```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

#import os
#for dirname, _, filenames in os.walk('/kaggle/input'):
    #for filename in filenames:
        #print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

Import libraries

```
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
import librosa
import librosa.display
from IPython.display import Audio
import warnings
warnings.filterwarnings('ignore')
```

Load dataset

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

```
Dataset is loaded
```

```
paths[:5]
```

```
['/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto
emotional speech set data/YAF_fear/YAF_home_fear.wav',
 '/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto
emotional speech set data/YAF_fear/YAF_youth_fear.wav',
 '/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto
emotional speech set data/YAF_fear/YAF_near_fear.wav',
 '/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto
emotional speech set data/YAF_fear/YAF_search_fear.wav',
 '/kaggle/input/toronto-emotional-speech-set-tess/TESS Toronto
emotional speech set data/YAF_fear/YAF_pick_fear.wav']
```

```
labels[:5]
```

```
['fear', 'fear', 'fear', 'fear', 'fear']
```

Create a Data frame

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

```
                                              speech label
0  /kaggle/input/toronto-emotional-speech-set-tes...  fear
1  /kaggle/input/toronto-emotional-speech-set-tes...  fear
2  /kaggle/input/toronto-emotional-speech-set-tes...  fear
3  /kaggle/input/toronto-emotional-speech-set-tes...  fear
4  /kaggle/input/toronto-emotional-speech-set-tes...  fear
```

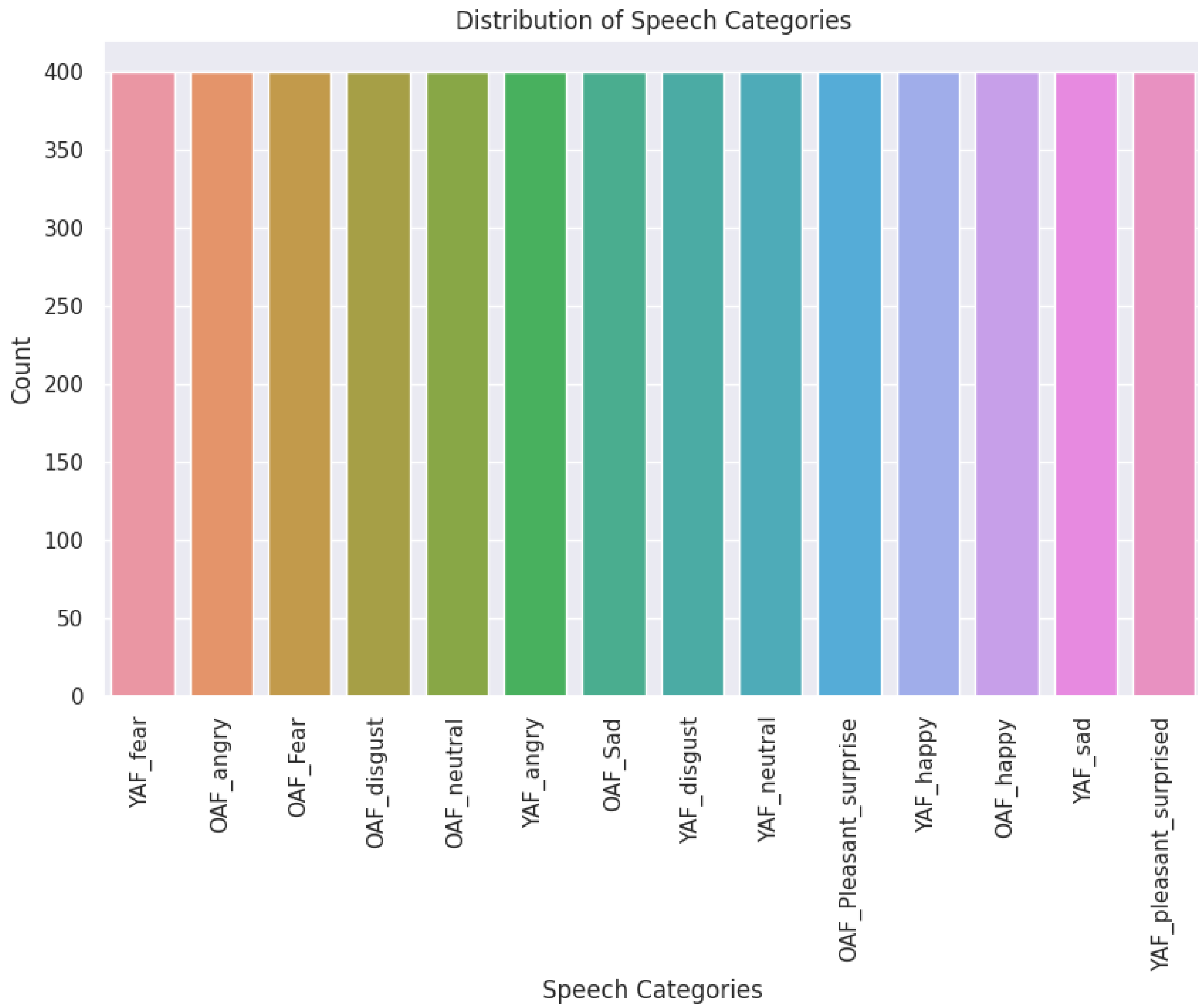
```
df['label'].value_counts()
```

```
label
fear       800
angry      800
disgust    800
neutral    800
sad        800
ps         800
happy      800
Name: count, dtype: int64
```

Exploratory data anlysis

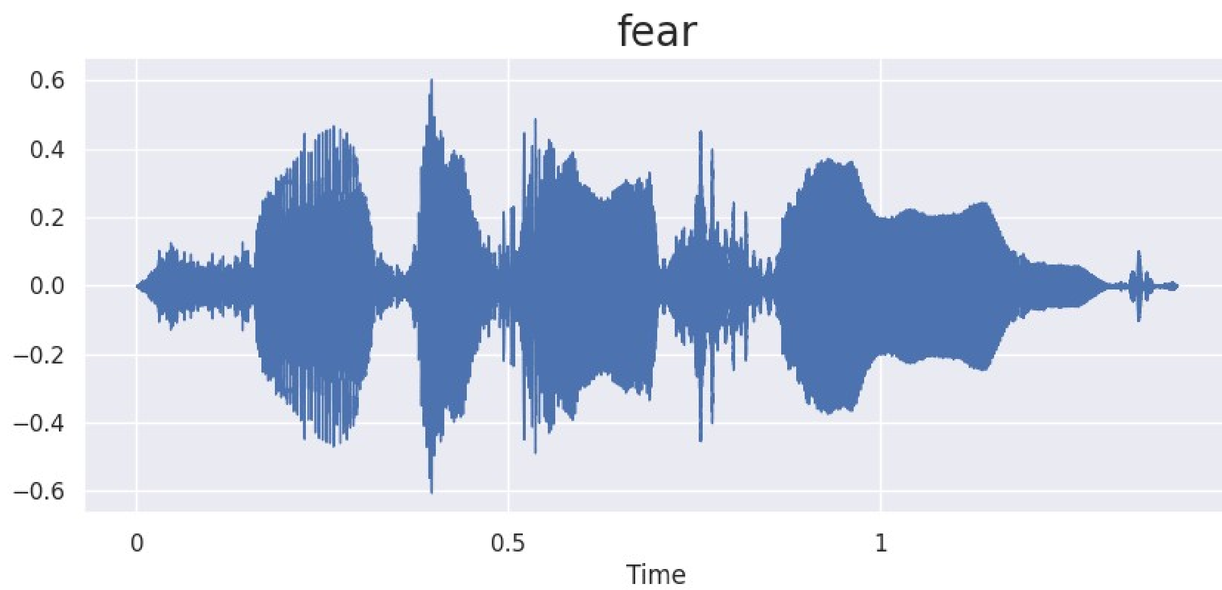
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

