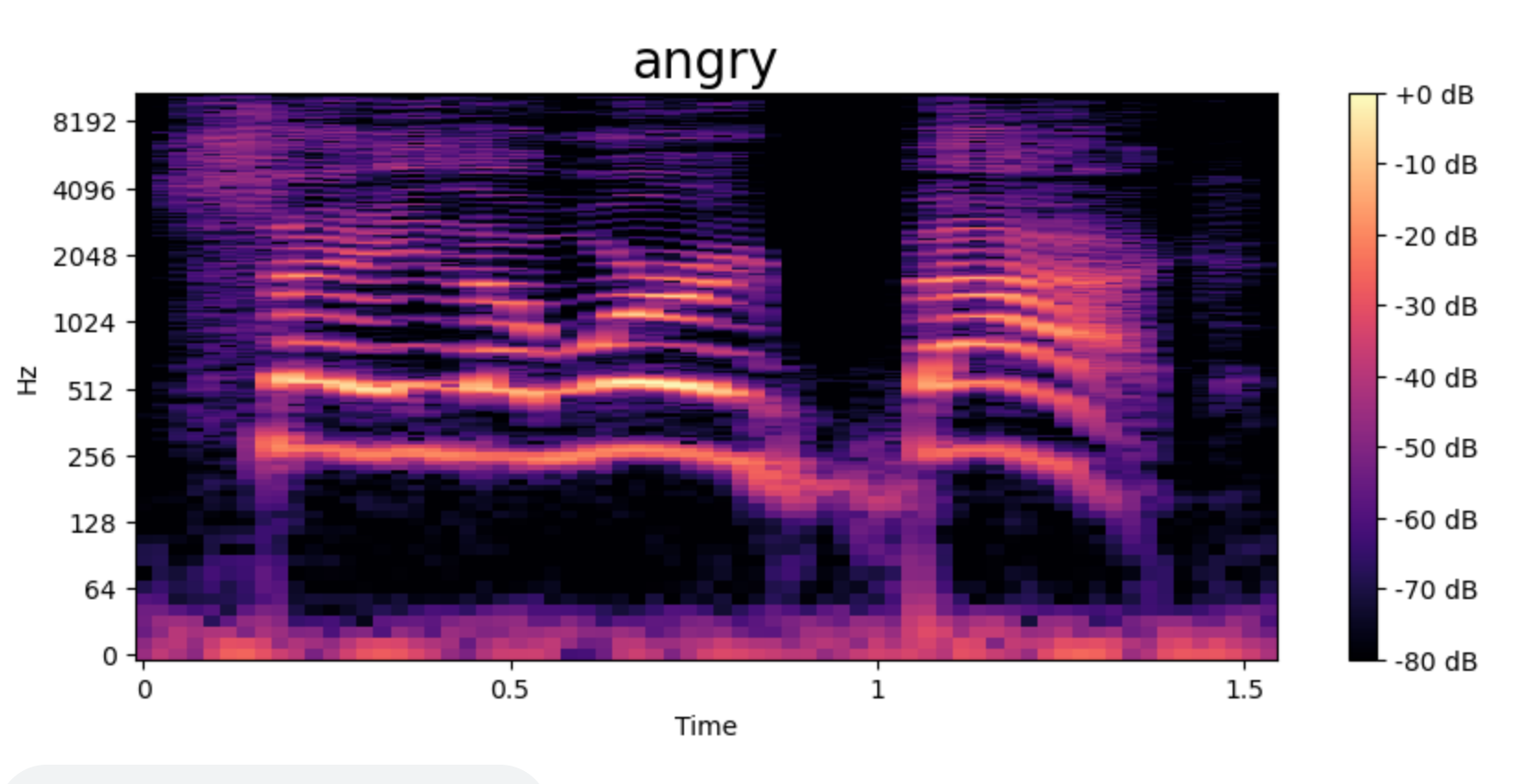
In [9]:
```python
sns.countplot(data=df, x='label')
plt.xlabel('Emotion Label')
plt.ylabel('Count')
plt.title('Distribution of Emotions')
plt.xticks(rotation=30)
plt.show()
```



In [10]:
```python
def waveplot(data, sr, emotion):
    plt.figure(figsize=(10, 4))
    plt.title(emotion, size=20)
    plt.plot(data)
    plt.xlabel('Time (s)')
    plt.ylabel('Amplitude')
    plt.show()
def spectogram(data, sr, emotion):
    plt.figure(figsize=(10, 4))
    plt.title(emotion, size=20)
    D = librosa.amplitude_to_db(np.abs(librosa.stft(data)), ref=np.max)
    librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='log')
    plt.colorbar(format='%+2.0f dB')
    plt.show()
```

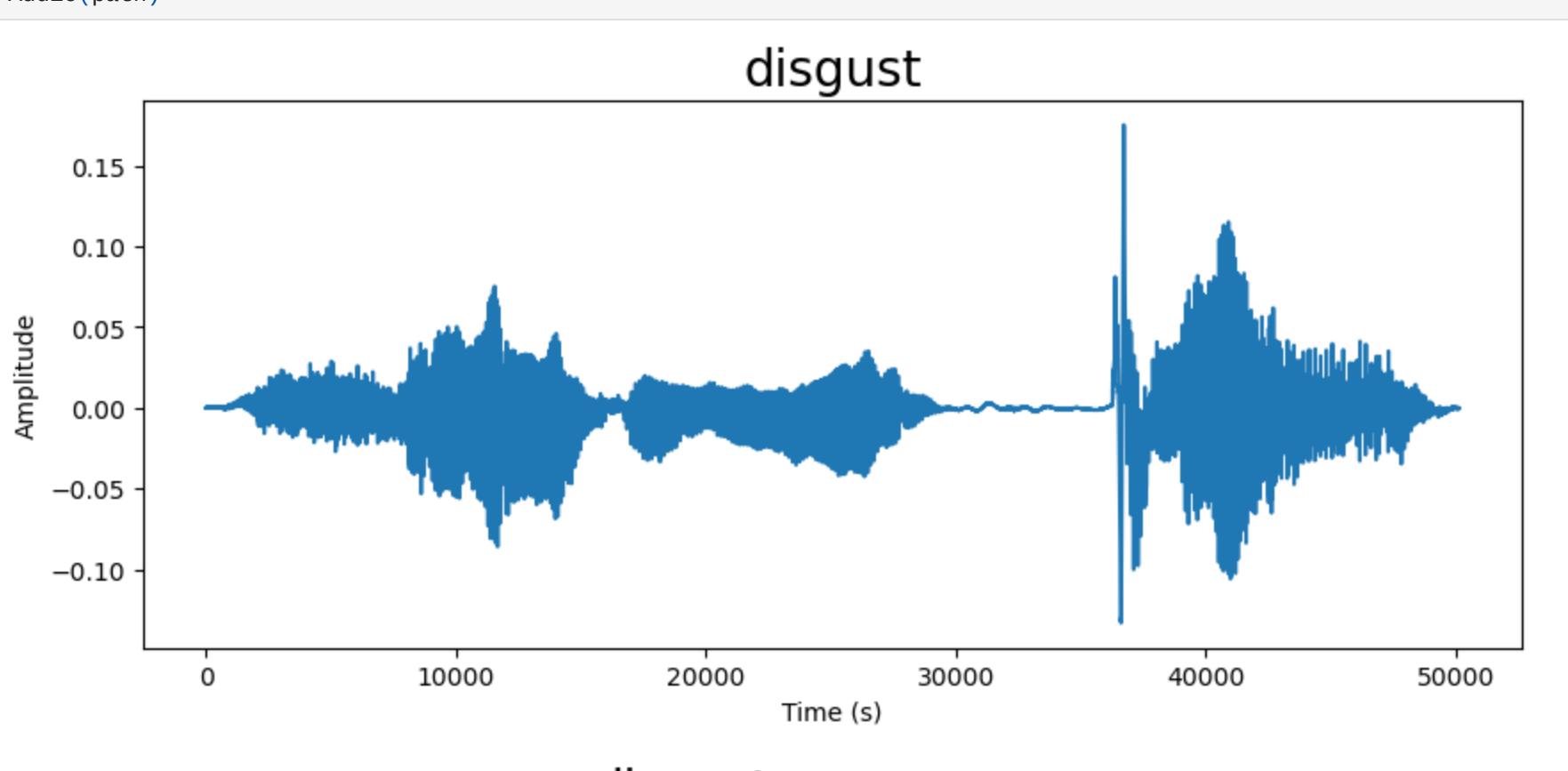
In [11]:
```python
emotion = 'angry'
path = df['speech'][df['label']== emotion][0]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate , emotion)
Audio(path)
```

