Speech Emotion Recognition by using Ravdess Dataset

```
# Loading the Dataset
import pandas as pd
import numpy as np
import os

# EDA
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

# Feature Extraction
import librosa
import librosa.display
from IPython.display import Audio

# Scaling
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder

# Training and Testing
from sklearn.model_selection import train_test_split

# Model Building
# LSTM
from keras.models import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Embedding,
BatchNormalization, Flatten
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical

# CNN
from tensorflow.keras.layers import Conv2D, Conv1D, MaxPooling2D,
MaxPooling1D

# Prediction
# Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier

# Accuracy and Loss Epochs
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

```python
# Metrics Evaluation
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score, confusion_matrix,
classification_report
from sklearn.metrics import precision_recall_fscore_support

paths = []
labels = []
for dirname, _,filenames in os.walk("/Zidio Internship Data Science
Program/Projects/audio_speech_actors_01-24"):
    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
```

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

```
['/Zidio Internship Data Science
Program/Projects/audio_speech_actors_01-24\\03-01-01-01-01-01-01.wav',
 '/Zidio Internship Data Science
Program/Projects/audio_speech_actors_01-24\\03-01-01-01-01-01-02.wav',
 '/Zidio Internship Data Science
Program/Projects/audio_speech_actors_01-24\\03-01-01-01-01-01-03.wav',
 '/Zidio Internship Data Science
Program/Projects/audio_speech_actors_01-24\\03-01-01-01-01-01-04.wav',
 '/Zidio Internship Data Science
Program/Projects/audio_speech_actors_01-24\\03-01-01-01-01-01-05.wav']
```

```python
# Dictionary mapping for emotion labels
emotion_dic = {
    '01': 'neutral',
    '02': 'calm',
    '03': 'happy',
    '04': 'sad',
    '05': 'angry',
    '06': 'fear',
    '07': 'disgust',
    '08': 'surprised'
}

# Lists to store paths and mapped emotion labels
emotion_list = []
for path in paths:
    filename = os.path.basename(path)
    parts = filename.split('-')
```

```
    emotion_code = parts[2]  # Assuming the emotion code is in the
third position (index 2)
    emotion_label = emotion_dic.get(emotion_code, "unknown")
    emotion_list.append(emotion_label)

# Convert to DataFrame if needed
emotion_df = pd.DataFrame({
    'path': paths,
    'emotion': emotion_list
})

# Display the first few rows to verify
emotion_df.head()
```

```
                                                path  emotion
0  /Zidio Internship Data Science Program/Project...  neutral
1  /Zidio Internship Data Science Program/Project...  neutral
2  /Zidio Internship Data Science Program/Project...  neutral
3  /Zidio Internship Data Science Program/Project...  neutral
4  /Zidio Internship Data Science Program/Project...  neutral
```

```
emotion_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1440 entries, 0 to 1439
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   path     1440 non-null   object
 1   emotion  1440 non-null   object
dtypes: object(2)
memory usage: 22.6+ KB
```

```
emotion_df['emotion'].unique()
```

```
array(['neutral', 'calm', 'happy', 'sad', 'angry', 'fear', 'disgust',
       'surprised'], dtype=object)
```

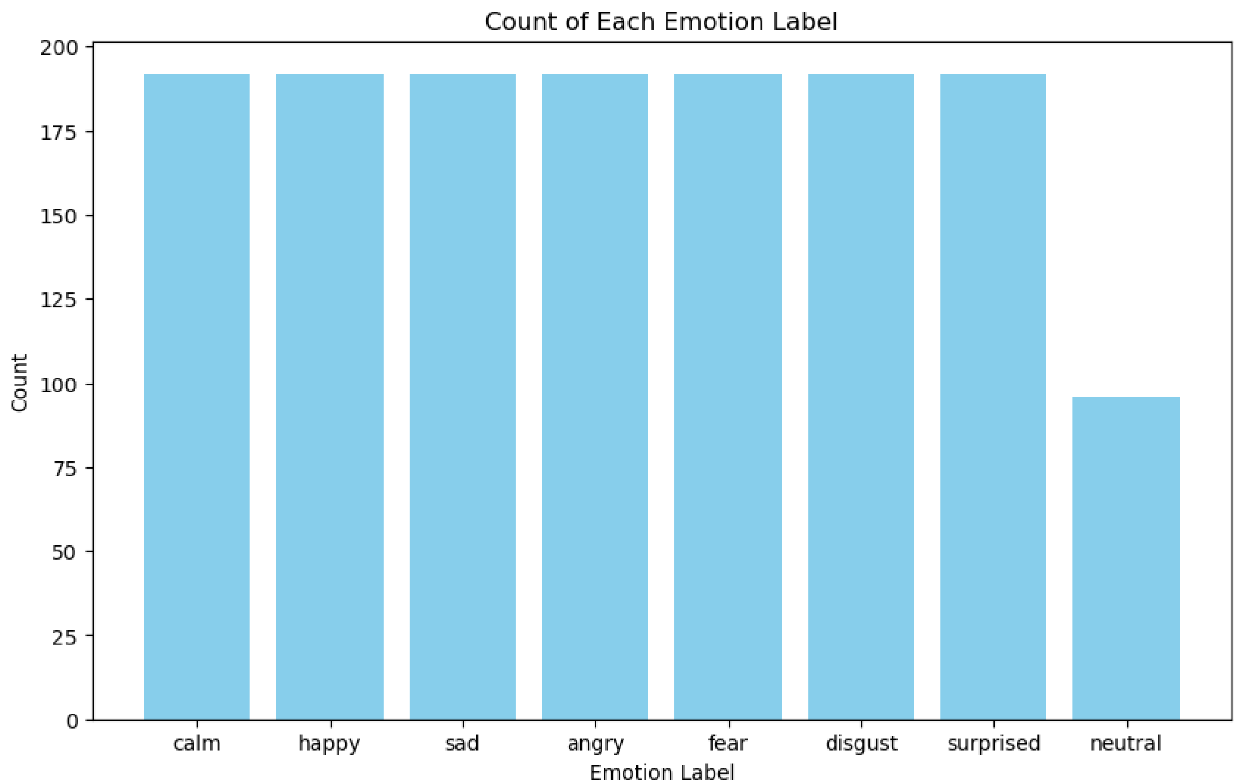
```
emotion_df['emotion'].value_counts()
```

```
emotion
calm         192
happy        192
sad          192
angry        192
fear         192
disgust      192
surprised    192
neutral       96
Name: count, dtype: int64
```

```
label_counts = emotion_df['emotion'].value_counts()

label_counts
```

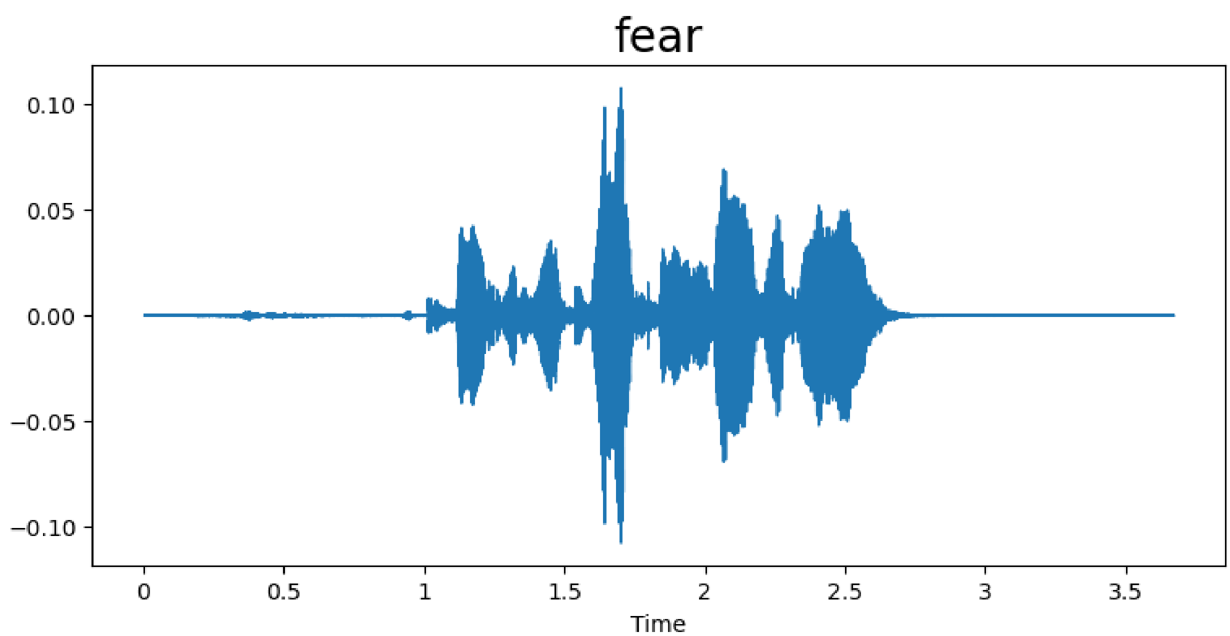
```
emotion
calm         192
happy        192
sad          192
angry        192
fear         192
disgust      192
surprised    192
neutral       96
Name: count, dtype: int64
```