```python
# Assuming you have a DataFrame named 'df' and a column named 'speech'
# Extract the speech category from the file paths
df['speech_category'] = df['speech'].str.split('/').str[-2]

# Create a countplot for the extracted speech categories
sns.set(style="darkgrid")
plt.figure(figsize=(10, 6))  # Optional: Set the figure size

sns.countplot(data=df, x='speech_category')
plt.xticks(rotation=90)  # Optional: Rotate x-axis labels for better
visibility

# Label your plot
plt.xlabel('Speech Categories')
plt.ylabel('Count')
plt.title('Distribution of Speech Categories')

plt.show()
```

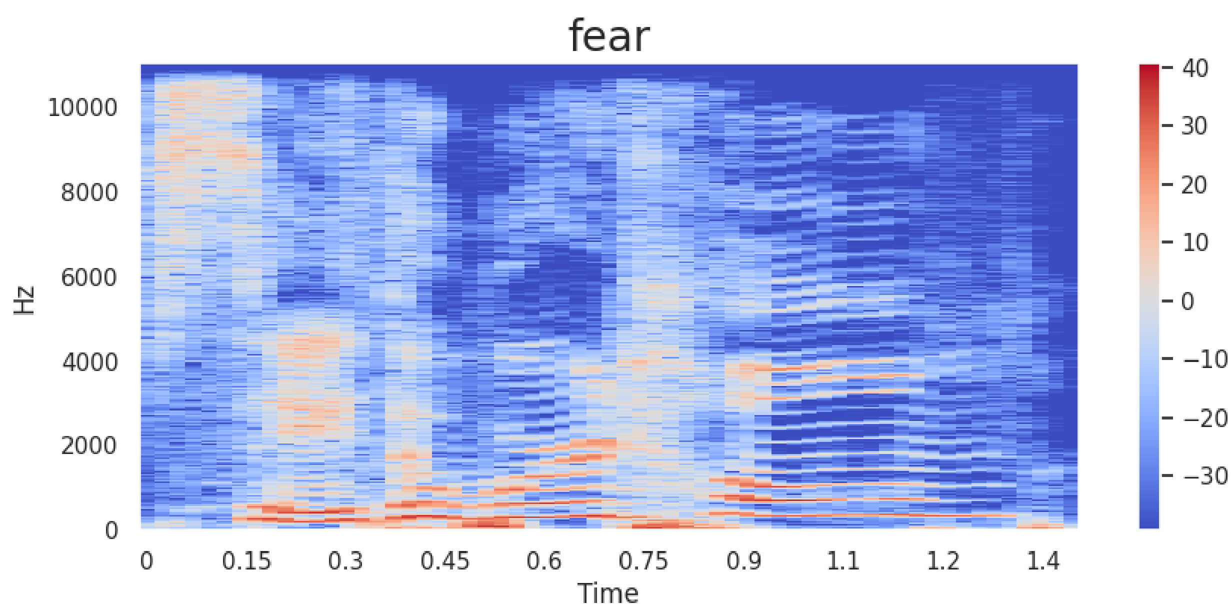



```
def waveplot(data, sr, emotion):
    plt.figure(figsize=(10,4))
    plt.title(emotion, size=20)
    librosa.display.waveshow(data, sr=sr)
    plt.show()

def spectogram(data, sr, emotion):
    x= librosa.stft(data)
    xdb= librosa.amplitude_to_db(abs(x))
    plt.figure(figsize=(10,4))
    plt.title(emotion, size=20)
    librosa.display.specshow(xdb, sr=sr, x_axis='time', y_axis='hz')
    plt.colorbar()

emotion = 'fear'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate =librosa.load(path)
waveplot(data, sampling_rate, emotion)
```

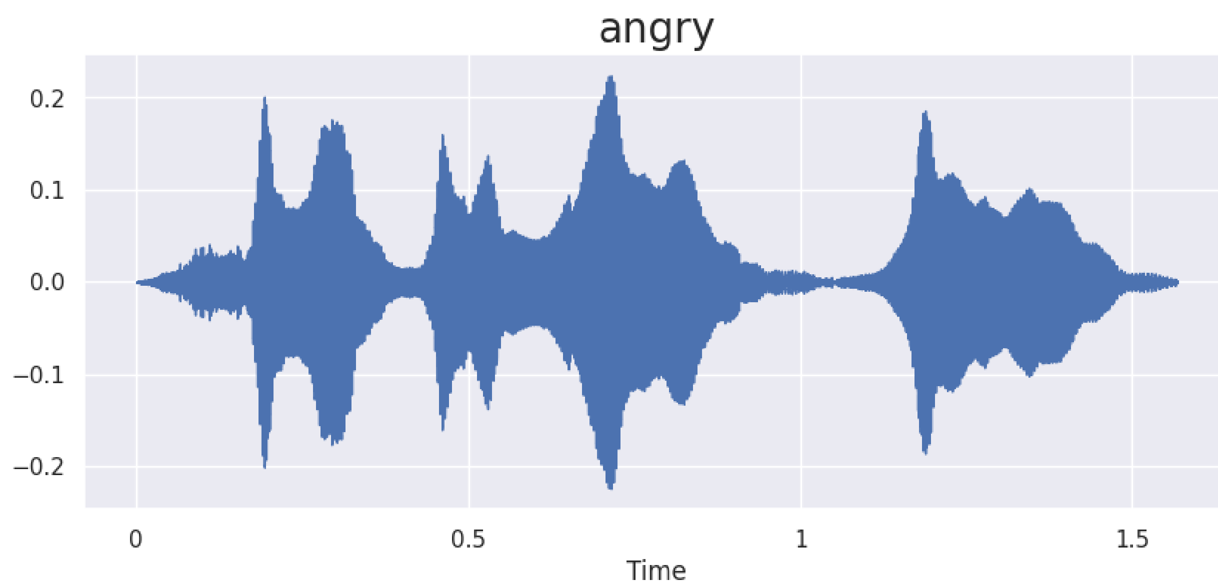
```
spectogram(data, sampling_rate, emotion)
Audio(path)
```

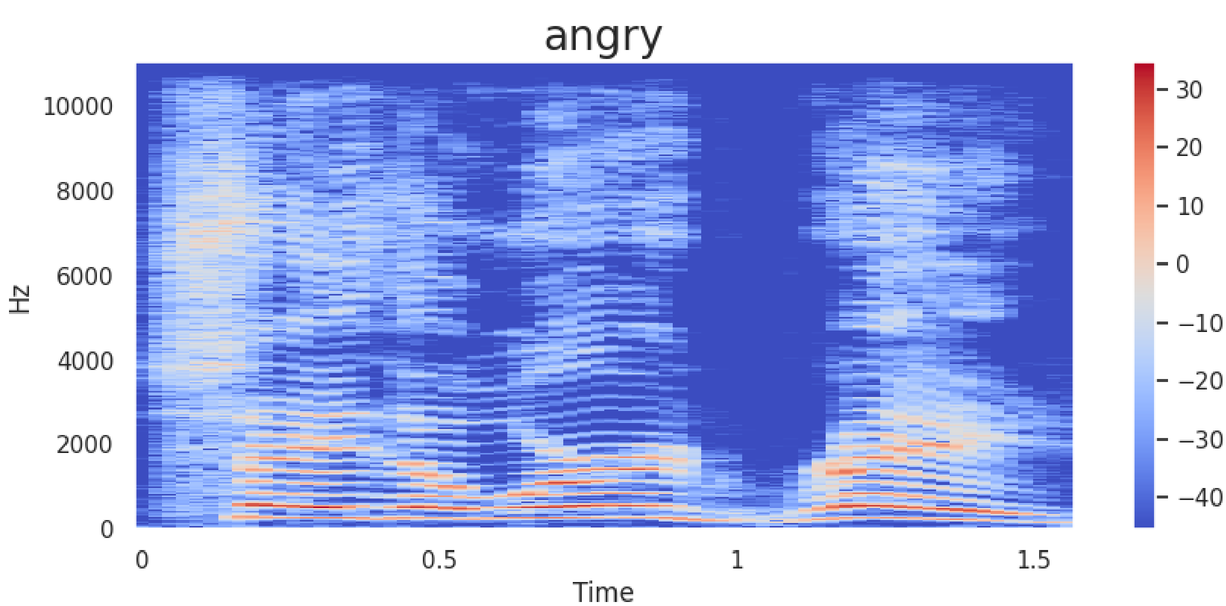