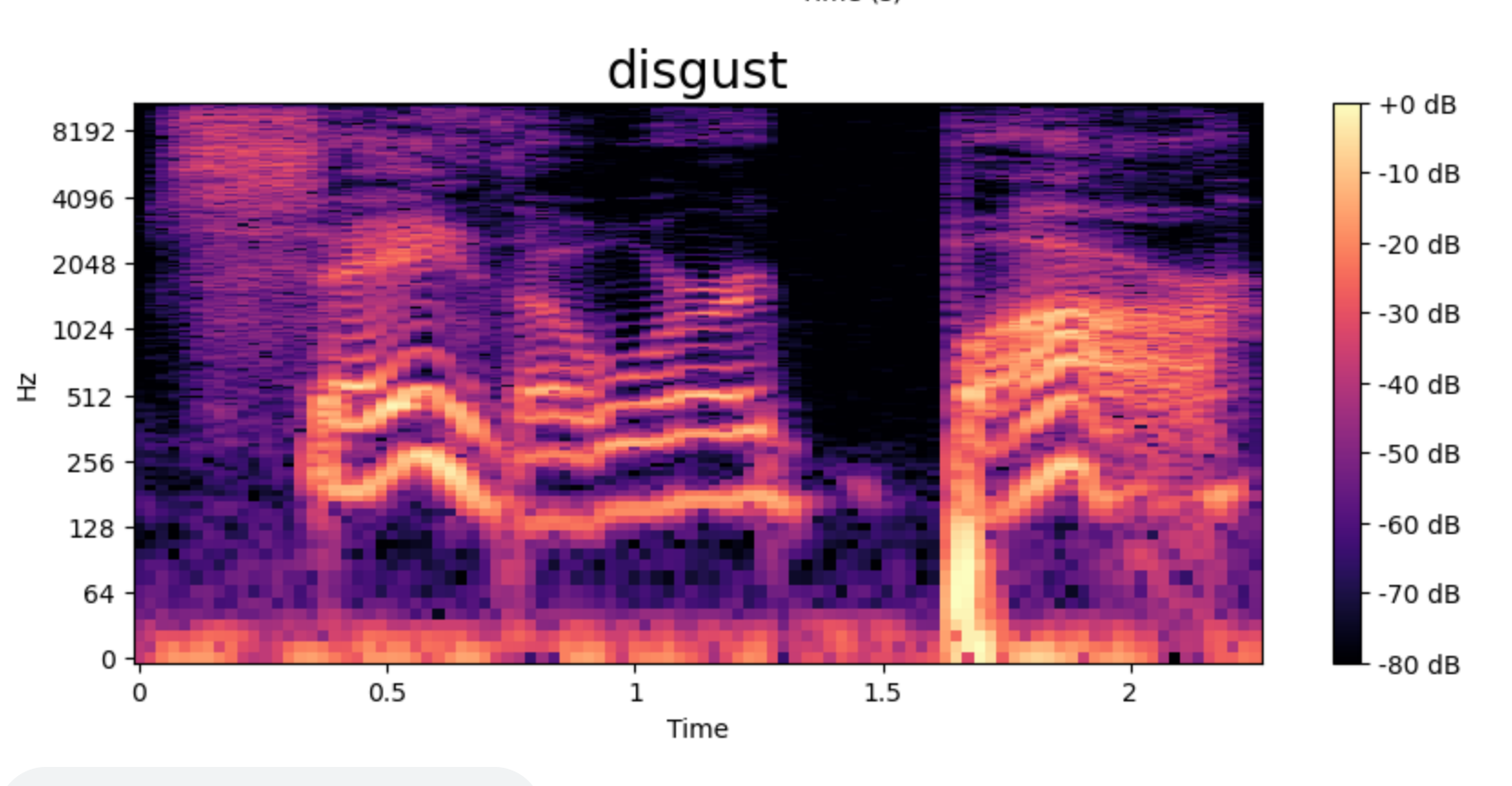




Out[11]: 0:00 / 0:01

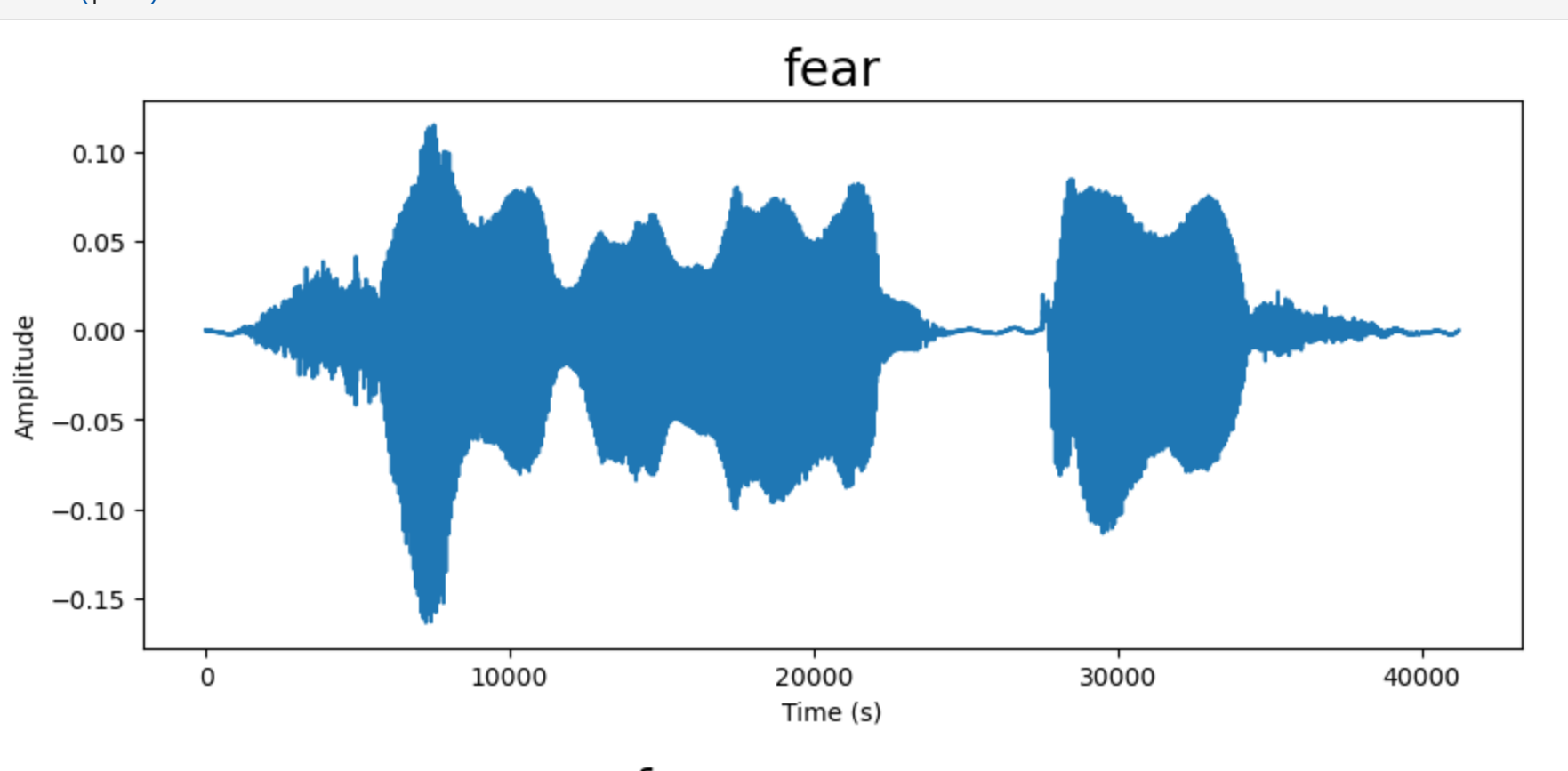
In [12]:
```python
emotion = 'disgust'
path = df['speech'][df['label'] == emotion].iloc[1]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate , emotion)
Audio(path)
```