```
plt.figure(figsize=(10, 6))
plt.bar(label_counts.index, label_counts.values, color='skyblue')
plt.xlabel('Emotion Label')
plt.ylabel('Count')
plt.title('Count of Each Emotion Label')
plt.xticks(rotation=360)
plt.show()
```

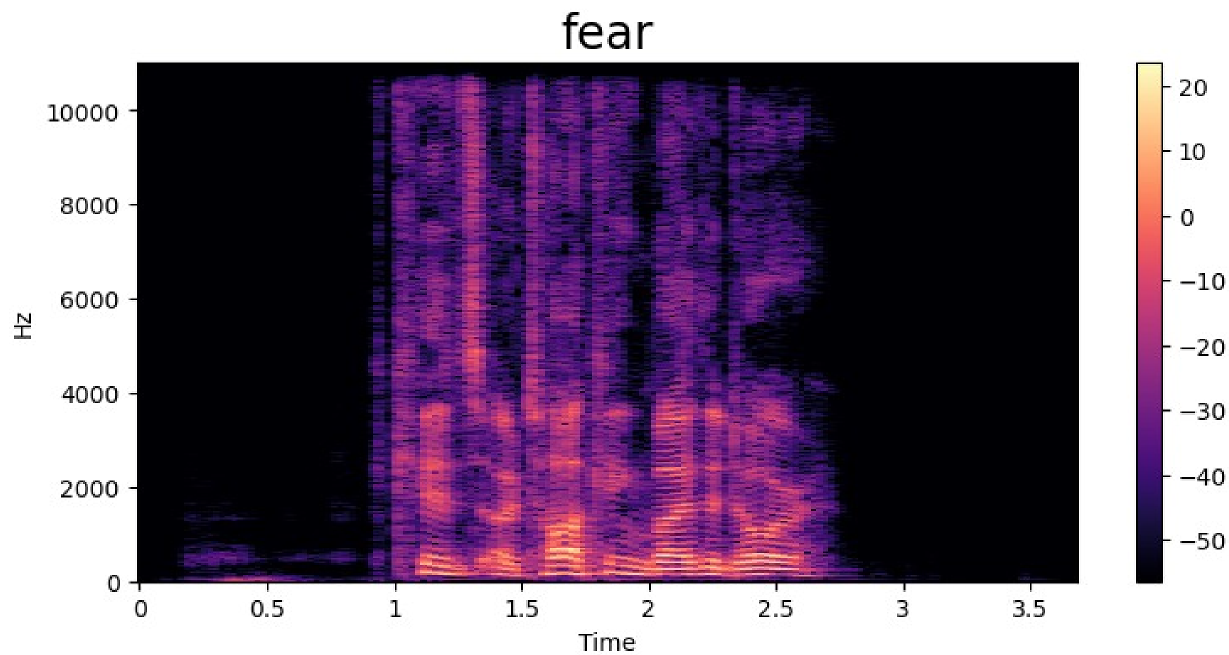



```
def waveplot(data,sr,emotion):
    plt.figure(figsize=(9,4))
```

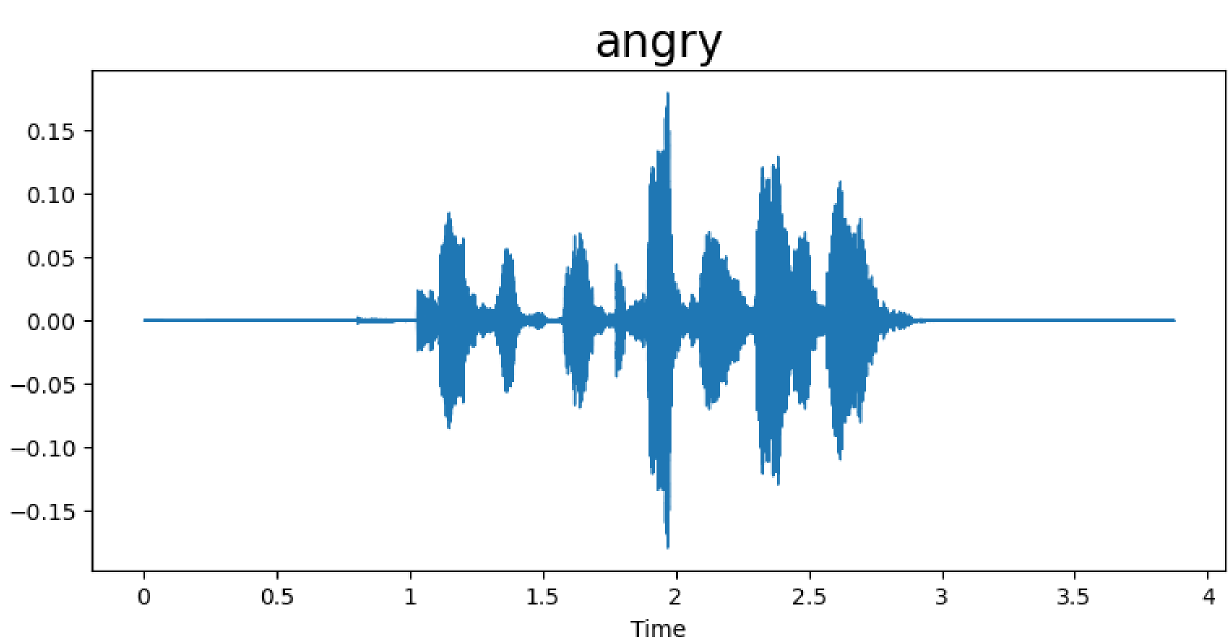
```
    plt.title(emotion,size=20)
    librosa.display.waveshow(data,sr=sr)
    plt.show()

def spectogram(data,sr,emotion):
    x = librosa.stft(data)
    xdb = librosa.amplitude_to_db(abs(x))
    plt.figure(figsize=(9,4))
    plt.title(emotion,size=20)
    librosa.display.specshow(xdb,sr=sr,x_axis="time",y_axis='hz')
    plt.colorbar()

emotion = 'fear'
path = np.array(emotion_df['path'][emotion_df['emotion']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data,sampling_rate,emotion)
spectogram(data,sampling_rate,emotion)
Audio(path)
```



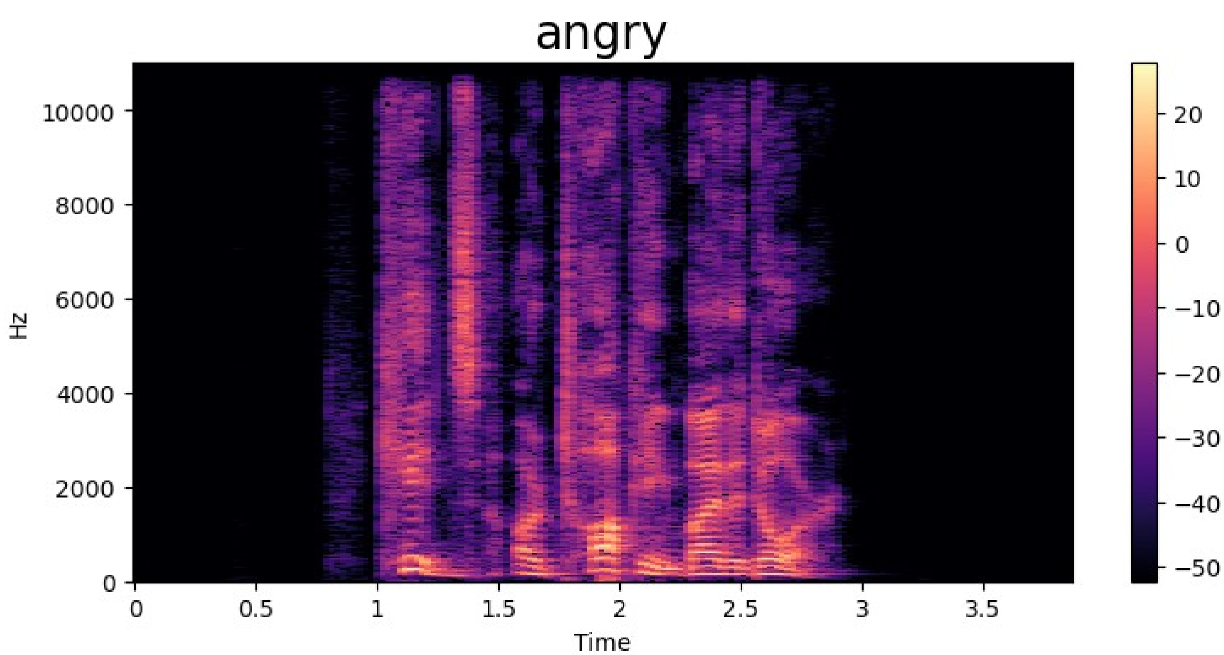

```
<IPython.lib.display.Audio object>
```



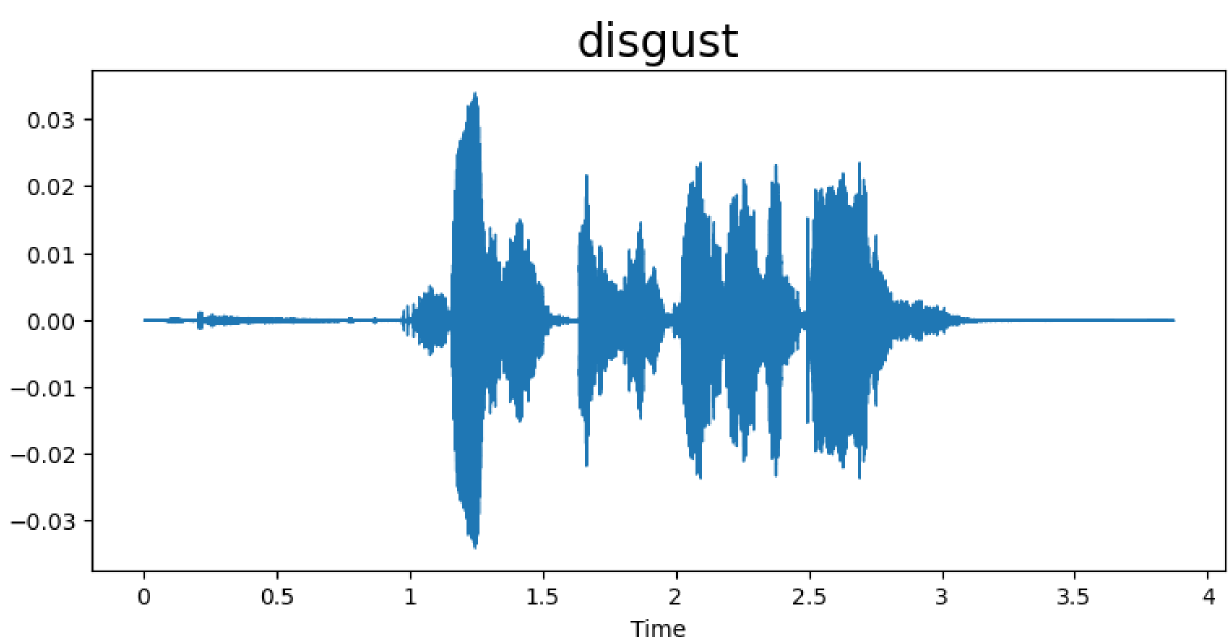