```
<IPython.lib.display.Audio object>
```

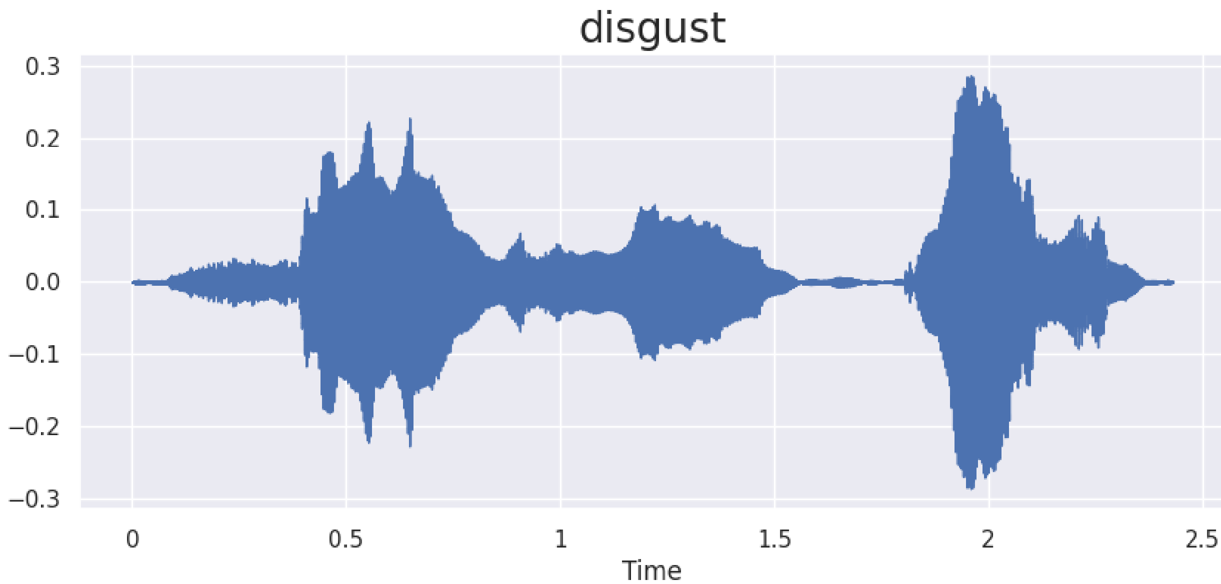


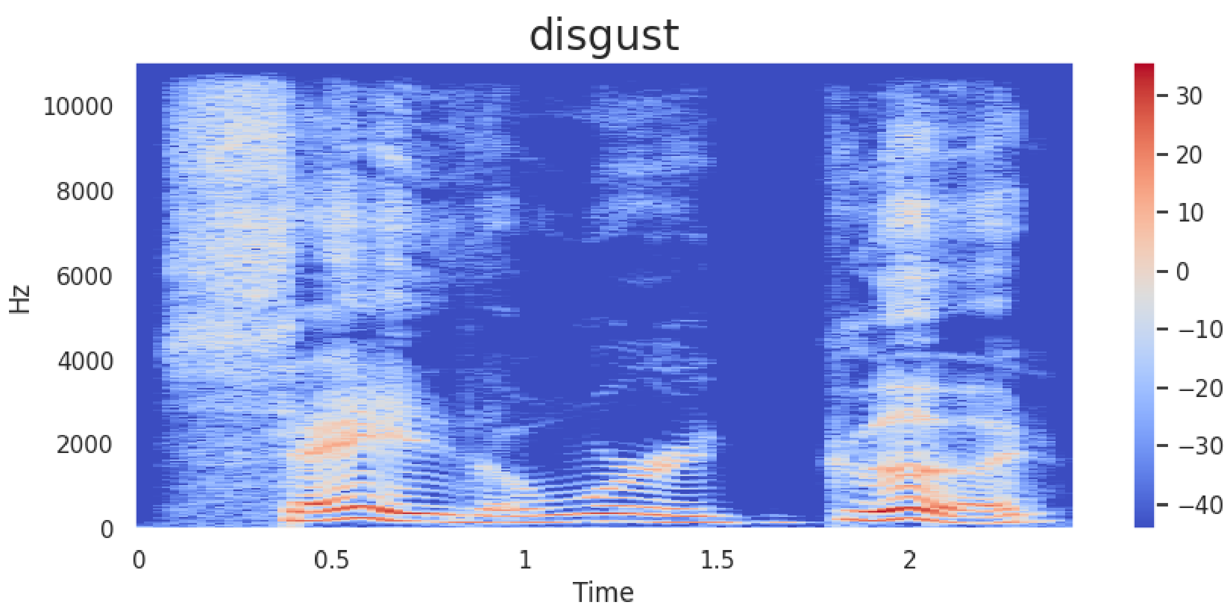

```
emotion = 'angry'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate =librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
```



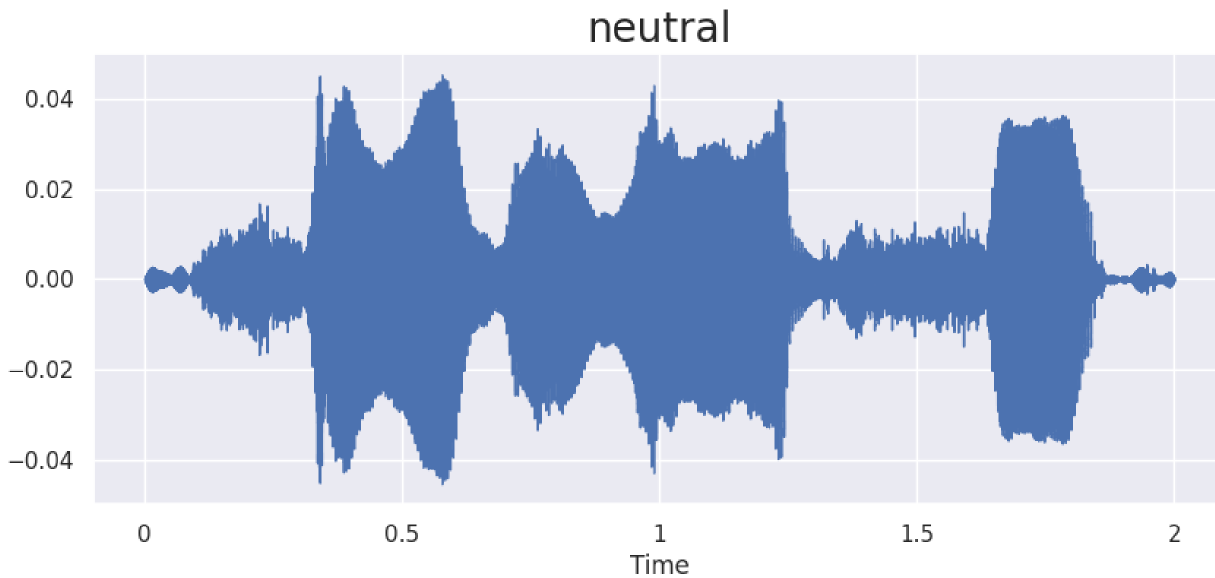


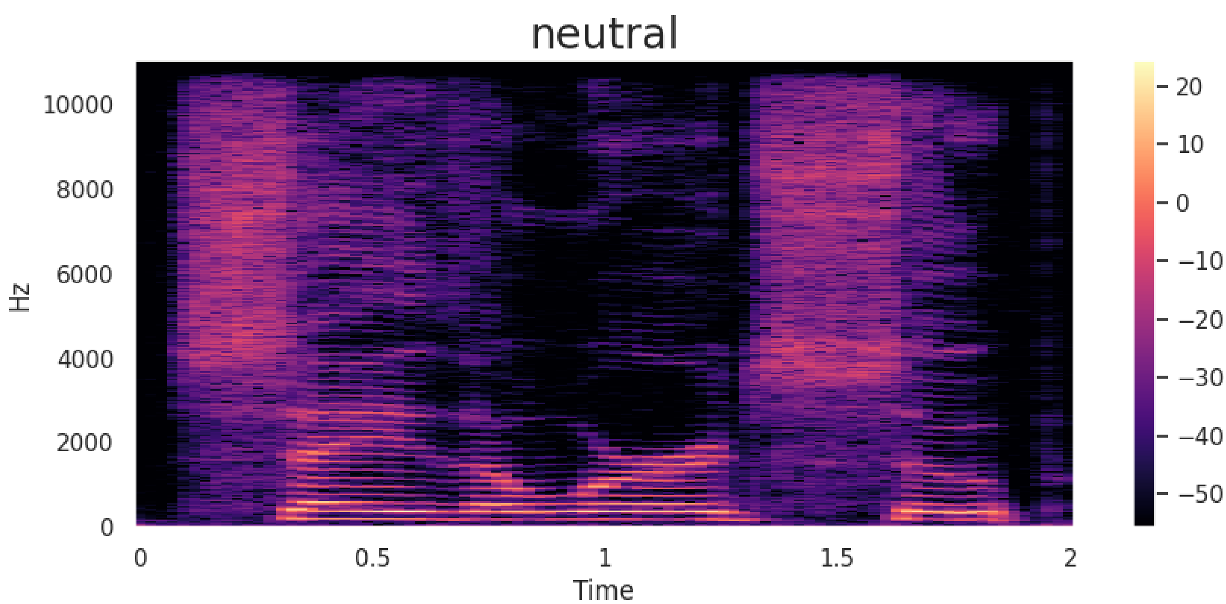