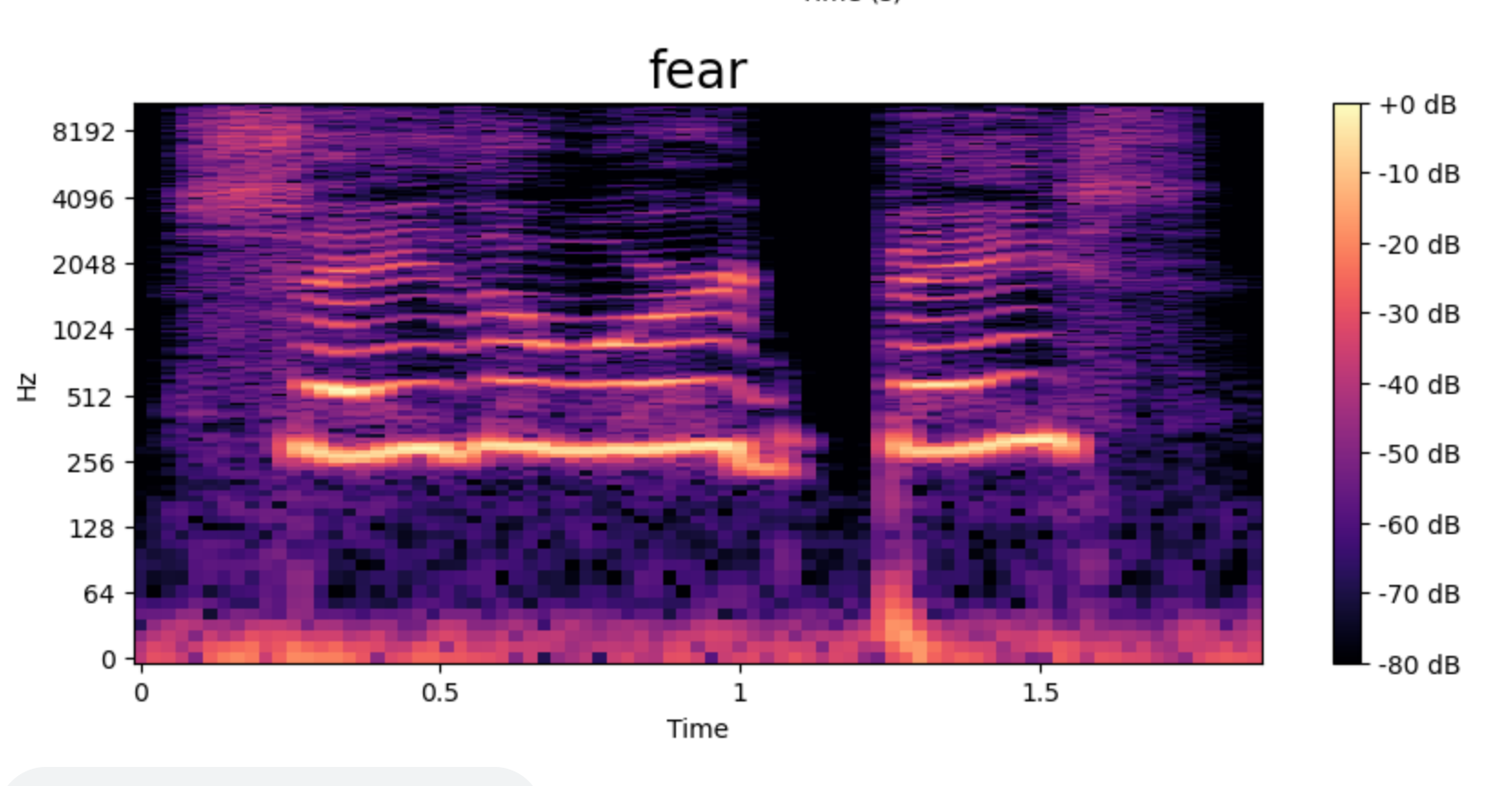




Out[12]: 0:00 / 0:02

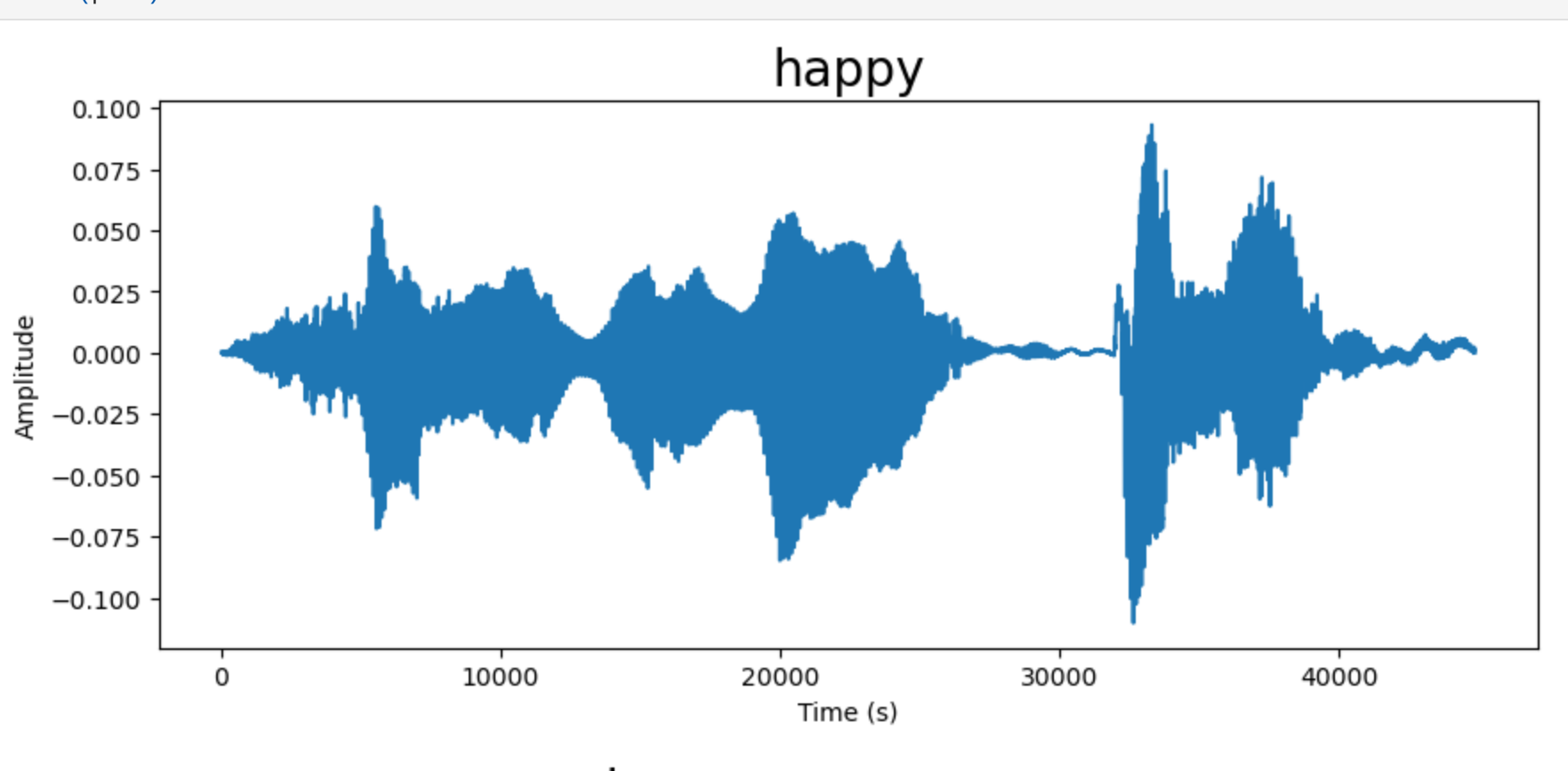
In [13]:
```python
emotion = 'fear'
path = df['speech'][df['label'] == emotion].iloc[2]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate , emotion)
Audio(path)
```