```
emotion = 'angry'
path = np.array(emotion_df['path'][emotion_df['emotion']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data,sampling_rate,emotion)
spectogram(data,sampling_rate,emotion)
Audio(path)
```



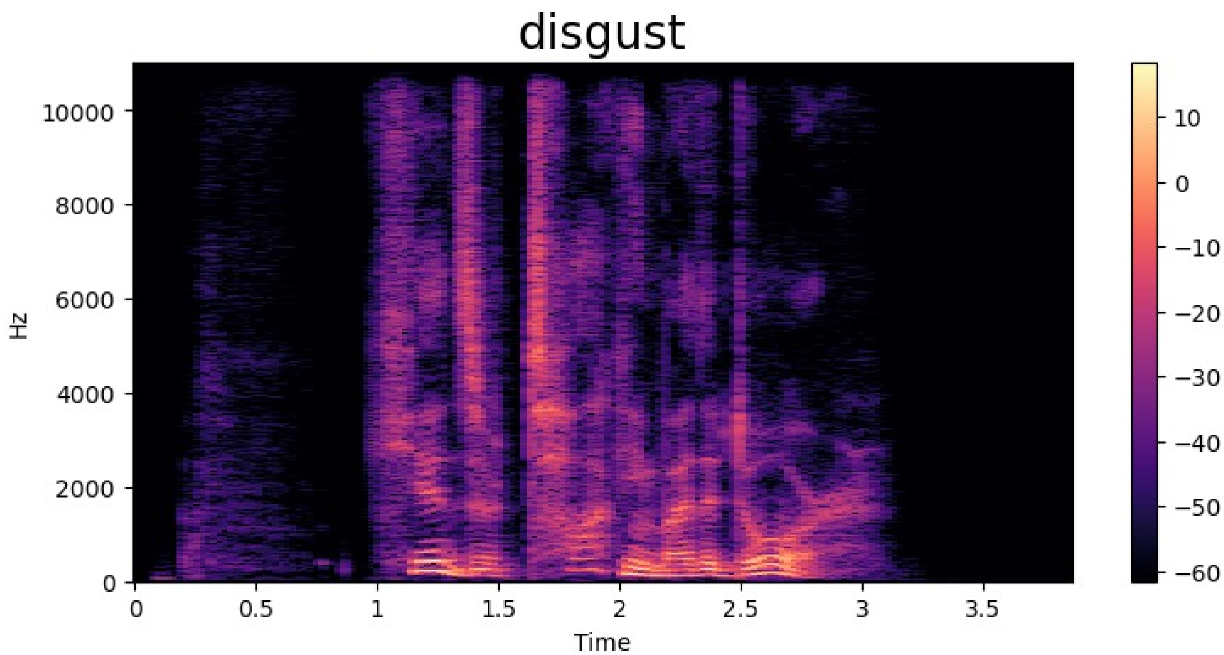

```
<IPython.lib.display.Audio object>
```



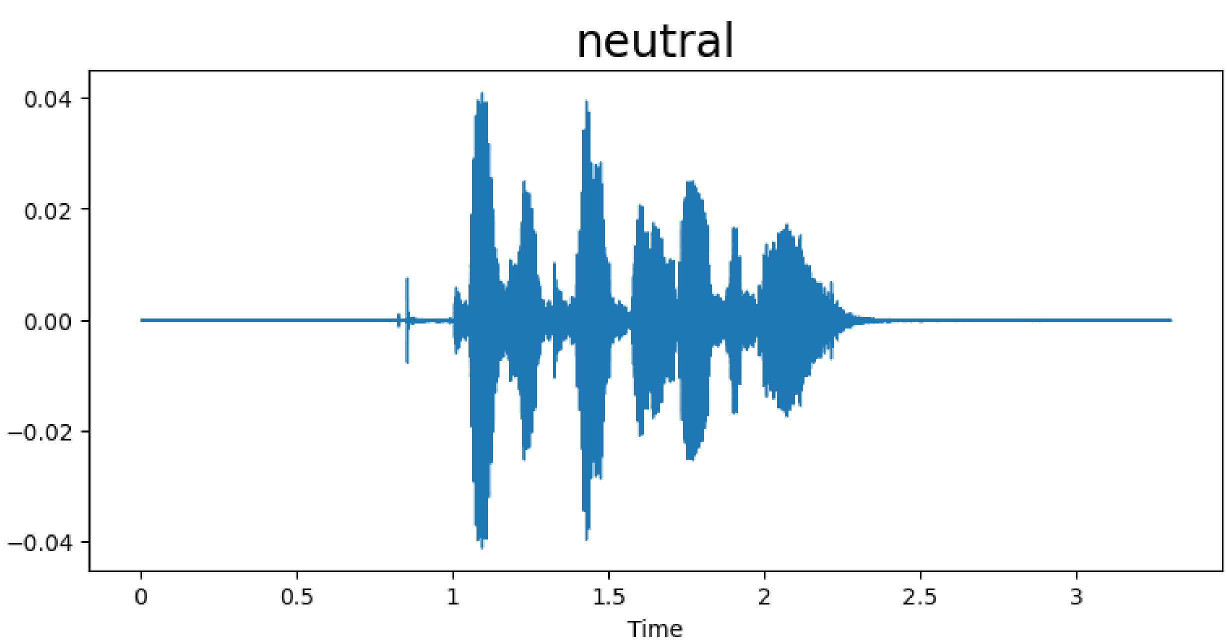

```
emotion = 'disgust'
path = np.array(emotion_df['path'][emotion_df['emotion']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data,sampling_rate,emotion)
spectogram(data,sampling_rate,emotion)
Audio(path)
```



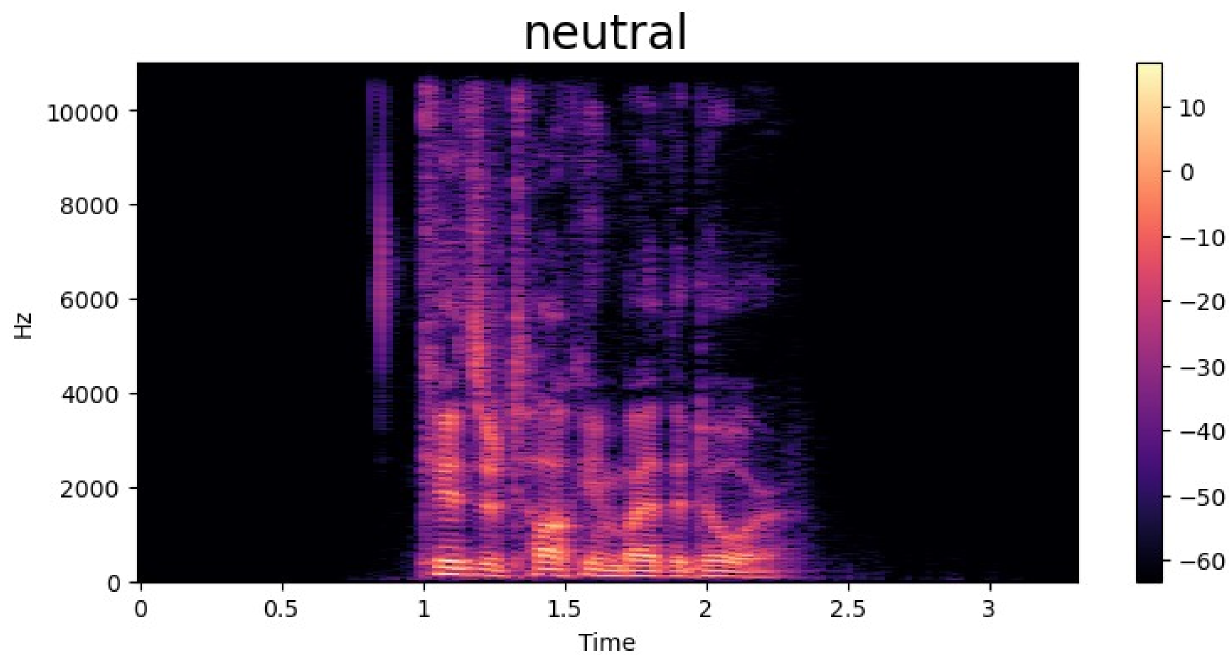

```
<IPython.lib.display.Audio object>
```



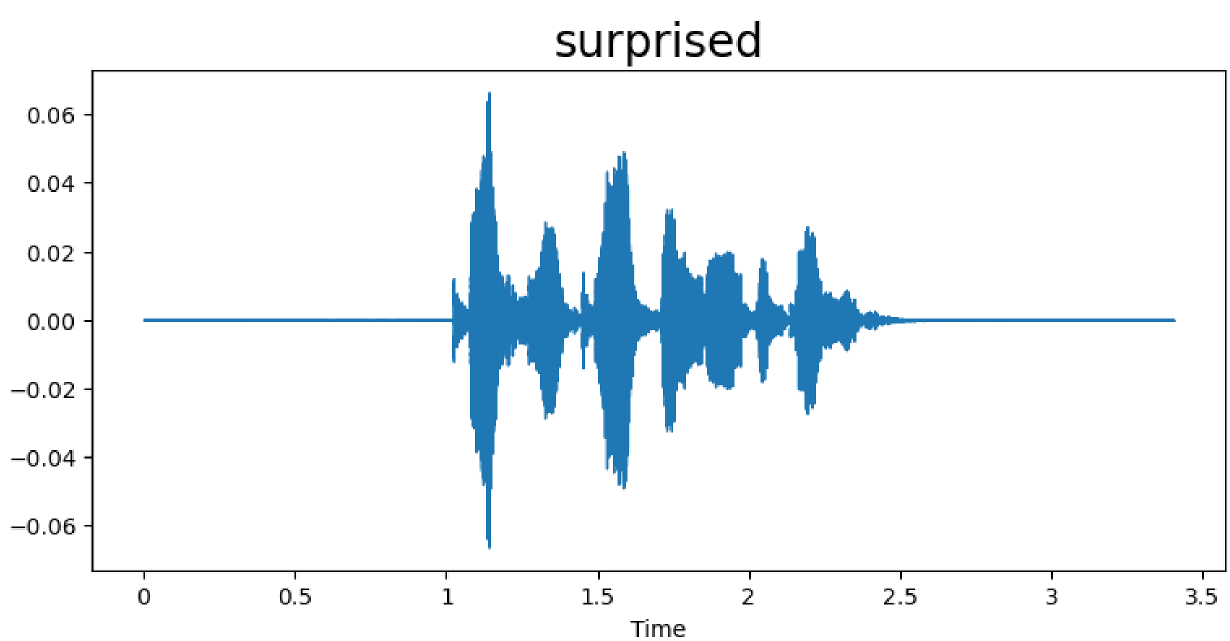