```
emotion = 'disgust'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate =librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
```



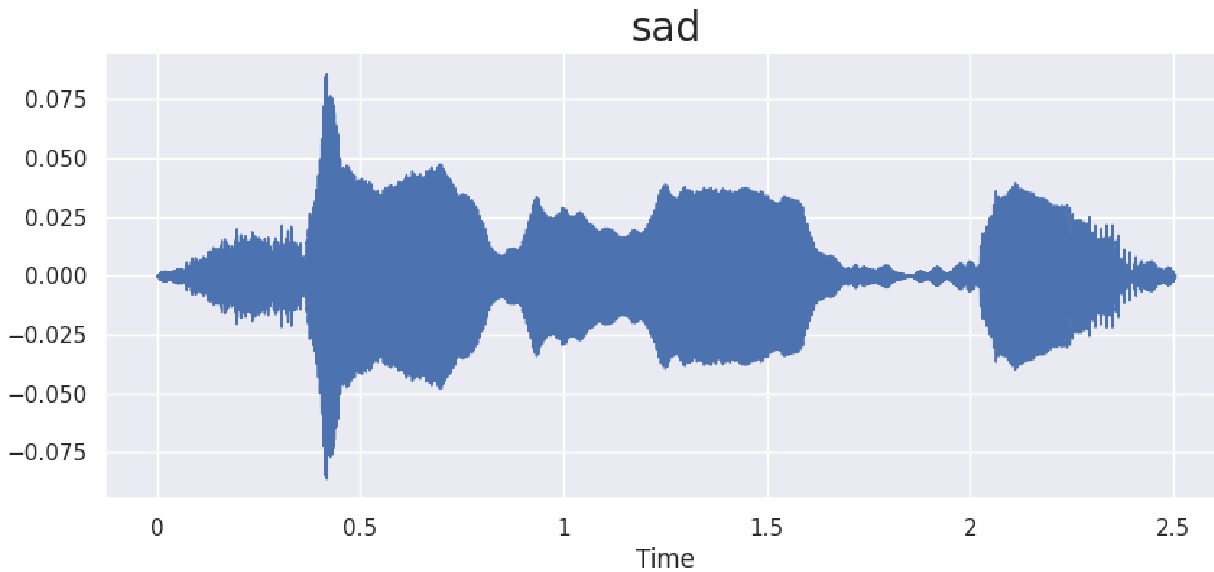


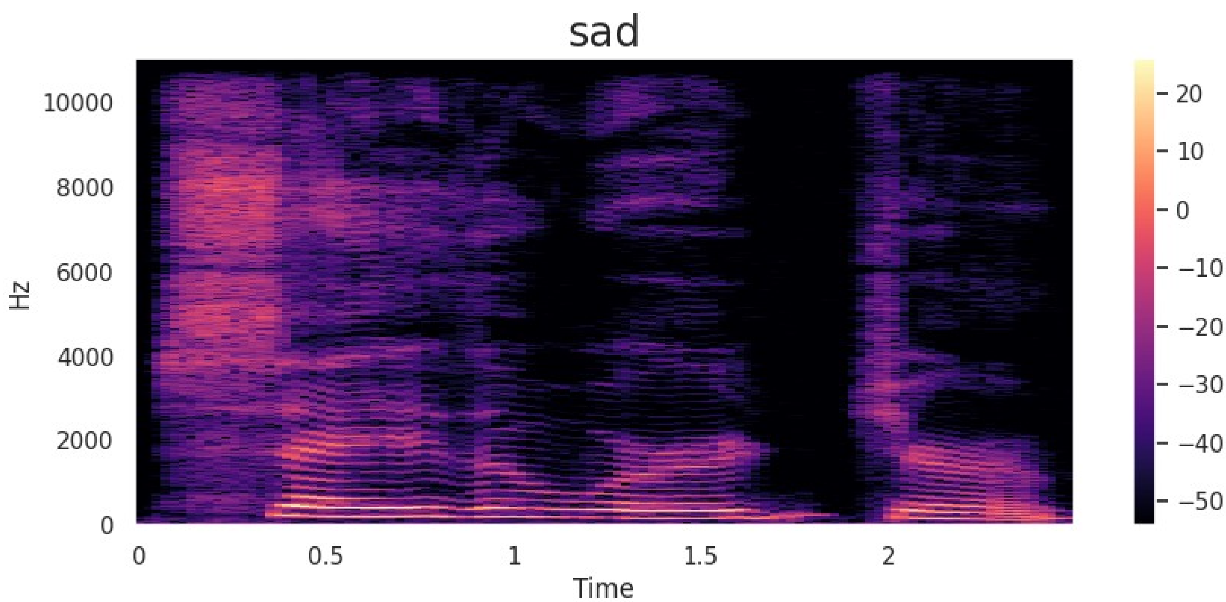