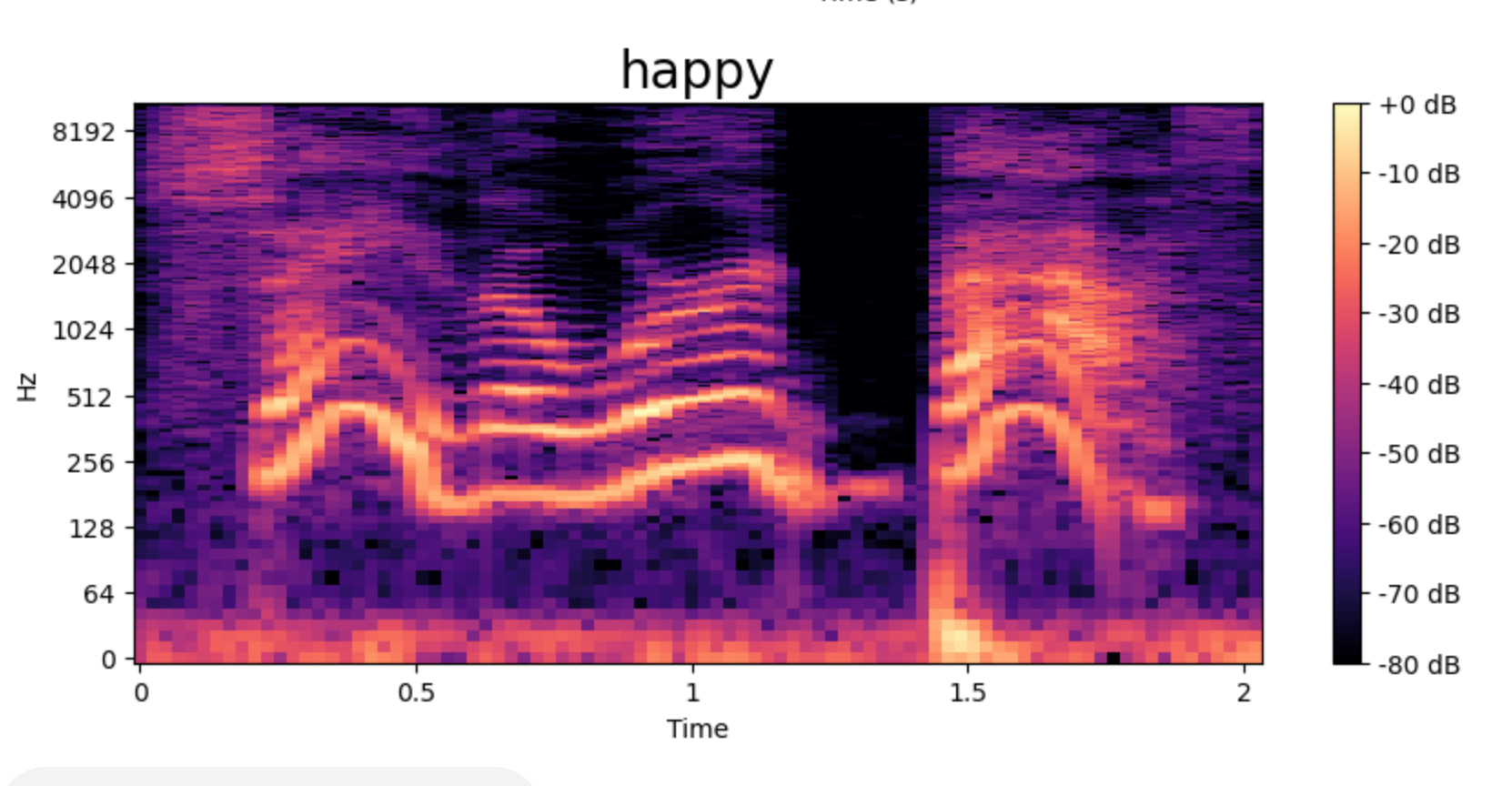




Out[13]: 0:00 / 0:01

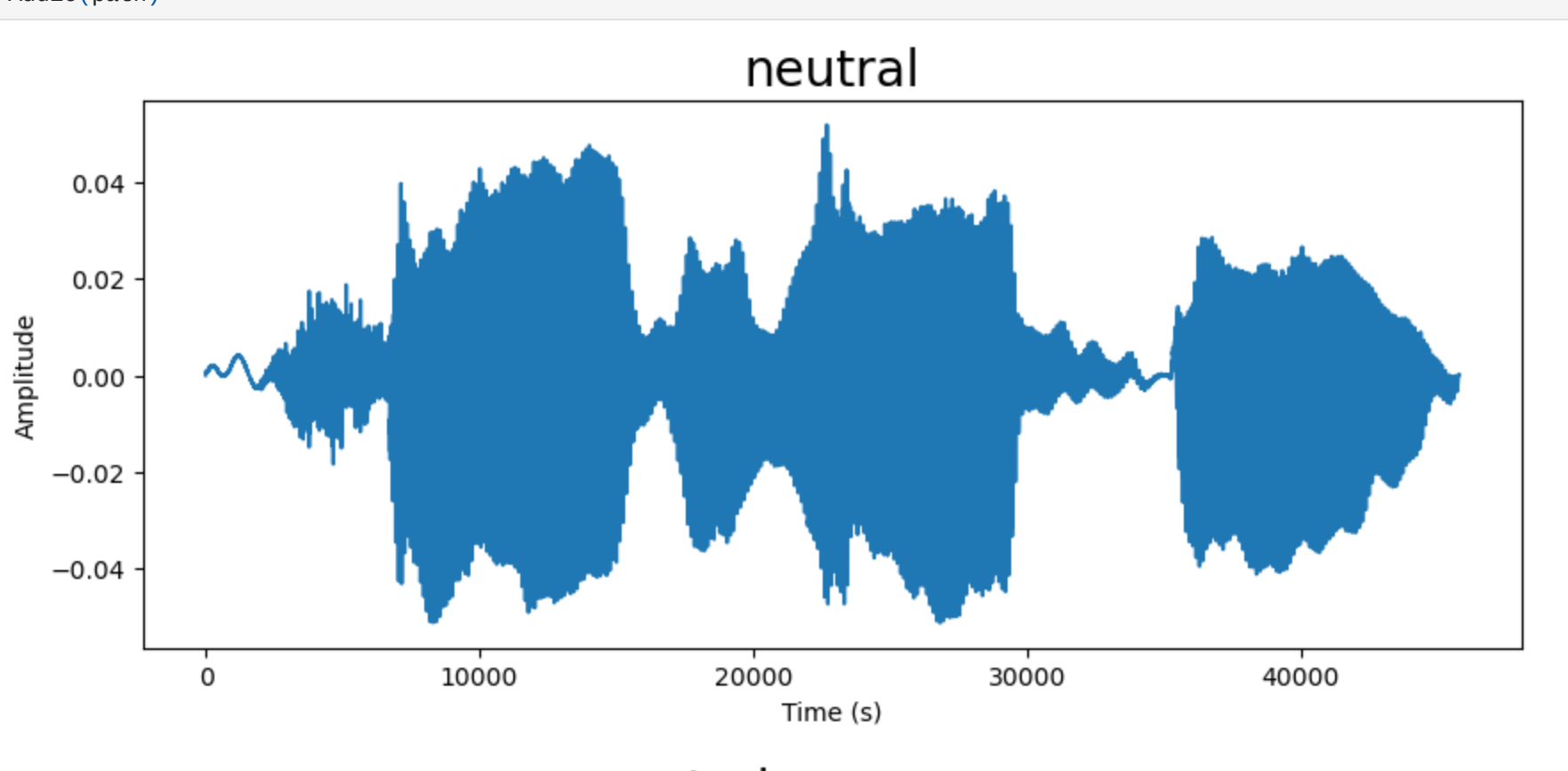
In [14]:
```python
emotion = 'happy'
path = df['speech'][df['label'] == emotion].iloc[3]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate , emotion)
Audio(path)
```