```
emotion = 'neutral'
path = np.array(emotion_df['path'][emotion_df['emotion']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data,sampling_rate,emotion)
spectogram(data,sampling_rate,emotion)
Audio(path)
```



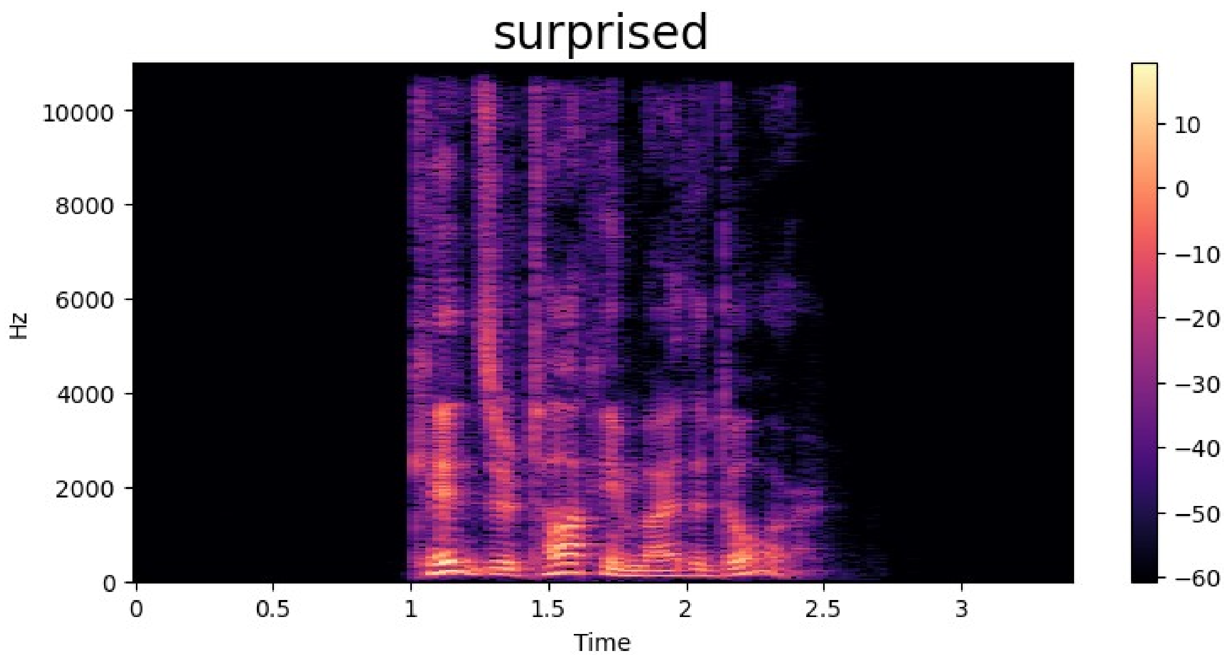

```
<IPython.lib.display.Audio object>
```



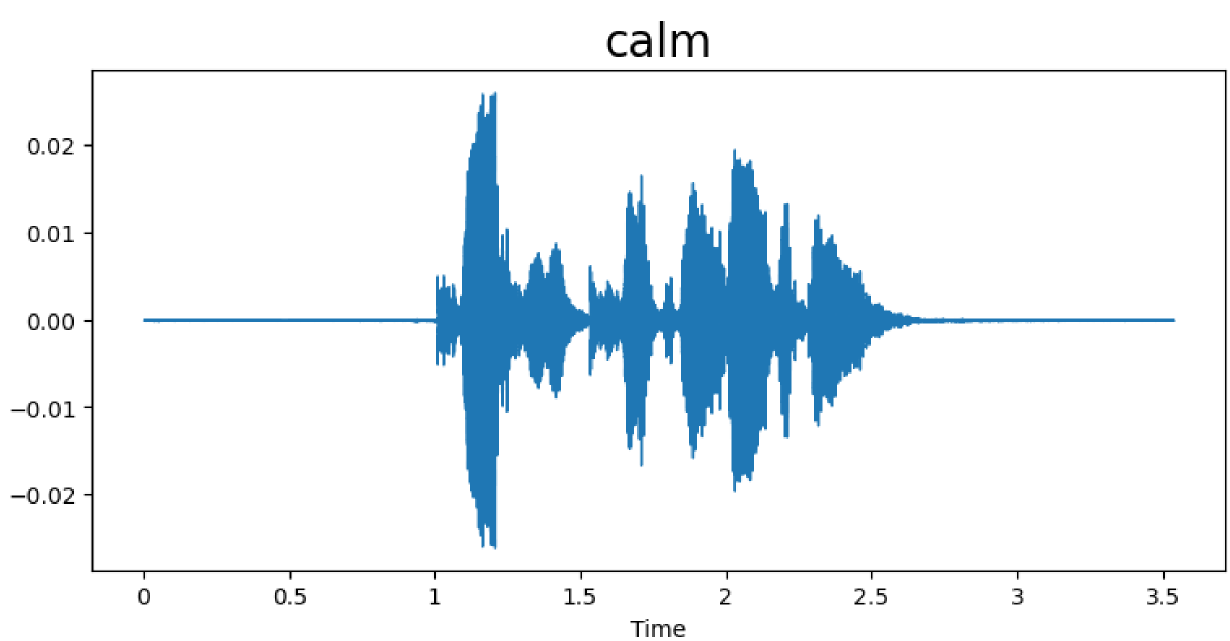

```
emotion = 'surprised'
path = np.array(emotion_df['path'][emotion_df['emotion']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data,sampling_rate,emotion)
spectogram(data,sampling_rate,emotion)
Audio(path)
```



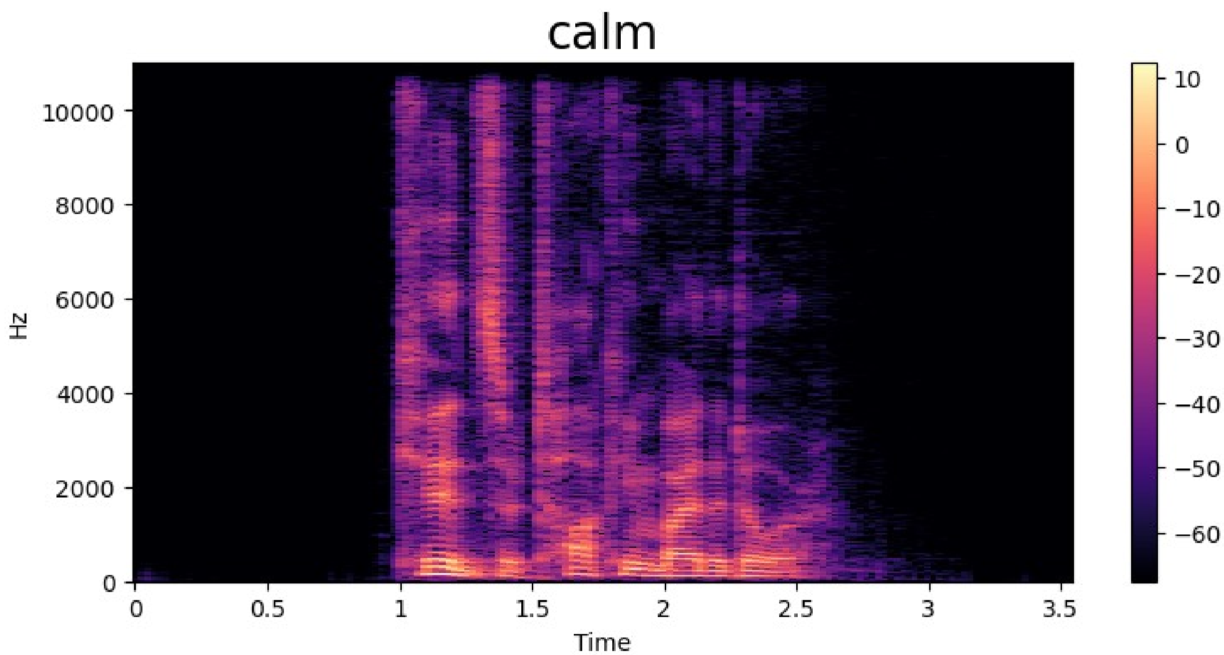

```
<IPython.lib.display.Audio object>
```



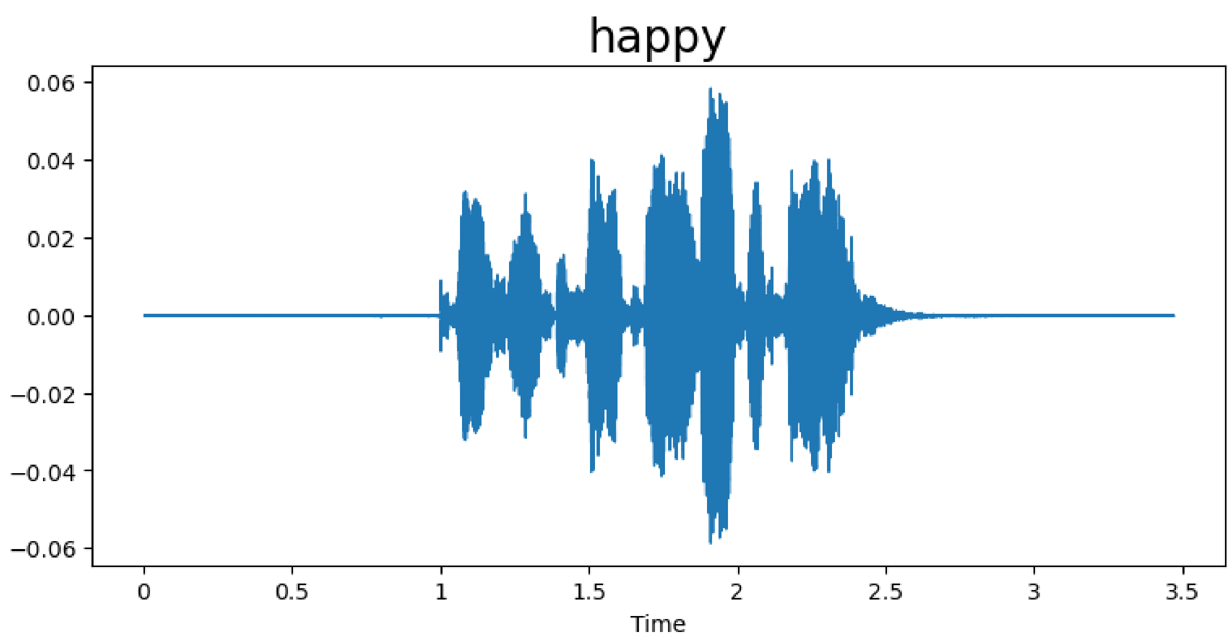