```
emotion = 'neutral'
path =np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate =librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
```



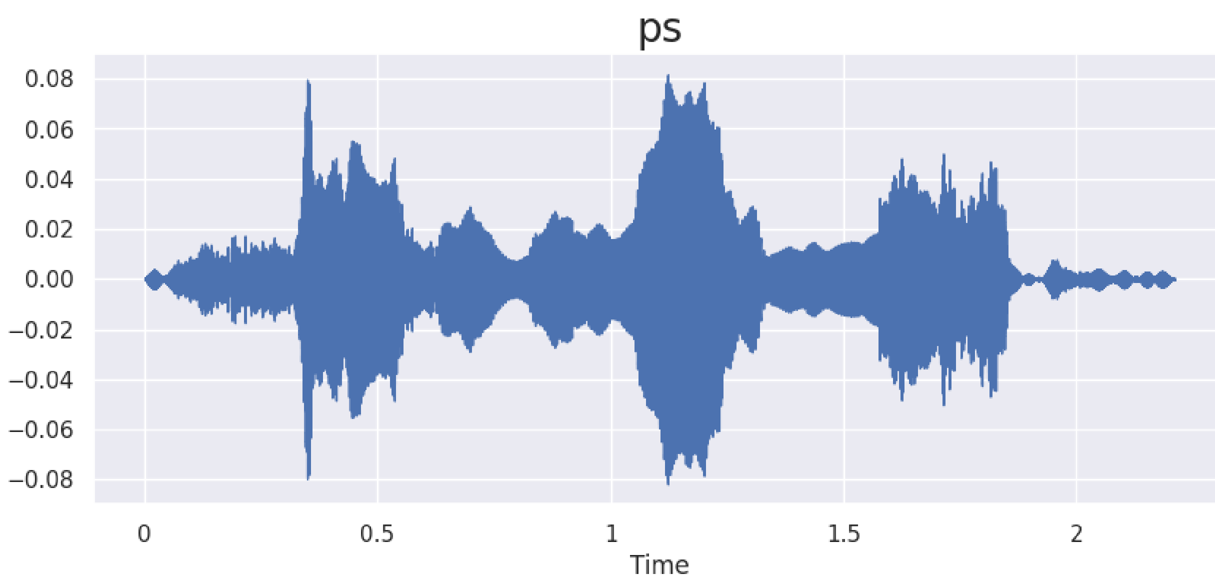


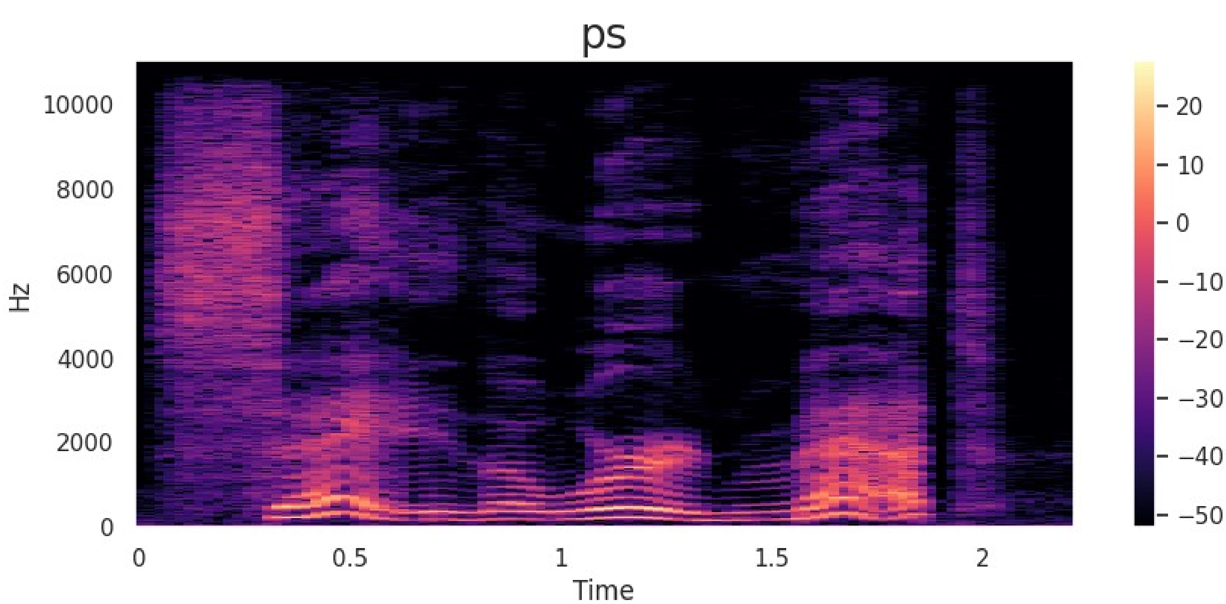

```
emotion = 'sad'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate =librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
```



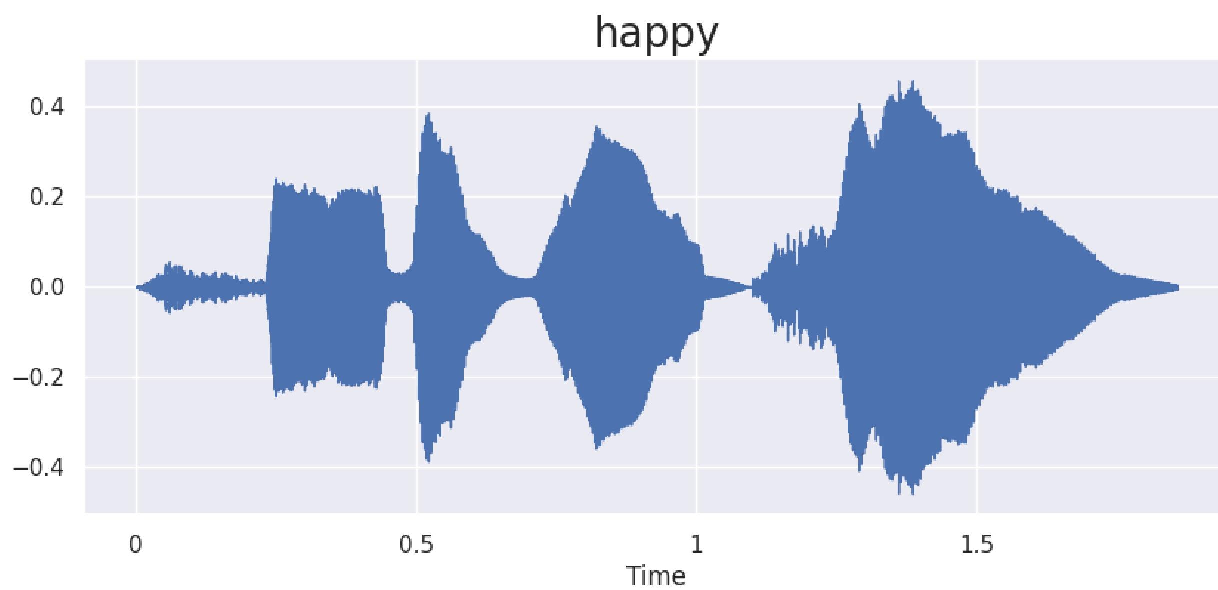


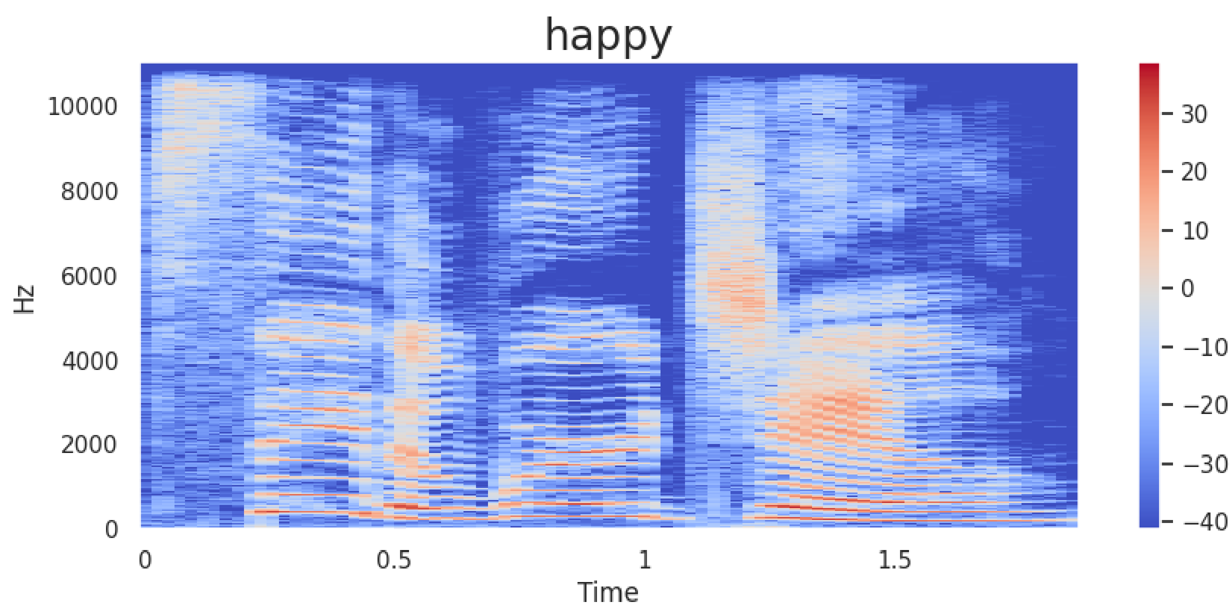