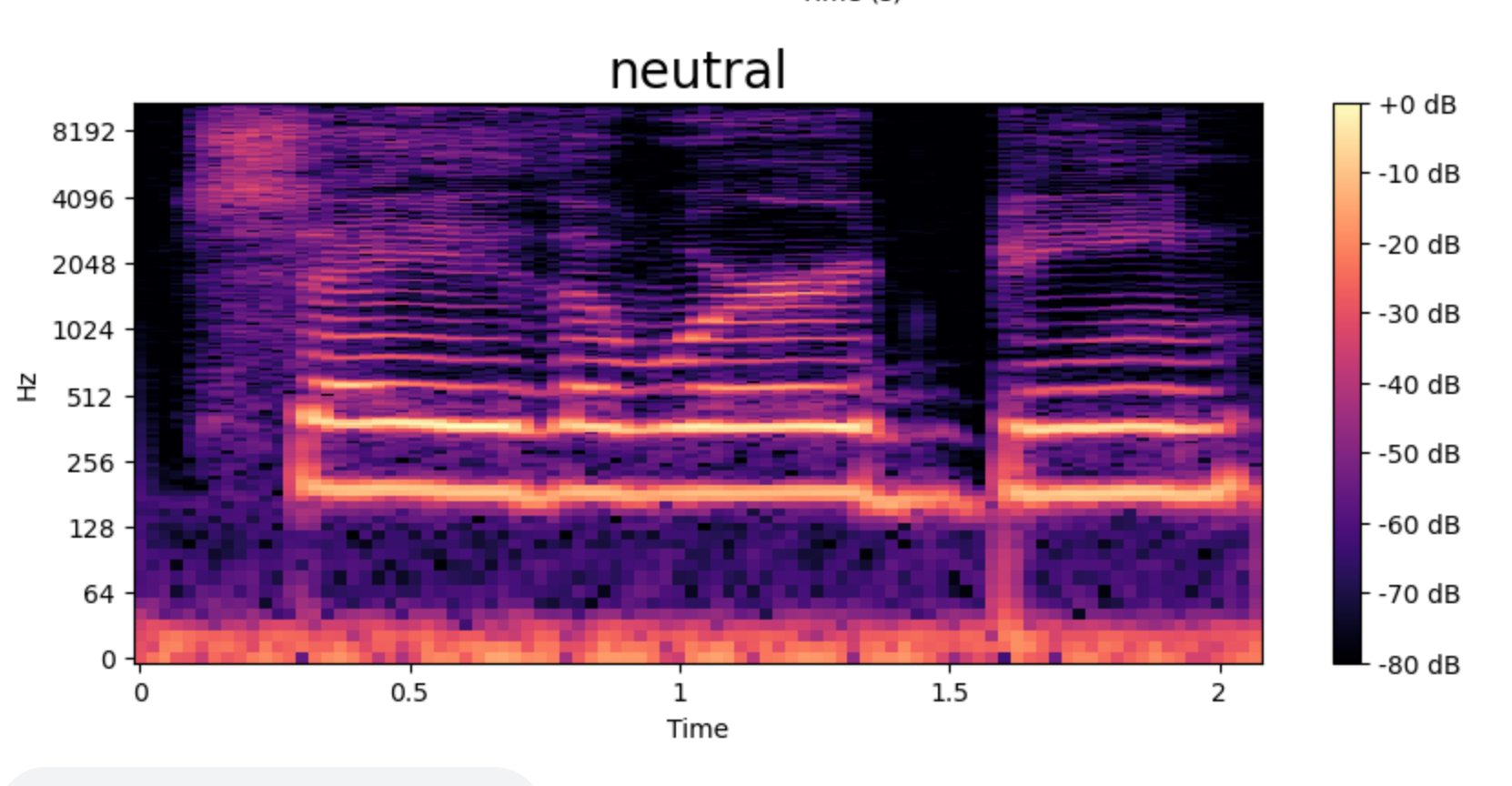




Out[14]: 0:00 / 0:02

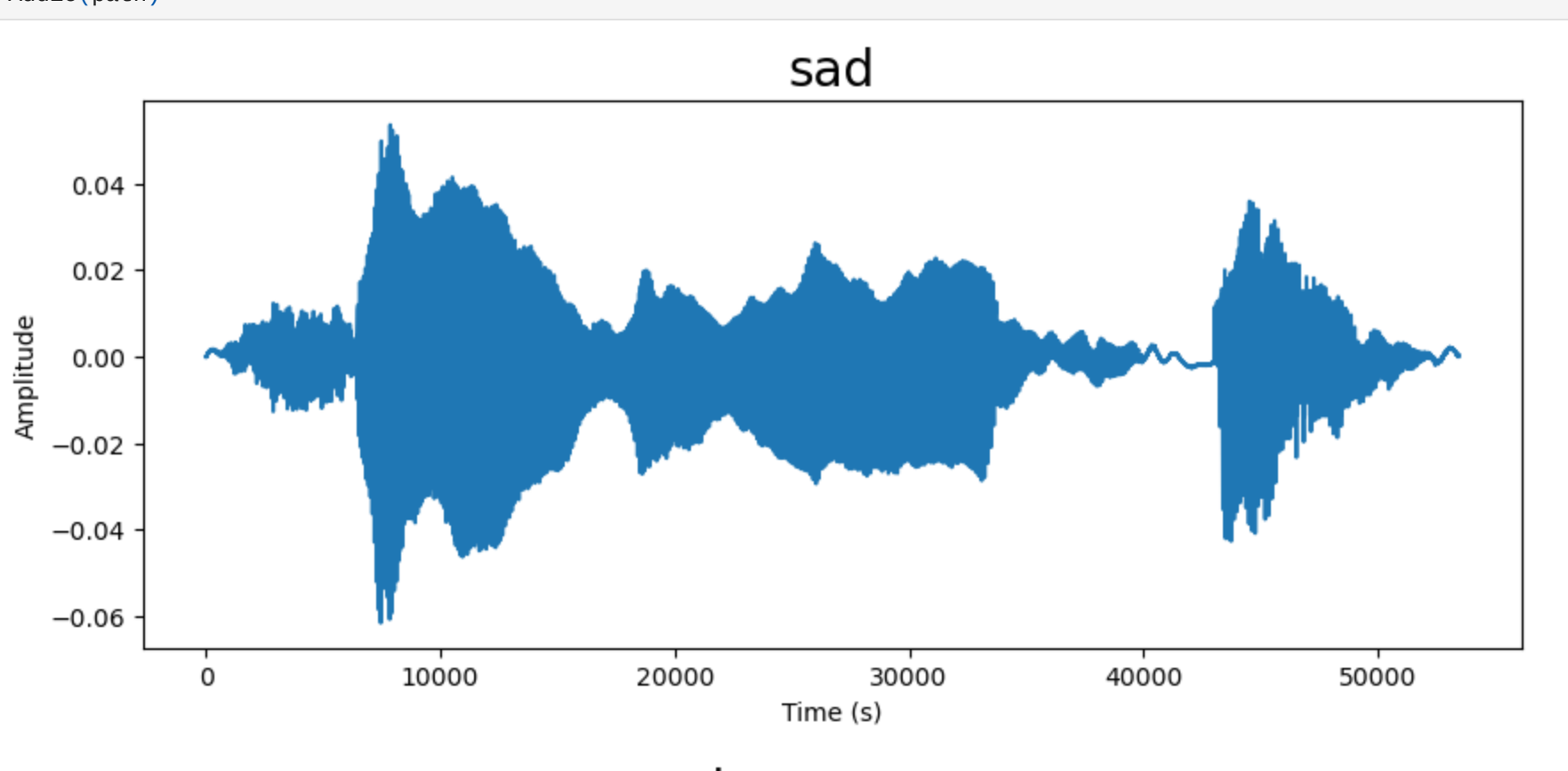
In [15]:
```python
emotion = 'neutral'
path = df['speech'][df['label'] == emotion].iloc[4]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate , emotion)
Audio(path)
```