```
emotion = 'calm'
path = np.array(emotion_df['path'][emotion_df['emotion']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data,sampling_rate,emotion)
spectogram(data,sampling_rate,emotion)
Audio(path)
```



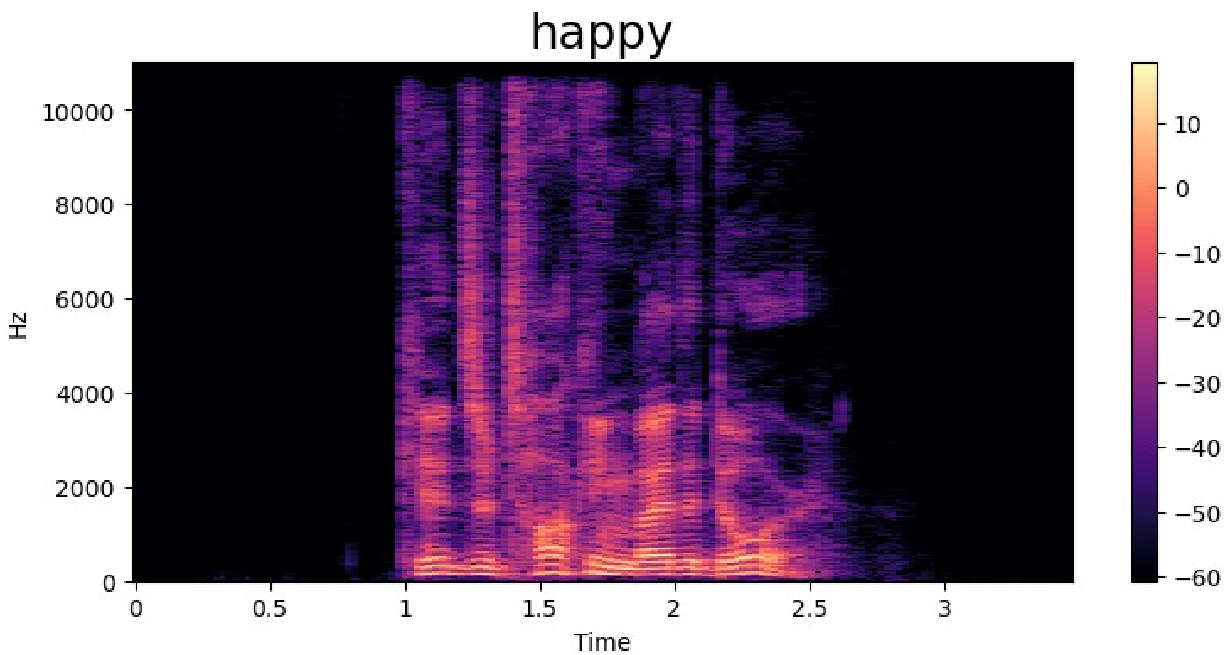

```
<IPython.lib.display.Audio object>
```




```
emotion = 'happy'
path = np.array(emotion_df['path'][emotion_df['emotion']==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>
```



Extracting Features

```
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(emotion_df['path'][0])

array([-6.7019543e+02,  6.5063850e+01,  8.8895434e-01,  1.4715980e+01,
        9.1821651e+00,  6.6057473e-01, -3.8468361e+00, -3.5839462e+00,
       -1.2959006e+01, -3.3001330e+00,  9.1077948e-01, -3.5970359e+00,
        2.3762746e+00, -4.3889413e+00,  5.4508036e-01,  8.9185160e-01,
       -4.8025908e+00, -2.1054137e+00, -1.6059692e+00, -1.0523903e+00,
       -7.0672808e+00, -6.2306100e-01, -2.7280300e+00, -5.3073611e+00,
       -1.9691167e+00, -9.4615275e-01, -5.7211108e+00,  3.3299121e-01,
       -2.5438452e+00,  1.8220837e-01, -2.3510976e+00, -2.5047269e+00,
       -3.1515074e+00, -2.1908991e+00, -3.8017602e+00, -1.8130876e+00,
       -1.2612224e+00, -2.1449544e+00, -4.1521730e+00, -
1.7796154e+00],
      dtype=float32)

x_mfcc = emotion_df['path'].apply(lambda x: extract_mfcc(x))

x_mfcc

0       [-670.19543, 65.06385, 0.88895434, 14.71598, 9...
1       [-614.99786, 64.27647, -12.088927, 9.41706, -5...
2       [-585.2709, 65.953766, -3.3687484, 12.292631, ...
3       [-663.22577, 50.415646, -8.746946, 10.637102, ...
```

```
4       [-683.20856, 80.11589, 7.9506874, 16.284683, 1...
                              ...
1435    [-560.3368, 38.236683, -20.307865, 4.6272163, ...
1436    [-484.9119, 52.183716, -3.460623, 12.275539, 3...
1437    [-496.7466, 29.797121, -16.75864, -7.013239, -...
1438    [-498.1352, 39.734634, -13.461919, -2.2899392,...
1439    [-485.24686, 34.517685, -2.6824563, -2.7403817...
Name: path, Length: 1440, dtype: object
```

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

```
(1440, 40)
```

Standard Scaling

```
enc = OneHotEncoder(sparse=False)  # sparse=False to get an array
directly
y = enc.fit_transform(emotion_df[['emotion']])
```