```
emotion = 'ps'
path =np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate =librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
```






```
emotion = 'happy'
path =np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate =librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
```





Feature Extraction

```
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

extract_mfcc(df['speech'][0])

array([-285.7373    ,   85.78295   ,   -2.1689105 ,   22.125532  ,
        -14.757396  ,   11.051346  ,   12.412452  ,   -3.0002632 ,
          1.0844971 ,   11.078272  ,  -17.419662  ,   -8.093215  ,
```

```
          6.5879736 ,   -4.220953  ,   -9.15508   ,    3.521479  ,
        -13.186381  ,   14.078853  ,   19.669733  ,   22.725618  ,
         32.574642  ,   16.325031  ,   -3.8427277 ,    0.89629626,
        -11.239262  ,    6.653462  ,   -2.5883691 ,   -7.7140174 ,
        -10.941658  ,   -2.4007556 ,   -5.2812862 ,    4.2711563 ,
        -11.202216  ,   -9.024621  ,   -3.6669848 ,    4.8697433 ,
         -1.6027985 ,    2.5600505 ,   11.454375  ,   11.23345   ],
      dtype=float32)
```

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

```
X_mfcc
```

```
0       [-285.7373, 85.78295, -2.1689105, 22.125532, -...
1       [-348.34332, 35.193233, -3.8413274, 14.658875,...
2       [-340.11435, 53.796444, -14.267782, 20.884031,...
3       [-306.6343, 21.25971, -4.4110823, 6.4871554, -...
4       [-344.7548, 46.329193, -24.171415, 19.392921, ...
                              ...
5595    [-374.39523, 60.865, 0.025058376, 8.431059, -2...
5596    [-313.9648, 39.847843, -5.6493053, -3.8675752,...
5597    [-357.54886, 77.88606, -15.224756, 2.1946328, ...
5598    [-353.14743, 101.68391, -14.175895, -12.037377...
5599    [-389.4595, 54.042767, 1.3469967, -1.4258995, ...
Name: speech, Length: 5600, dtype: object
```

```
X=[x for x in X_mfcc]
X = np.array(X)
X.shape
```

```
(5600, 40)
```

```
# input spilit
X = np.expand_dims(X, -1)
X.shape
```

```
(5600, 40, 1)
```

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

```
y = y.toarray()
```

```
y.shape
```

```
(5600, 7)
```

Create the LSTM MODEL

```
from keras.models import Sequential
from keras.layers import Dense,LSTM,Dropout

model = Sequential([
    LSTM(123, return_sequences=False, input_shape=(40,1)),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(7, activation='softmax')
])

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

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 123)               61500

 dense_3 (Dense)             (None, 64)                7936

 dropout_2 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 32)                2080

 dropout_3 (Dropout)         (None, 32)                0

 dense_5 (Dense)             (None, 7)                 231

=================================================================
Total params: 71,747
Trainable params: 71,747
Non-trainable params: 0
_________________________________________________________________
```

```
#Train the model
history = model.fit(X,y,validation_split=0.2, epochs=100,
batch_size=512, shuffle=True)
```

```
Epoch 1/100
9/9 [==============================] - 7s 73ms/step - loss: 1.7797 -
accuracy: 0.3971 - val_loss: 1.8630 - val_accuracy: 0.1071
Epoch 2/100
9/9 [==============================] - 0s 12ms/step - loss: 1.3714 -
accuracy: 0.6025 - val_loss: 1.7423 - val_accuracy: 0.1161
Epoch 3/100
9/9 [==============================] - 0s 12ms/step - loss: 0.9436 -
```

```
accuracy: 0.6781 - val_loss: 1.4734 - val_accuracy: 0.2688
Epoch 4/100
9/9 [==============================] - 0s 12ms/step - loss: 0.7101 -
accuracy: 0.7513 - val_loss: 1.1127 - val_accuracy: 0.4652
Epoch 5/100
9/9 [==============================] - 0s 11ms/step - loss: 0.5605 -
accuracy: 0.7955 - val_loss: 0.9059 - val_accuracy: 0.5688
Epoch 6/100
9/9 [==============================] - 0s 11ms/step - loss: 0.4656 -
accuracy: 0.8333 - val_loss: 0.7293 - val_accuracy: 0.6875
Epoch 7/100
9/9 [==============================] - 0s 11ms/step - loss: 0.3845 -
accuracy: 0.8667 - val_loss: 0.5869 - val_accuracy: 0.7723
Epoch 8/100
9/9 [==============================] - 0s 12ms/step - loss: 0.3459 -
accuracy: 0.8837 - val_loss: 0.4592 - val_accuracy: 0.8536
Epoch 9/100
9/9 [==============================] - 0s 11ms/step - loss: 0.2906 -
accuracy: 0.8989 - val_loss: 0.3793 - val_accuracy: 0.8839
Epoch 10/100
9/9 [==============================] - 0s 11ms/step - loss: 0.2375 -
accuracy: 0.9217 - val_loss: 0.3151 - val_accuracy: 0.9062
Epoch 11/100
9/9 [==============================] - 0s 11ms/step - loss: 0.2159 -
accuracy: 0.9297 - val_loss: 0.2605 - val_accuracy: 0.9241
Epoch 12/100
9/9 [==============================] - 0s 11ms/step - loss: 0.1841 -
accuracy: 0.9433 - val_loss: 0.2541 - val_accuracy: 0.9304
Epoch 13/100
9/9 [==============================] - 0s 11ms/step - loss: 0.1512 -
accuracy: 0.9547 - val_loss: 0.1885 - val_accuracy: 0.9446
Epoch 14/100
9/9 [==============================] - 0s 12ms/step - loss: 0.1376 -
accuracy: 0.9592 - val_loss: 0.1603 - val_accuracy: 0.9518
Epoch 15/100
9/9 [==============================] - 0s 12ms/step - loss: 0.1247 -
accuracy: 0.9607 - val_loss: 0.1392 - val_accuracy: 0.9554
Epoch 16/100
9/9 [==============================] - 0s 12ms/step - loss: 0.1117 -
accuracy: 0.9676 - val_loss: 0.0999 - val_accuracy: 0.9723
Epoch 17/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0868 -
accuracy: 0.9752 - val_loss: 0.1077 - val_accuracy: 0.9652
Epoch 18/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0865 -
accuracy: 0.9737 - val_loss: 0.0927 - val_accuracy: 0.9723
Epoch 19/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0825 -
accuracy: 0.9743 - val_loss: 0.1582 - val_accuracy: 0.9527
```