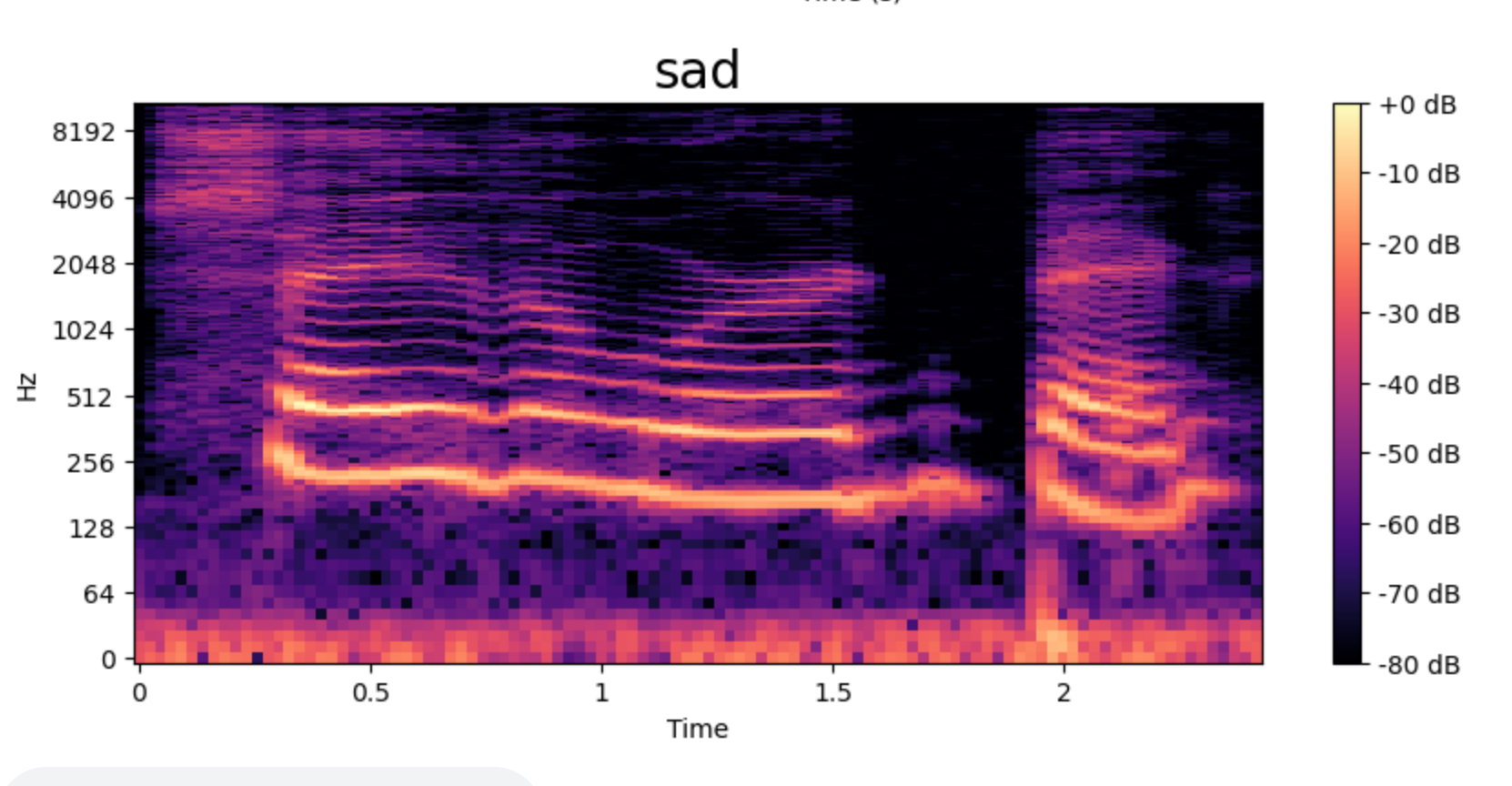




Out[15]: 0:00 / 0:02

In [16]:
```python
emotion = 'sad'
path = df['speech'][df['label'] == emotion].iloc[5]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate , emotion)
Audio(path)
```





Out[16]: 0:00 / 0:02

Feature Extraction

In [17]:
```python
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration=3, offset=0.5)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40).T, axis=0)
    return mfcc
```

In [18]:
```python
extract_mfcc(df['speech'][0])
```

Out[18]:
```
array([-3.96986206e+02,  7.74405365e+01, -1.95927906e+01, -2.16666889e+01,
       -2.11275506e+00,  1.00753632e+01, -2.03667068e+01, -6.09244919e+00,
       -7.21228361e+00, -5.57365894e-01, -1.83255327e+00,  2.02101514e-01,
        7.27551281e-01,  1.31773770e+00,  2.88633752e+00,  2.85579133e+00,
       -4.71292162e+00, -4.43651056e+00, -1.62115920e+00, -1.02398405e+01,
       -7.55126143e+00, -1.79688025e+00, -7.03765202e+00,  9.43658447e+00,
        8.35585499e+00,  2.17123604e+01,  1.92169895e+01,  2.03489265e+01,
        1.34133663e+01,  8.33917141e+00,  3.94722402e-01,  5.11131430e+00,
        9.56873894e+00,  5.45486832e+00,  2.50996375e+00, -1.82390714e+00,
        4.86896276e+00,  9.31392384e+00,  2.08915019e+00, -1.90649128e+00],
      dtype=float32)
```