```
y.shape
```

```
(1440, 8)
```

Training and Testing the data

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
# Check shapes of the resulting datasets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (1152, 40)
X_test shape: (288, 40)
y_train shape: (1152, 8)
y_test shape: (288, 8)
```

```
# Input Split
x_train = np.expand_dims(X_train,-1)
x_train.shape
```

```
(1152, 40, 1)
```

```
# Input Split
x_test = np.expand_dims(X_test,-1)
x_test.shape
```

```
(288, 40, 1)
```

```python
#Create LSTM Model
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

model = Sequential([
    LSTM(128,return_sequences=False,input_shape=(40,1)),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32,activation='relu'),
    Dropout(0.2),
    Dense(8,activation='softmax')
])

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

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                 ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
━━━━━━━━━━━━━━━━━┩
│ lstm (LSTM)                           │ (None, 128)                  │
66,560 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dense (Dense)                         │ (None, 64)                   │
8,256 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dropout (Dropout)                     │ (None, 64)                   │
0 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dense_1 (Dense)                       │ (None, 32)                   │
2,080 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dropout_1 (Dropout)                   │ (None, 32)                   │
0 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dense_2 (Dense)                       │ (None, 8)                    │
264 │
```

```
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 77,160 (301.41 KB)

 Trainable params: 77,160 (301.41 KB)

 Non-trainable params: 0 (0.00 B)

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

# Train the model
history =
model.fit(X_train,y_train,epochs=100,batch_size=512,validation_data =
(X_test, y_test))

Epoch 1/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 117ms/step - accuracy: 0.9958 - loss:
0.0130 - val_accuracy: 1.0000 - val_loss: 0.0052
Epoch 2/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.9980 - loss:
0.0127 - val_accuracy: 1.0000 - val_loss: 0.0049
Epoch 3/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9959 - loss:
0.0127 - val_accuracy: 1.0000 - val_loss: 0.0048
Epoch 4/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9988 - loss:
0.0093 - val_accuracy: 1.0000 - val_loss: 0.0049
Epoch 5/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.9988 - loss:
0.0097 - val_accuracy: 1.0000 - val_loss: 0.0050
Epoch 6/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9977 - loss:
0.0105 - val_accuracy: 1.0000 - val_loss: 0.0048
Epoch 7/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9984 - loss:
0.0118 - val_accuracy: 1.0000 - val_loss: 0.0045
Epoch 8/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9984 - loss:
0.0113 - val_accuracy: 1.0000 - val_loss: 0.0043
Epoch 9/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.9970 - loss:
0.0115 - val_accuracy: 1.0000 - val_loss: 0.0042
Epoch 10/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9977 - loss:
0.0123 - val_accuracy: 1.0000 - val_loss: 0.0041
Epoch 11/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 0.9982 - loss:
0.0121 - val_accuracy: 1.0000 - val_loss: 0.0039
Epoch 12/100
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9993 - loss:
0.0092 - val_accuracy: 1.0000 - val_loss: 0.0038
Epoch 13/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.9982 - loss:
0.0126 - val_accuracy: 1.0000 - val_loss: 0.0038
Epoch 14/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step - accuracy: 0.9975 - loss:
0.0118 - val_accuracy: 1.0000 - val_loss: 0.0037
Epoch 15/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.9966 - loss:
0.0135 - val_accuracy: 1.0000 - val_loss: 0.0035
Epoch 16/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.9988 - loss:
0.0107 - val_accuracy: 1.0000 - val_loss: 0.0034
Epoch 17/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9977 - loss:
0.0134 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 18/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9934 - loss:
0.0137 - val_accuracy: 1.0000 - val_loss: 0.0032
Epoch 19/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 102ms/step - accuracy: 0.9963 - loss:
0.0126 - val_accuracy: 1.0000 - val_loss: 0.0032
Epoch 20/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.9954 - loss:
0.0140 - val_accuracy: 1.0000 - val_loss: 0.0032
Epoch 21/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 1.0000 - loss:
0.0084 - val_accuracy: 1.0000 - val_loss: 0.0032
Epoch 22/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 0.9968 - loss:
0.0118 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 23/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9970 - loss:
0.0116 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 24/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.9993 - loss:
0.0064 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 25/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9972 - loss:
0.0098 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 26/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9966 - loss:
0.0115 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 27/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step - accuracy: 0.9965 - loss:
0.0120 - val_accuracy: 1.0000 - val_loss: 0.0034
Epoch 28/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9940 - loss:
```

```
0.0143 - val_accuracy: 1.0000 - val_loss: 0.0034
Epoch 29/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9982 - loss: 
0.0116 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 30/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9945 - loss: 
0.0137 - val_accuracy: 1.0000 - val_loss: 0.0034
Epoch 31/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9966 - loss: 
0.0133 - val_accuracy: 1.0000 - val_loss: 0.0036
Epoch 32/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.9988 - loss: 
0.0107 - val_accuracy: 1.0000 - val_loss: 0.0038
Epoch 33/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 1.0000 - loss: 
0.0066 - val_accuracy: 1.0000 - val_loss: 0.0040
Epoch 34/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.9968 - loss: 
0.0143 - val_accuracy: 1.0000 - val_loss: 0.0042
Epoch 35/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 1.0000 - loss: 
0.0093 - val_accuracy: 1.0000 - val_loss: 0.0042
Epoch 36/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.9964 - loss: 
0.0135 - val_accuracy: 1.0000 - val_loss: 0.0042
Epoch 37/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9958 - loss: 
0.0116 - val_accuracy: 1.0000 - val_loss: 0.0040
Epoch 38/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9988 - loss: 
0.0086 - val_accuracy: 1.0000 - val_loss: 0.0038
Epoch 39/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.9945 - loss: 
0.0157 - val_accuracy: 1.0000 - val_loss: 0.0036
Epoch 40/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9982 - loss: 
0.0112 - val_accuracy: 1.0000 - val_loss: 0.0035
Epoch 41/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9984 - loss: 
0.0092 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 42/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9993 - loss: 
0.0087 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 43/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9984 - loss: 
0.0115 - val_accuracy: 1.0000 - val_loss: 0.0032
Epoch 44/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9973 - loss: 
0.0140 - val_accuracy: 1.0000 - val_loss: 0.0033
```

```
Epoch 45/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9988 - loss: 0.0094 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 46/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9977 - loss: 0.0095 - val_accuracy: 1.0000 - val_loss: 0.0034
Epoch 47/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.9984 - loss: 0.0097 - val_accuracy: 1.0000 - val_loss: 0.0038
Epoch 48/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step - accuracy: 0.9985 - loss: 0.0082 - val_accuracy: 1.0000 - val_loss: 0.0044
Epoch 49/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9982 - loss: 0.0096 - val_accuracy: 0.9965 - val_loss: 0.0051
Epoch 50/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step - accuracy: 0.9956 - loss: 0.0122 - val_accuracy: 0.9965 - val_loss: 0.0057
Epoch 51/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step - accuracy: 0.9980 - loss: 0.0105 - val_accuracy: 0.9965 - val_loss: 0.0058
Epoch 52/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9984 - loss: 0.0128 - val_accuracy: 0.9965 - val_loss: 0.0056
Epoch 53/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9968 - loss: 0.0128 - val_accuracy: 0.9965 - val_loss: 0.0052
Epoch 54/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.9982 - loss: 0.0095 - val_accuracy: 0.9965 - val_loss: 0.0051
Epoch 55/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 0.9982 - loss: 0.0103 - val_accuracy: 0.9965 - val_loss: 0.0049
Epoch 56/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9982 - loss: 0.0117 - val_accuracy: 0.9965 - val_loss: 0.0049
Epoch 57/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step - accuracy: 0.9988 - loss: 0.0081 - val_accuracy: 0.9965 - val_loss: 0.0050
Epoch 58/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step - accuracy: 0.9970 - loss: 0.0083 - val_accuracy: 0.9965 - val_loss: 0.0053
Epoch 59/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9970 - loss: 0.0112 - val_accuracy: 0.9965 - val_loss: 0.0054
Epoch 60/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 0.9993 - loss: 0.0058 - val_accuracy: 0.9965 - val_loss: 0.0054
Epoch 61/100
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - accuracy: 0.9986 - loss:
0.0110 - val_accuracy: 0.9965 - val_loss: 0.0052
Epoch 62/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.9993 - loss:
0.0092 - val_accuracy: 0.9965 - val_loss: 0.0053
Epoch 63/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9970 - loss:
0.0111 - val_accuracy: 0.9965 - val_loss: 0.0052
Epoch 64/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.9977 - loss:
0.0106 - val_accuracy: 0.9965 - val_loss: 0.0052
Epoch 65/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9970 - loss:
0.0111 - val_accuracy: 0.9965 - val_loss: 0.0052
Epoch 66/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9996 - loss:
0.0056 - val_accuracy: 0.9965 - val_loss: 0.0051
Epoch 67/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step - accuracy: 0.9953 - loss:
0.0133 - val_accuracy: 1.0000 - val_loss: 0.0047
Epoch 68/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.9982 - loss:
0.0131 - val_accuracy: 1.0000 - val_loss: 0.0042
Epoch 69/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 104ms/step - accuracy: 0.9965 - loss:
0.0126 - val_accuracy: 1.0000 - val_loss: 0.0037
Epoch 70/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 1.0000 - loss:
0.0075 - val_accuracy: 1.0000 - val_loss: 0.0034
Epoch 71/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 1.0000 - loss:
0.0070 - val_accuracy: 1.0000 - val_loss: 0.0032
Epoch 72/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 0.9993 - loss:
0.0071 - val_accuracy: 1.0000 - val_loss: 0.0031
Epoch 73/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9982 - loss:
0.0095 - val_accuracy: 1.0000 - val_loss: 0.0031
Epoch 74/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9993 - loss:
0.0062 - val_accuracy: 1.0000 - val_loss: 0.0032
Epoch 75/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 105ms/step - accuracy: 1.0000 - loss:
0.0093 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 76/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9947 - loss:
0.0164 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 77/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.9977 - loss:
```

```
0.0082 - val_accuracy: 1.0000 - val_loss: 0.0035
Epoch 78/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9982 - loss:
0.0087 - val_accuracy: 1.0000 - val_loss: 0.0037
Epoch 79/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 0.9988 - loss:
0.0100 - val_accuracy: 1.0000 - val_loss: 0.0038
Epoch 80/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9988 - loss:
0.0081 - val_accuracy: 1.0000 - val_loss: 0.0038
Epoch 81/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 1.0000 - loss:
0.0070 - val_accuracy: 1.0000 - val_loss: 0.0039
Epoch 82/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.9965 - loss:
0.0144 - val_accuracy: 1.0000 - val_loss: 0.0039
Epoch 83/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.9963 - loss:
0.0126 - val_accuracy: 1.0000 - val_loss: 0.0039
Epoch 84/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9982 - loss:
0.0081 - val_accuracy: 1.0000 - val_loss: 0.0039
Epoch 85/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 0.9993 - loss:
0.0075 - val_accuracy: 1.0000 - val_loss: 0.0039
Epoch 86/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.9988 - loss:
0.0086 - val_accuracy: 1.0000 - val_loss: 0.0039
Epoch 87/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9977 - loss:
0.0102 - val_accuracy: 1.0000 - val_loss: 0.0038
Epoch 88/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9982 - loss:
0.0098 - val_accuracy: 1.0000 - val_loss: 0.0038
Epoch 89/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9977 - loss:
0.0100 - val_accuracy: 1.0000 - val_loss: 0.0036
Epoch 90/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 106ms/step - accuracy: 0.9986 - loss:
0.0084 - val_accuracy: 1.0000 - val_loss: 0.0034
Epoch 91/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9958 - loss:
0.0140 - val_accuracy: 1.0000 - val_loss: 0.0032
Epoch 92/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9993 - loss:
0.0073 - val_accuracy: 1.0000 - val_loss: 0.0031
Epoch 93/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step - accuracy: 1.0000 - loss:
0.0053 - val_accuracy: 1.0000 - val_loss: 0.0030
```

```
Epoch 94/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - accuracy: 0.9977 - loss:
0.0082 - val_accuracy: 1.0000 - val_loss: 0.0029
Epoch 95/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9989 - loss:
0.0076 - val_accuracy: 1.0000 - val_loss: 0.0029
Epoch 96/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.9970 - loss:
0.0118 - val_accuracy: 1.0000 - val_loss: 0.0028
Epoch 97/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 0.9996 - loss:
0.0069 - val_accuracy: 1.0000 - val_loss: 0.0028
Epoch 98/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step - accuracy: 0.9977 - loss:
0.0071 - val_accuracy: 1.0000 - val_loss: 0.0028
Epoch 99/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 1.0000 - loss:
0.0050 - val_accuracy: 1.0000 - val_loss: 0.0028
Epoch 100/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - accuracy: 0.9975 - loss:
0.0108 - val_accuracy: 1.0000 - val_loss: 0.0027

lstm_train = model.predict(x_train)
lstm_test = model.predict(x_test)


# Initialize and train Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=200,
random_state=42)
rf_classifier.fit(lstm_train, y_train)

# Make predictions using Random Forest classifier
predictions_lstm = rf_classifier.predict(lstm_test)

36/36 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step

model = Sequential([
    Conv1D(filters=32, kernel_size=3, activation='relu',
input_shape=(40, 1)),
    MaxPooling1D(pool_size=2),
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(8, activation='softmax')
])

#model.compile(loss='categorical_crossentropy', optimizer='adam',
```

```
metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_5"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                           ┃ Output Shape                 ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
━━━━━━━━━━━━━━━━━┩
│ conv1d_2 (Conv1D)                      │ (None, 38, 32)               │
128 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ max_pooling1d_2 (MaxPooling1D)         │ (None, 19, 32)               │
0 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ conv1d_3 (Conv1D)                      │ (None, 17, 64)               │
6,208 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ max_pooling1d_3 (MaxPooling1D)         │ (None, 8, 64)                │
0 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ flatten_1 (Flatten)                    │ (None, 512)                  │
0 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dense_14 (Dense)                       │ (None, 128)                  │
65,664 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dropout_9 (Dropout)                    │ (None, 128)                  │
0 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dense_15 (Dense)                       │ (None, 8)                    │
1,032 │
└───────────────────────────────────────┴──────────────────────────────┘
─────────────────┘

 Total params: 73,032 (285.28 KB)

 Trainable params: 73,032 (285.28 KB)

 Non-trainable params: 0 (0.00 B)
```

```
optimizer = Adam(learning_rate=0.001)
```