```
Epoch 20/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0815 -
accuracy: 0.9761 - val_loss: 0.1537 - val_accuracy: 0.9554
Epoch 21/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0720 -
accuracy: 0.9817 - val_loss: 0.0912 - val_accuracy: 0.9688
Epoch 22/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0781 -
accuracy: 0.9761 - val_loss: 0.0782 - val_accuracy: 0.9723
Epoch 23/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0828 -
accuracy: 0.9737 - val_loss: 0.1045 - val_accuracy: 0.9688
Epoch 24/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0717 -
accuracy: 0.9754 - val_loss: 0.0586 - val_accuracy: 0.9795
Epoch 25/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0609 -
accuracy: 0.9806 - val_loss: 0.0798 - val_accuracy: 0.9732
Epoch 26/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0618 -
accuracy: 0.9833 - val_loss: 0.0737 - val_accuracy: 0.9786
Epoch 27/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0504 -
accuracy: 0.9859 - val_loss: 0.0528 - val_accuracy: 0.9839
Epoch 28/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0501 -
accuracy: 0.9842 - val_loss: 0.0616 - val_accuracy: 0.9786
Epoch 29/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0443 -
accuracy: 0.9862 - val_loss: 0.0467 - val_accuracy: 0.9857
Epoch 30/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0344 -
accuracy: 0.9908 - val_loss: 0.0520 - val_accuracy: 0.9839
Epoch 31/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0380 -
accuracy: 0.9893 - val_loss: 0.0511 - val_accuracy: 0.9839
Epoch 32/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0345 -
accuracy: 0.9904 - val_loss: 0.0427 - val_accuracy: 0.9866
Epoch 33/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0360 -
accuracy: 0.9893 - val_loss: 0.0659 - val_accuracy: 0.9786
Epoch 34/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0361 -
accuracy: 0.9886 - val_loss: 0.0706 - val_accuracy: 0.9795
Epoch 35/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0413 -
accuracy: 0.9893 - val_loss: 0.0450 - val_accuracy: 0.9875
Epoch 36/100
```

```
9/9 [==============================] - 0s 12ms/step - loss: 0.0359 -
accuracy: 0.9900 - val_loss: 0.0420 - val_accuracy: 0.9830
Epoch 37/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0353 -
accuracy: 0.9891 - val_loss: 0.0782 - val_accuracy: 0.9750
Epoch 38/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0307 -
accuracy: 0.9900 - val_loss: 0.0496 - val_accuracy: 0.9821
Epoch 39/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0282 -
accuracy: 0.9920 - val_loss: 0.0649 - val_accuracy: 0.9741
Epoch 40/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0327 -
accuracy: 0.9908 - val_loss: 0.0316 - val_accuracy: 0.9893
Epoch 41/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0286 -
accuracy: 0.9917 - val_loss: 0.0417 - val_accuracy: 0.9839
Epoch 42/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0278 -
accuracy: 0.9911 - val_loss: 0.0407 - val_accuracy: 0.9857
Epoch 43/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0257 -
accuracy: 0.9911 - val_loss: 0.0197 - val_accuracy: 0.9937
Epoch 44/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0292 -
accuracy: 0.9911 - val_loss: 0.0284 - val_accuracy: 0.9902
Epoch 45/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0260 -
accuracy: 0.9926 - val_loss: 0.0227 - val_accuracy: 0.9902
Epoch 46/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0251 -
accuracy: 0.9937 - val_loss: 0.0352 - val_accuracy: 0.9884
Epoch 47/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0193 -
accuracy: 0.9933 - val_loss: 0.0191 - val_accuracy: 0.9955
Epoch 48/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0166 -
accuracy: 0.9962 - val_loss: 0.0165 - val_accuracy: 0.9955
Epoch 49/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0240 -
accuracy: 0.9940 - val_loss: 0.0182 - val_accuracy: 0.9946
Epoch 50/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0188 -
accuracy: 0.9953 - val_loss: 0.0197 - val_accuracy: 0.9937
Epoch 51/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0165 -
accuracy: 0.9955 - val_loss: 0.0139 - val_accuracy: 0.9973
Epoch 52/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0121 -
```

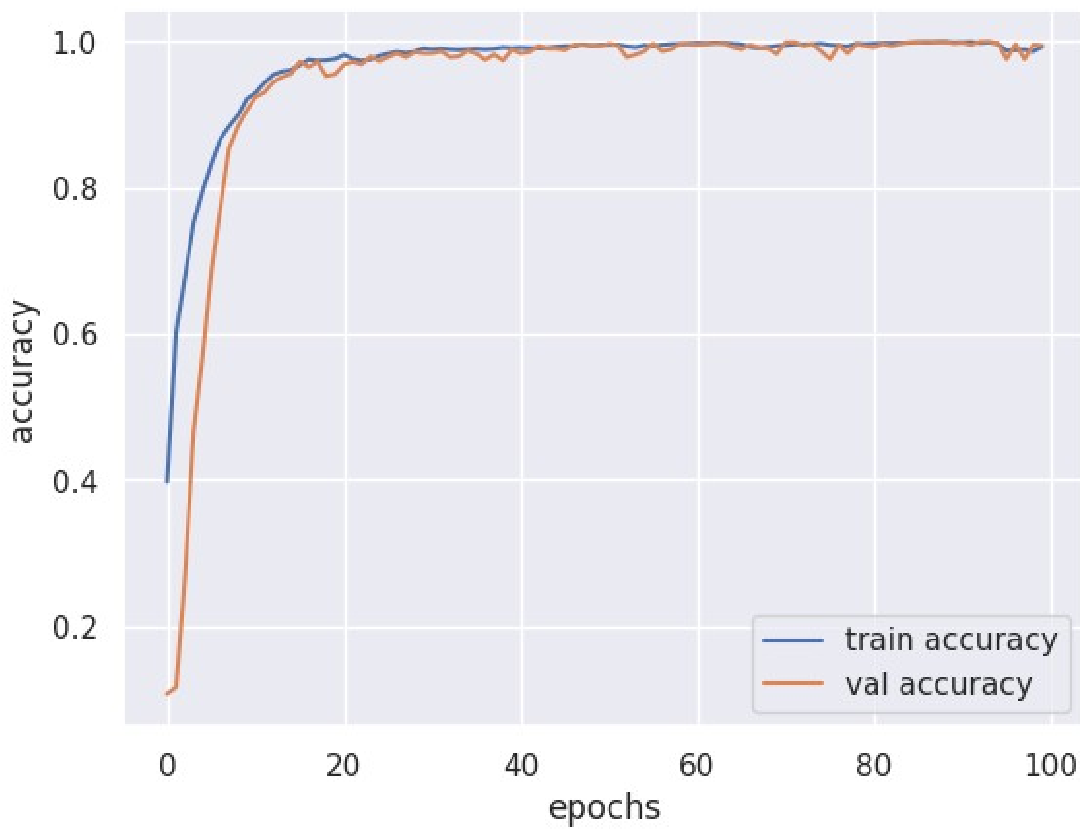
```
accuracy: 0.9958 - val_loss: 0.0173 - val_accuracy: 0.9946
Epoch 53/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0228 -
accuracy: 0.9933 - val_loss: 0.0632 - val_accuracy: 0.9786
Epoch 54/100
9/9 [==============================] - 0s 14ms/step - loss: 0.0267 -
accuracy: 0.9920 - val_loss: 0.0509 - val_accuracy: 0.9821
Epoch 55/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0190 -
accuracy: 0.9949 - val_loss: 0.0360 - val_accuracy: 0.9866
Epoch 56/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0203 -
accuracy: 0.9935 - val_loss: 0.0127 - val_accuracy: 0.9973
Epoch 57/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0142 -
accuracy: 0.9955 - val_loss: 0.0348 - val_accuracy: 0.9866
Epoch 58/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0138 -
accuracy: 0.9967 - val_loss: 0.0275 - val_accuracy: 0.9893
Epoch 59/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0127 -
accuracy: 0.9969 - val_loss: 0.0127 - val_accuracy: 0.9964
Epoch 60/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0103 -
accuracy: 0.9973 - val_loss: 0.0115 - val_accuracy: 0.9964
Epoch 61/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0081 -
accuracy: 0.9980 - val_loss: 0.0108 - val_accuracy: 0.9955
Epoch 62/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0095 -
accuracy: 0.9975 - val_loss: 0.0102 - val_accuracy: 0.9964
Epoch 63/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0075 -
accuracy: 0.9984 - val_loss: 0.0109 - val_accuracy: 0.9973
Epoch 64/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0082 -
accuracy: 0.9980 - val_loss: 0.0113 - val_accuracy: 0.9973
Epoch 65/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0101 -
accuracy: 0.9971 - val_loss: 0.0202 - val_accuracy: 0.9920
Epoch 66/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0107 -
accuracy: 0.9962 - val_loss: 0.0207 - val_accuracy: 0.9893
Epoch 67/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0249 -
accuracy: 0.9924 - val_loss: 0.0093 - val_accuracy: 0.9955
Epoch 68/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0369 -
accuracy: 0.9908 - val_loss: 0.0348 - val_accuracy: 0.9911
```