In [19]:
```python
X_mfcc = df['speech'].apply(lambda x: extract_mfcc(x))
```

In [20]:
```python
X_mfcc
```

Out[20]:
```
0       [-396.9862, 77.44054, -19.59279, -21.666689, -...
1       [-465.73267, 98.77373, 0.65600896, -32.74544, ...
2       [-429.79196, 46.124, 1.5550478, -0.21709539, 2...
3       [-403.46118, 76.32369, -12.531774, -22.288858,...
4       [-434.05756, 77.4455, 10.8655, 16.092943, 8.04...
                              ...
5595    [-406.48053, 80.379875, 32.462395, 46.579094, ...
5596    [-426.90918, 102.54757, 24.800041, 43.048096, ...
5597    [-378.50494, 80.91062, 35.300533, 39.74792, -4...
5598    [-434.8618, 89.906364, 28.37326, 39.57707, -2....
5599    [-421.8341, 70.69788, 32.476387, 45.642555, 4....
Name: speech, Length: 5600, dtype: object
```

In [21]:
```python
X = [x for x in X_mfcc]
X = np.array(X)
X.shape
```

Out[21]:
```
(5600, 40)
```

In [22]:
```python
X = np.expand_dims(X, -1)
X.shape
```

Out[22]:
```
(5600, 40, 1)
```

In [23]:
```python
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
y = enc.fit_transform(df[['label']])
```

In [24]:
```python
y = y.toarray()
```

In [25]:
```python
y.shape
```

Out[25]:
```
(5600, 7)
```

Create the LSTM Model

In [26]:
```python
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

model = Sequential([
    LSTM(256, return_sequences=False, input_shape=(40,1)),
    Dropout(0.2),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(7, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 256)               264192

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_2 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 7)                 455

=================================================================
Total params: 305799 (1.17 MB)
Trainable params: 305799 (1.17 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

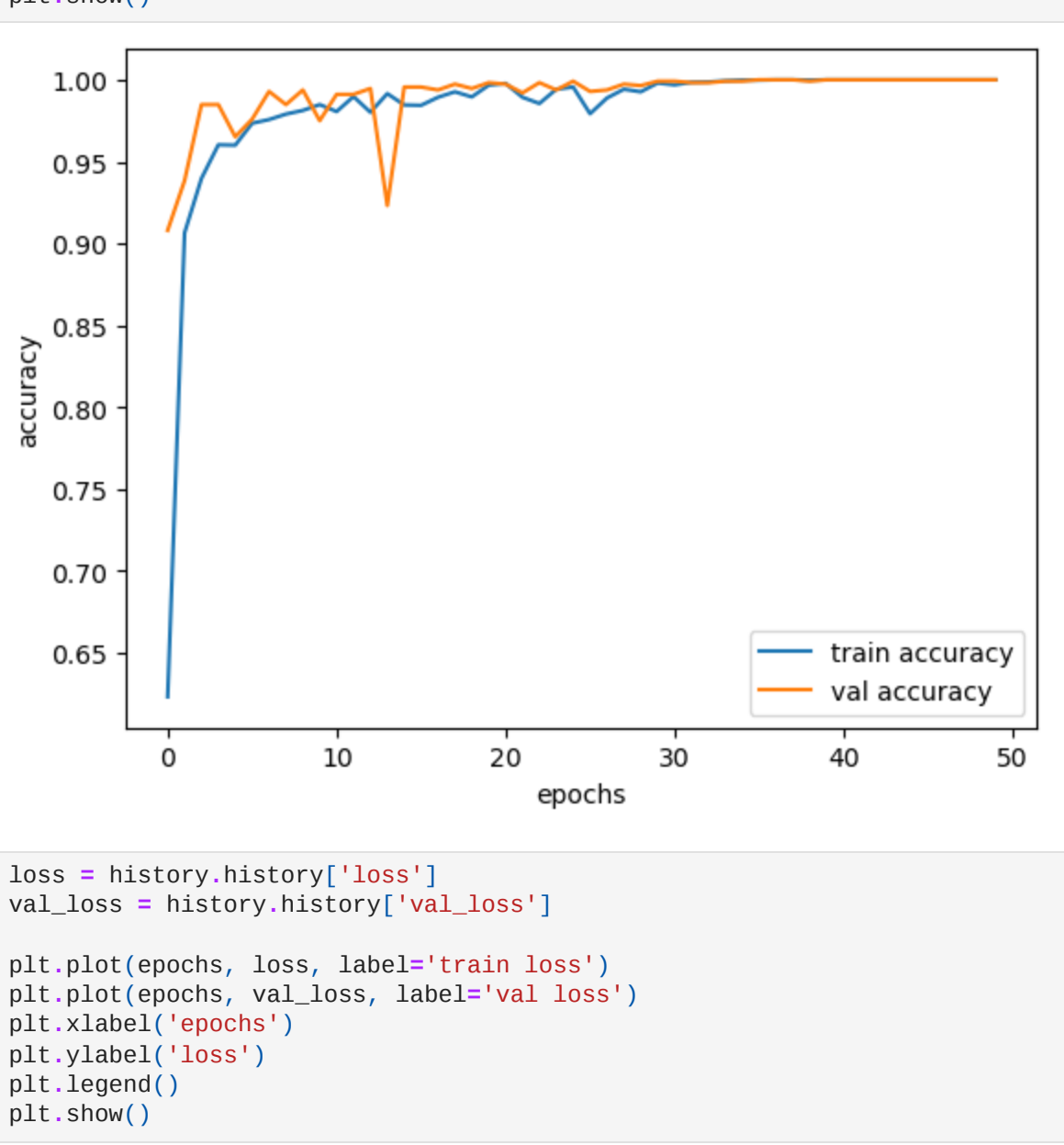
In [27]:
```python
history = model.fit(X, y, validation_split=0.2, epochs=50, batch_size=64)
```