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

history = model.fit(X, y, batch_size=64, epochs=100, verbose=1,
validation_data = (X, y))

Epoch 1/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8694 - loss:
0.3453 - val_accuracy: 0.9278 - val_loss: 0.2173
Epoch 2/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.8722 - loss:
0.3801 - val_accuracy: 0.9410 - val_loss: 0.1912
Epoch 3/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8831 - loss:
0.3231 - val_accuracy: 0.9590 - val_loss: 0.1554
Epoch 4/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9094 - loss:
0.2651 - val_accuracy: 0.9424 - val_loss: 0.1789
Epoch 5/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.8971 - loss:
0.2912 - val_accuracy: 0.9160 - val_loss: 0.2407
Epoch 6/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.8914 - loss:
0.3132 - val_accuracy: 0.9618 - val_loss: 0.1332
Epoch 7/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9080 - loss:
0.2567 - val_accuracy: 0.9576 - val_loss: 0.1525
Epoch 8/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9140 - loss:
0.2541 - val_accuracy: 0.9646 - val_loss: 0.1252
Epoch 9/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9192 - loss:
0.2250 - val_accuracy: 0.9674 - val_loss: 0.1230
Epoch 10/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9131 - loss:
0.2516 - val_accuracy: 0.9521 - val_loss: 0.1592
Epoch 11/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9011 - loss:
0.2592 - val_accuracy: 0.9576 - val_loss: 0.1230
Epoch 12/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9085 - loss:
0.2653 - val_accuracy: 0.9722 - val_loss: 0.1142
Epoch 13/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9204 - loss:
0.2324 - val_accuracy: 0.9757 - val_loss: 0.1073
Epoch 14/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9369 - loss:
0.1805 - val_accuracy: 0.9736 - val_loss: 0.0909
Epoch 15/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9375 - loss:
```

```
0.1790 - val_accuracy: 0.9826 - val_loss: 0.0787
Epoch 16/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9387 - loss:
0.1726 - val_accuracy: 0.9694 - val_loss: 0.1088
Epoch 17/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9387 - loss:
0.1939 - val_accuracy: 0.9806 - val_loss: 0.0758
Epoch 18/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9503 - loss:
0.1606 - val_accuracy: 0.9736 - val_loss: 0.0910
Epoch 19/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9351 - loss:
0.1859 - val_accuracy: 0.9688 - val_loss: 0.0998
Epoch 20/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9407 - loss:
0.1928 - val_accuracy: 0.9868 - val_loss: 0.0696
Epoch 21/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9409 - loss:
0.1715 - val_accuracy: 0.9757 - val_loss: 0.0867
Epoch 22/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9283 - loss:
0.2059 - val_accuracy: 0.9625 - val_loss: 0.1215
Epoch 23/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9237 - loss:
0.2105 - val_accuracy: 0.9618 - val_loss: 0.1530
Epoch 24/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9228 - loss:
0.2346 - val_accuracy: 0.9667 - val_loss: 0.1342
Epoch 25/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9203 - loss:
0.2524 - val_accuracy: 0.9479 - val_loss: 0.1754
Epoch 26/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9186 - loss:
0.2491 - val_accuracy: 0.9646 - val_loss: 0.1209
Epoch 27/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.8999 - loss:
0.2788 - val_accuracy: 0.9264 - val_loss: 0.2190
Epoch 28/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.8874 - loss:
0.3281 - val_accuracy: 0.9597 - val_loss: 0.1274
Epoch 29/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9112 - loss:
0.2770 - val_accuracy: 0.9556 - val_loss: 0.1509
Epoch 30/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9184 - loss:
0.2229 - val_accuracy: 0.9639 - val_loss: 0.1139
Epoch 31/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9366 - loss:
0.2007 - val_accuracy: 0.9764 - val_loss: 0.0826
```

```
Epoch 32/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9559 - loss:
0.1484 - val_accuracy: 0.9806 - val_loss: 0.0842
Epoch 33/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9658 - loss:
0.1333 - val_accuracy: 0.9875 - val_loss: 0.0603
Epoch 34/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9671 - loss:
0.1170 - val_accuracy: 0.9944 - val_loss: 0.0363
Epoch 35/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9799 - loss:
0.0981 - val_accuracy: 0.9958 - val_loss: 0.0308
Epoch 36/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9711 - loss:
0.1029 - val_accuracy: 0.9889 - val_loss: 0.0421
Epoch 37/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9621 - loss:
0.1148 - val_accuracy: 0.9882 - val_loss: 0.0488
Epoch 38/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9468 - loss:
0.1608 - val_accuracy: 0.9715 - val_loss: 0.1073
Epoch 39/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9187 - loss:
0.2714 - val_accuracy: 0.9375 - val_loss: 0.2192
Epoch 40/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.8184 - loss:
0.5813 - val_accuracy: 0.8653 - val_loss: 0.4317
Epoch 41/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8658 - loss:
0.4218 - val_accuracy: 0.9556 - val_loss: 0.1599
Epoch 42/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9223 - loss:
0.2770 - val_accuracy: 0.9785 - val_loss: 0.0853
Epoch 43/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9465 - loss:
0.1670 - val_accuracy: 0.9792 - val_loss: 0.1001
Epoch 44/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9432 - loss:
0.1706 - val_accuracy: 0.9861 - val_loss: 0.0666
Epoch 45/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9589 - loss:
0.1234 - val_accuracy: 0.9951 - val_loss: 0.0378
Epoch 46/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9786 - loss:
0.0935 - val_accuracy: 0.9979 - val_loss: 0.0253
Epoch 47/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9780 - loss:
0.0801 - val_accuracy: 0.9986 - val_loss: 0.0208
Epoch 48/100
```

```
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9762 - loss:
0.0837 - val_accuracy: 0.9986 - val_loss: 0.0204
Epoch 49/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9751 - loss:
0.0787 - val_accuracy: 0.9958 - val_loss: 0.0253
Epoch 50/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9739 - loss:
0.0808 - val_accuracy: 0.9903 - val_loss: 0.0302
Epoch 51/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9712 - loss:
0.1092 - val_accuracy: 0.9924 - val_loss: 0.0289
Epoch 52/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9679 - loss:
0.1006 - val_accuracy: 0.9903 - val_loss: 0.0370
Epoch 53/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9661 - loss:
0.1158 - val_accuracy: 0.9875 - val_loss: 0.0487
Epoch 54/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9671 - loss:
0.1099 - val_accuracy: 0.9736 - val_loss: 0.1151
Epoch 55/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9239 - loss:
0.2759 - val_accuracy: 0.9694 - val_loss: 0.1093
Epoch 56/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9421 - loss:
0.1763 - val_accuracy: 0.9917 - val_loss: 0.0478
Epoch 57/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9670 - loss:
0.1439 - val_accuracy: 0.9861 - val_loss: 0.0556
Epoch 58/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9704 - loss:
0.1262 - val_accuracy: 0.9931 - val_loss: 0.0401
Epoch 59/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9704 - loss:
0.1063 - val_accuracy: 0.9958 - val_loss: 0.0253
Epoch 60/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9748 - loss:
0.0780 - val_accuracy: 0.9924 - val_loss: 0.0359
Epoch 61/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9556 - loss:
0.1261 - val_accuracy: 0.9896 - val_loss: 0.0471
Epoch 62/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9718 - loss:
0.0844 - val_accuracy: 0.9917 - val_loss: 0.0318
Epoch 63/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9585 - loss:
0.1147 - val_accuracy: 0.9868 - val_loss: 0.0576
Epoch 64/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9637 - loss:
```