```
Epoch 69/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0294 -
accuracy: 0.9924 - val_loss: 0.0296 - val_accuracy: 0.9902
Epoch 70/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0196 -
accuracy: 0.9940 - val_loss: 0.0406 - val_accuracy: 0.9830
Epoch 71/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0152 -
accuracy: 0.9949 - val_loss: 0.0061 - val_accuracy: 0.9991
Epoch 72/100
9/9 [==============================] - 0s 13ms/step - loss: 0.0112 -
accuracy: 0.9964 - val_loss: 0.0050 - val_accuracy: 0.9991
Epoch 73/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0104 -
accuracy: 0.9962 - val_loss: 0.0196 - val_accuracy: 0.9937
Epoch 74/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0126 -
accuracy: 0.9964 - val_loss: 0.0100 - val_accuracy: 0.9964
Epoch 75/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0122 -
accuracy: 0.9973 - val_loss: 0.0362 - val_accuracy: 0.9866
Epoch 76/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0128 -
accuracy: 0.9953 - val_loss: 0.0890 - val_accuracy: 0.9759
Epoch 77/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0207 -
accuracy: 0.9944 - val_loss: 0.0098 - val_accuracy: 0.9955
Epoch 78/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0204 -
accuracy: 0.9933 - val_loss: 0.0403 - val_accuracy: 0.9839
Epoch 79/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0096 -
accuracy: 0.9973 - val_loss: 0.0073 - val_accuracy: 0.9964
Epoch 80/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0118 -
accuracy: 0.9967 - val_loss: 0.0128 - val_accuracy: 0.9937
Epoch 81/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0094 -
accuracy: 0.9973 - val_loss: 0.0202 - val_accuracy: 0.9920
Epoch 82/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0091 -
accuracy: 0.9978 - val_loss: 0.0104 - val_accuracy: 0.9964
Epoch 83/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0075 -
accuracy: 0.9980 - val_loss: 0.0110 - val_accuracy: 0.9937
Epoch 84/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0069 -
accuracy: 0.9982 - val_loss: 0.0056 - val_accuracy: 0.9973
Epoch 85/100
```

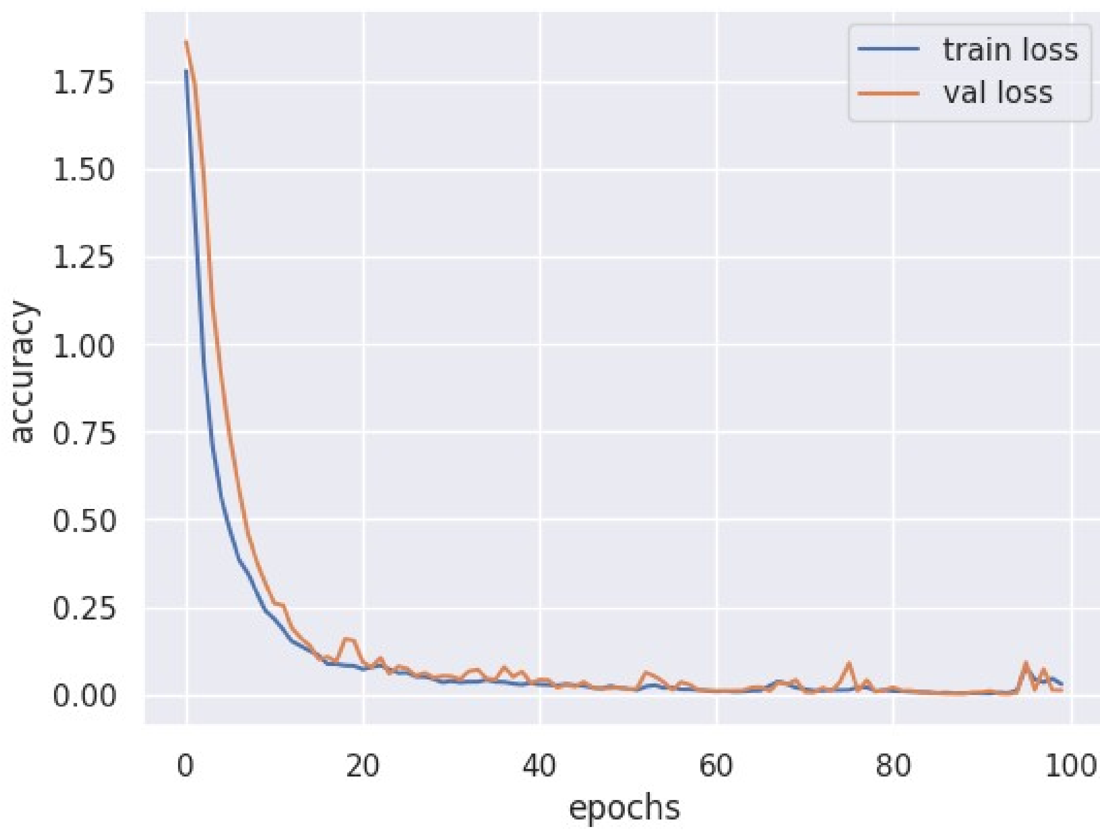
```
9/9 [==============================] - 0s 11ms/step - loss: 0.0064 -
accuracy: 0.9984 - val_loss: 0.0032 - val_accuracy: 0.9991
Epoch 86/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0035 -
accuracy: 0.9993 - val_loss: 0.0032 - val_accuracy: 0.9991
Epoch 87/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0046 -
accuracy: 0.9989 - val_loss: 0.0017 - val_accuracy: 1.0000
Epoch 88/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0028 -
accuracy: 0.9996 - val_loss: 0.0026 - val_accuracy: 0.9991
Epoch 89/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0030 -
accuracy: 0.9998 - val_loss: 0.0018 - val_accuracy: 1.0000
Epoch 90/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0034 -
accuracy: 0.9989 - val_loss: 0.0058 - val_accuracy: 0.9973
Epoch 91/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0030 -
accuracy: 0.9991 - val_loss: 0.0064 - val_accuracy: 0.9982
Epoch 92/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0027 -
accuracy: 0.9996 - val_loss: 0.0105 - val_accuracy: 0.9955
Epoch 93/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0054 -
accuracy: 0.9980 - val_loss: 0.0019 - val_accuracy: 1.0000
Epoch 94/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0036 -
accuracy: 0.9989 - val_loss: 9.8965e-04 - val_accuracy: 1.0000
Epoch 95/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0126 -
accuracy: 0.9969 - val_loss: 0.0050 - val_accuracy: 0.9973
Epoch 96/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0774 -
accuracy: 0.9875 - val_loss: 0.0910 - val_accuracy: 0.9759
Epoch 97/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0431 -
accuracy: 0.9884 - val_loss: 0.0123 - val_accuracy: 0.9964
Epoch 98/100
9/9 [==============================] - 0s 11ms/step - loss: 0.0349 -
accuracy: 0.9891 - val_loss: 0.0720 - val_accuracy: 0.9759
Epoch 99/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0450 -
accuracy: 0.9864 - val_loss: 0.0125 - val_accuracy: 0.9955
Epoch 100/100
9/9 [==============================] - 0s 12ms/step - loss: 0.0297 -
accuracy: 0.9929 - val_loss: 0.0118 - val_accuracy: 0.9946
```

plot the results

```
epochs = range(100)
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()
```




```
epochs = range(100)
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('accuracy')
plt.legend()
plt.show()
```


train loss
val loss
accuracy
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
epochs