```
Epoch 1/50
70/70 [==============================] - 27s 311ms/step - loss: 0.9956 - accuracy: 0.6232 - val_loss: 0.2969 - val_accuracy: 0.9080
Epoch 2/50
70/70 [==============================] - 21s 298ms/step - loss: 0.2954 - accuracy: 0.9065 - val_loss: 0.1678 - val_accuracy: 0.9384
Epoch 3/50
70/70 [==============================] - 20s 289ms/step - loss: 0.1908 - accuracy: 0.9397 - val_loss: 0.0486 - val_accuracy: 0.9848
Epoch 4/50
70/70 [==============================] - 21s 299ms/step - loss: 0.1293 - accuracy: 0.9603 - val_loss: 0.0500 - val_accuracy: 0.9848
Epoch 5/50
70/70 [==============================] - 21s 299ms/step - loss: 0.1314 - accuracy: 0.9600 - val_loss: 0.0979 - val_accuracy: 0.9652
Epoch 6/50
70/70 [==============================] - 19s 268ms/step - loss: 0.0896 - accuracy: 0.9734 - val_loss: 0.0755 - val_accuracy: 0.9759
Epoch 7/50
70/70 [==============================] - 19s 270ms/step - loss: 0.0906 - accuracy: 0.9757 - val_loss: 0.0235 - val_accuracy: 0.9929
Epoch 8/50
70/70 [==============================] - 19s 268ms/step - loss: 0.0706 - accuracy: 0.9790 - val_loss: 0.0456 - val_accuracy: 0.9848
Epoch 9/50
70/70 [==============================] - 19s 269ms/step - loss: 0.0636 - accuracy: 0.9812 - val_loss: 0.0204 - val_accuracy: 0.9937
Epoch 10/50
70/70 [==============================] - 19s 276ms/step - loss: 0.0544 - accuracy: 0.9848 - val_loss: 0.0799 - val_accuracy: 0.9750
Epoch 11/50
70/70 [==============================] - 20s 287ms/step - loss: 0.0632 - accuracy: 0.9806 - val_loss: 0.0270 - val_accuracy: 0.9911
Epoch 12/50
70/70 [==============================] - 19s 268ms/step - loss: 0.0348 - accuracy: 0.9897 - val_loss: 0.0260 - val_accuracy: 0.9911
Epoch 13/50
70/70 [==============================] - 19s 272ms/step - loss: 0.0642 - accuracy: 0.9801 - val_loss: 0.0179 - val_accuracy: 0.9946
Epoch 14/50
70/70 [==============================] - 21s 304ms/step - loss: 0.0287 - accuracy: 0.9915 - val_loss: 0.4239 - val_accuracy: 0.9232
Epoch 15/50
70/70 [==============================] - 20s 279ms/step - loss: 0.0470 - accuracy: 0.9846 - val_loss: 0.0147 - val_accuracy: 0.9955
Epoch 16/50
70/70 [==============================] - 20s 282ms/step - loss: 0.0494 - accuracy: 0.9844 - val_loss: 0.0127 - val_accuracy: 0.9955
Epoch 17/50
70/70 [==============================] - 19s 271ms/step - loss: 0.0331 - accuracy: 0.9893 - val_loss: 0.0178 - val_accuracy: 0.9937
Epoch 18/50
70/70 [==============================] - 20s 287ms/step - loss: 0.0237 - accuracy: 0.9926 - val_loss: 0.0070 - val_accuracy: 0.9973
Epoch 19/50
70/70 [==============================] - 20s 290ms/step - loss: 0.0341 - accuracy: 0.9895 - val_loss: 0.0180 - val_accuracy: 0.9946
Epoch 20/50
70/70 [==============================] - 21s 301ms/step - loss: 0.0098 - accuracy: 0.9967 - val_loss: 0.0039 - val_accuracy: 0.9982
Epoch 21/50
70/70 [==============================] - 20s 292ms/step - loss: 0.0084 - accuracy: 0.9978 - val_loss: 0.0092 - val_accuracy: 0.9973
Epoch 22/50
70/70 [==============================] - 20s 280ms/step - loss: 0.0416 - accuracy: 0.9893 - val_loss: 0.0249 - val_accuracy: 0.9920
Epoch 23/50
70/70 [==============================] - 20s 291ms/step - loss: 0.0471 - accuracy: 0.9855 - val_loss: 0.0042 - val_accuracy: 0.9982
Epoch 24/50
70/70 [==============================] - 21s 299ms/step - loss: 0.0183 - accuracy: 0.9942 - val_loss: 0.0192 - val_accuracy: 0.9937
Epoch 25/50
70/70 [==============================] - 21s 294ms/step - loss: 0.0166 - accuracy: 0.9955 - val_loss: 0.0040 - val_accuracy: 0.9991
Epoch 26/50
70/70 [==============================] - 19s 277ms/step - loss: 0.0759 - accuracy: 0.9792 - val_loss: 0.0215 - val_accuracy: 0.9929
Epoch 27/50
70/70 [==============================] - 19s 276ms/step - loss: 0.0328 - accuracy: 0.9888 - val_loss: 0.0141 - val_accuracy: 0.9937
Epoch 28/50
70/70 [==============================] - 22s 313ms/step - loss: 0.0192 - accuracy: 0.9942 - val_loss: 0.0056 - val_accuracy: 0.9973
Epoch 29/50
70/70 [==============================] - 21s 305ms/step - loss: 0.0242 - accuracy: 0.9926 - val_loss: 0.0069 - val_accuracy: 0.9964
Epoch 30/50
70/70 [==============================] - 21s 294ms/step - loss: 0.0096 - accuracy: 0.9980 - val_loss: 0.0031 - val_accuracy: 0.9991
Epoch 31/50
70/70 [==============================] - 21s 296ms/step - loss: 0.0123 - accuracy: 0.9969 - val_loss: 0.0028 - val_accuracy: 0.9991
Epoch 32/50
70/70 [==============================] - 20s 292ms/step - loss: 0.0061 - accuracy: 0.9984 - val_loss: 0.0025 - val_accuracy: 0.9982
Epoch 33/50
70/70 [==============================] - 20s 289ms/step - loss: 0.0039 - accuracy: 0.9987 - val_loss: 0.0037 - val_accuracy: 0.9982
Epoch 34/50
70/70 [==============================] - 19s 274ms/step - loss: 0.0026 - accuracy: 0.9993 - val_loss: 0.0021 - val_accuracy: 0.9991
Epoch 35/50
70/70 [==============================] - 20s 288ms/step - loss: 9.0372e-04 - accuracy: 0.9998 - val_loss: 7.0922e-04 - val_accuracy: 0.9991
Epoch 36/50
70/70 [==============================] - 20s 287ms/step - loss: 7.6067e-04 - accuracy: 0.9998 - val_loss: 2.6690e-04 - val_accuracy: 1.0000
Epoch 37/50
70/70 [==============================] - 19s 275ms/step - loss: 4.3686e-04 - accuracy: 1.0000 - val_loss: 2.2761e-04 - val_accuracy: 1.0000
Epoch 38/50
70/70 [==============================] - 18s 260ms/step - loss: 4.7087e-04 - accuracy: 1.0000 - val_loss: 2.6566e-04 - val_accuracy: 1.0000
Epoch 39/50
70/70 [==============================] - 18s 259ms/step - loss: 3.3512e-04 - accuracy: 0.9998 - val_loss: 0.0018 - val_accuracy: 0.9991
Epoch 40/50
70/70 [==============================] - 18s 262ms/step - loss: 6.0216e-04 - accuracy: 1.0000 - val_loss: 7.4167e-05 - val_accuracy: 1.0000
Epoch 41/50
70/70 [==============================] - 18s 260ms/step - loss: 2.5231e-04 - accuracy: 1.0000 - val_loss: 3.2960e-05 - val_accuracy: 1.0000
Epoch 42/50
70/70 [==============================] - 18s 252ms/step - loss: 1.6576e-04 - accuracy: 1.0000 - val_loss: 3.8490e-05 - val_accuracy: 1.0000
Epoch 43/50
70/70 [==============================] - 18s 263ms/step - loss: 1.5266e-04 - accuracy: 1.0000 - val_loss: 4.3742e-05 - val_accuracy: 1.0000
Epoch 44/50
70/70 [==============================] - 18s 261ms/step - loss: 1.5447e-04 - accuracy: 1.0000 - val_loss: 3.3560e-05 - val_accuracy: 1.0000
Epoch 45/50
70/70 [==============================] - 18s 262ms/step - loss: 1.4033e-04 - accuracy: 1.0000 - val_loss: 2.5339e-05 - val_accuracy: 1.0000
Epoch 46/50
70/70 [==============================] - 17s 237ms/step - loss: 7.5803e-05 - accuracy: 1.0000 - val_loss: 2.2854e-05 - val_accuracy: 1.0000
Epoch 47/50
70/70 [==============================] - 19s 275ms/step - loss: 1.2502e-04 - accuracy: 1.0000 - val_loss: 1.2173e-05 - val_accuracy: 1.0000
Epoch 48/50
70/70 [==============================] - 21s 296ms/step - loss: 2.0107e-04 - accuracy: 1.0000 - val_loss: 6.2120e-06 - val_accuracy: 1.0000
Epoch 49/50
70/70 [==============================] - 21s 294ms/step - loss: 6.4513e-05 - accuracy: 1.0000 - val_loss: 3.7389e-06 - val_accuracy: 1.0000
Epoch 50/50
70/70 [==============================] - 18s 262ms/step - loss: 4.7628e-05 - accuracy: 1.0000 - val_loss: 4.3593e-06 - val_accuracy: 1.0000
```

Plot the Result

In [28]:
```python
epochs = list(range(50))
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.plot(epochs, acc, label='train accuracy')
plt.plot(epochs, val_acc, label='val accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```



In [29]:
```python
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.plot(epochs, loss, label='train loss')
plt.plot(epochs, val_loss, label='val loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```