```
0.1156 - val_accuracy: 0.9875 - val_loss: 0.0465
Epoch 65/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9697 - loss:
0.1046 - val_accuracy: 0.9937 - val_loss: 0.0293
Epoch 66/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9522 - loss:
0.1737 - val_accuracy: 0.9819 - val_loss: 0.0608
Epoch 67/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9201 - loss:
0.2660 - val_accuracy: 0.9785 - val_loss: 0.0877
Epoch 68/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9215 - loss:
0.2206 - val_accuracy: 0.9660 - val_loss: 0.1279
Epoch 69/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9154 - loss:
0.2488 - val_accuracy: 0.9646 - val_loss: 0.1025
Epoch 70/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9310 - loss:
0.2027 - val_accuracy: 0.9715 - val_loss: 0.0868
Epoch 71/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9530 - loss:
0.1459 - val_accuracy: 0.9937 - val_loss: 0.0413
Epoch 72/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9701 - loss:
0.1173 - val_accuracy: 0.9979 - val_loss: 0.0213
Epoch 73/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9786 - loss:
0.0782 - val_accuracy: 0.9937 - val_loss: 0.0283
Epoch 74/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9812 - loss:
0.0768 - val_accuracy: 0.9993 - val_loss: 0.0122
Epoch 75/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9814 - loss:
0.0556 - val_accuracy: 0.9979 - val_loss: 0.0177
Epoch 76/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9823 - loss:
0.0706 - val_accuracy: 0.9979 - val_loss: 0.0122
Epoch 77/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.9837 - loss:
0.0574 - val_accuracy: 0.9979 - val_loss: 0.0153
Epoch 78/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9863 - loss:
0.0505 - val_accuracy: 0.9958 - val_loss: 0.0175
Epoch 79/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9746 - loss:
0.0805 - val_accuracy: 0.9965 - val_loss: 0.0210
Epoch 80/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9678 - loss:
0.1016 - val_accuracy: 0.9910 - val_loss: 0.0352
```

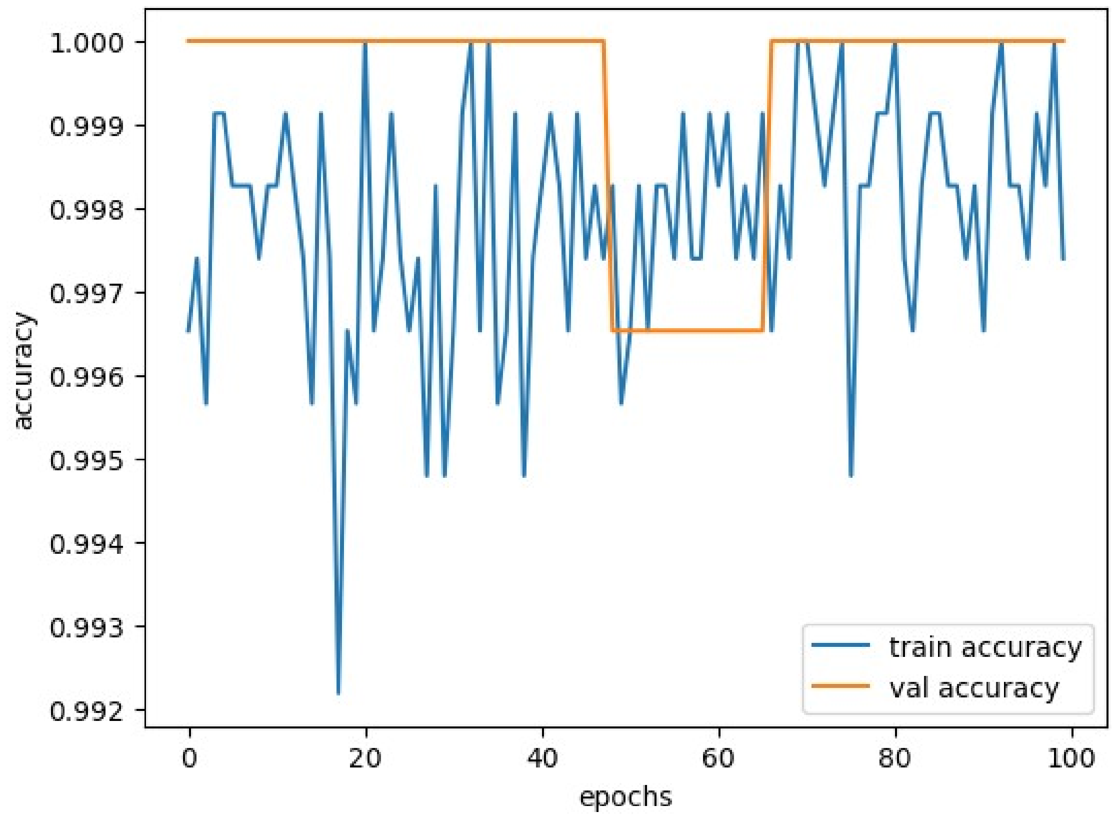
```
Epoch 81/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9629 - loss:
0.1029 - val_accuracy: 0.9889 - val_loss: 0.0427
Epoch 82/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9679 - loss:
0.0922 - val_accuracy: 0.9847 - val_loss: 0.0429
Epoch 83/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9629 - loss:
0.1161 - val_accuracy: 0.9944 - val_loss: 0.0342
Epoch 84/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.9842 - loss:
0.0661 - val_accuracy: 0.9660 - val_loss: 0.1079
Epoch 85/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.9554 - loss:
0.1665 - val_accuracy: 0.9764 - val_loss: 0.0890
Epoch 86/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9593 - loss:
0.1260 - val_accuracy: 0.9875 - val_loss: 0.0387
Epoch 87/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9657 - loss:
0.0999 - val_accuracy: 0.9667 - val_loss: 0.0950
Epoch 88/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9392 - loss:
0.2065 - val_accuracy: 0.9590 - val_loss: 0.1375
Epoch 89/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.8949 - loss:
0.3345 - val_accuracy: 0.9417 - val_loss: 0.1780
Epoch 90/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9226 - loss:
0.2764 - val_accuracy: 0.9819 - val_loss: 0.0722
Epoch 91/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9608 - loss:
0.1260 - val_accuracy: 0.9993 - val_loss: 0.0244
Epoch 92/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9704 - loss:
0.0850 - val_accuracy: 0.9972 - val_loss: 0.0199
Epoch 93/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9840 - loss:
0.0623 - val_accuracy: 0.9944 - val_loss: 0.0242
Epoch 94/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9825 - loss:
0.0563 - val_accuracy: 0.9993 - val_loss: 0.0107
Epoch 95/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9847 - loss:
0.0405 - val_accuracy: 0.9986 - val_loss: 0.0107
Epoch 96/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9854 - loss:
0.0509 - val_accuracy: 0.9889 - val_loss: 0.0399
Epoch 97/100
```

```
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9839 - loss:
0.0542 - val_accuracy: 0.9993 - val_loss: 0.0087
Epoch 98/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9839 - loss:
0.0530 - val_accuracy: 1.0000 - val_loss: 0.0055
Epoch 99/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9900 - loss:
0.0437 - val_accuracy: 0.9979 - val_loss: 0.0101
Epoch 100/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9874 - loss:
0.0413 - val_accuracy: 0.9993 - val_loss: 0.0057
```

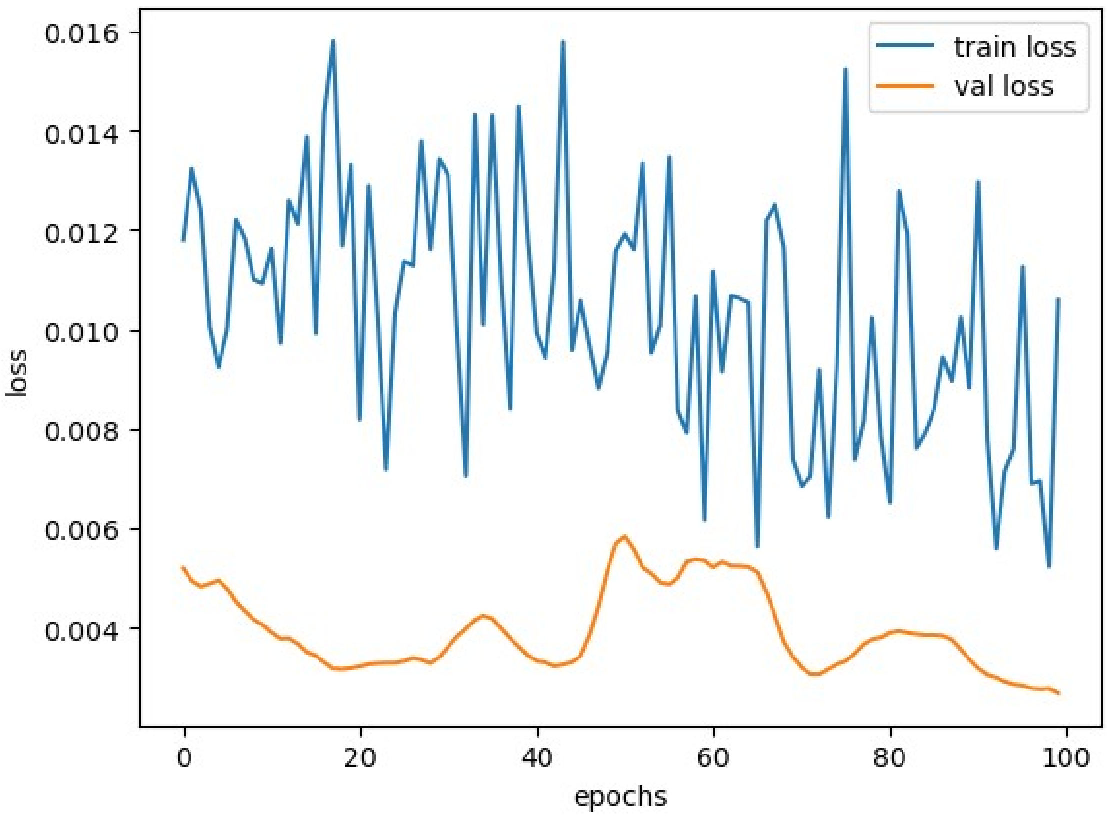
```
cnn_train = model.predict(x_train)
cnn_test = model.predict(x_test)


# Initialize and train Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=200,
random_state=42)
rf_classifier.fit(cnn_train, y_train)

# Make predictions using Random Forest classifier
predictions_cnn = rf_classifier.predict(cnn_test)
```

```
36/36 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
```

```
#Plot the results

epochs = list(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()
```




```
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()
```




```
accuracy_cnn = accuracy_score(y_test,predictions_cnn)
print("Accuracy: ", accuracy_cnn)

Accuracy:  0.9965277777777778

precision_cnn = precision_score(y_test, predictions_cnn,
average='weighted')
print("Precision: ", precision_cnn)

Precision:  0.9966066919191918

recall_cnn = recall_score(y_test, predictions_cnn, average='weighted')
print("Recall: ", recall_cnn)

Recall:  0.9965277777777778

precision_f1 = precision_score(y_test, predictions_cnn,
average='weighted')
recall_f1 = recall_score(y_test, predictions_cnn, average='weighted')

f1_score_cnn = 2 * (precision_f1 * recall_f1) / (precision_f1 +
recall_f1)

print("F1-Score: ", f1_score_cnn)

F1-Score:  0.9965672332862615
```

LSTM Metrics

```
accuracy_lstm = accuracy_score(y_test,predictions_lstm)
print("Accuracy: ", accuracy_lstm)

Accuracy:  0.9965277777777778

precision_lstm = precision_score(y_test, predictions_lstm,
average='weighted')
print("Precision: ", precision_lstm)

Precision:  0.9966066919191918

recall_lstm = recall_score(y_test, predictions_lstm,
average='weighted')
print("Recall: ", recall_lstm)

Recall:  0.9965277777777778

precision_f1 = precision_score(y_test, predictions_lstm,
average='weighted')
recall_f1 = recall_score(y_test, predictions_lstm, average='weighted')

f1_score_lstm = 2 * (precision_f1 * recall_f1) / (precision_f1 +
recall_f1)

print("F1-Score: ", f1_score_lstm)

F1-Score:  0.9965672332862615

# Results for each model
results = {
    'Model': ['CNN', 'LSTM'],
    'Accuracy': [accuracy_cnn, accuracy_lstm],
    'Precision': [precision_cnn, precision_lstm],
    'Recall': [recall_cnn, recall_lstm],
    'F1 Score': [f1_score_cnn, f1_score_lstm]
}

# Create DataFrame
results_df = pd.DataFrame(results)

# Display DataFrame
results_df
```

	Model	Accuracy	Precision	Recall	F1 Score
0	CNN	0.996528	0.996607	0.996528	0.996567
1	LSTM	0.996528	0.996607	0.996528	0.996567

CNN and LSTM model predicted the speech emotion recognition with 100% Accuracy.