```
#importing libraries
import pandas as pd
import numpy as np
import os
import sys

import librosa
import librosa.display
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split


import IPython.display as ipd
from IPython.display import Audio
import keras
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM,BatchNormalization , GRU
from keras.preprocessing.text import Tokenizer

from tensorflow.keras.utils import to_categorical
from keras.layers import Input, Flatten, Dropout, Activation
from keras.layers import Conv1D, MaxPooling1D, AveragePooling1D
from keras.models import Model
from keras.callbacks import ModelCheckpoint,
EarlyStopping,ReduceLROnPlateau
from tensorflow.keras.optimizers import SGD
import warnings
if not sys.warnoptions:
    warnings.simplefilter("ignore")
warnings.filterwarnings("ignore", category=DeprecationWarning)
import tensorflow as tf
print ("Done")
```

```
Done
```

```
#Importing from Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
dataframe =
"/content/drive/MyDrive/actoraudiofiles/audio_speech_actors_01-24"
```

```
data_directory = os.listdir(dataframe)
print(data_directory)
```

```
['Actor_15', 'Actor_17', 'Actor_23', 'Actor_24', 'Actor_22',
'Actor_20', 'Actor_16', 'Actor_19', 'Actor_21', 'Actor_18',
'Actor_06', 'Actor_09', 'Actor_12', 'Actor_11', 'Actor_07',
'Actor_13', 'Actor_14', 'Actor_05', 'Actor_10', 'Actor_08',
'Actor_03', 'Actor_01', 'Actor_04', 'Actor_02']
```

```
file_emotion = []
file_path = []

for dir in data_directory:
    actor = os.listdir(os.path.join(dataframe, dir))
    for file in actor:
        part = file.split('.')[0]
        part = part.split('-')
        file_emotion.append(int(part[2]))
        file_path.append(os.path.join(dataframe, dir, file))

emotion_data = pd.DataFrame(file_emotion, columns=['Emotions'])
path_data = pd.DataFrame(file_path, columns=['Path'])
data_1 = pd.concat([emotion_data, path_data], axis=1)

data_1.Emotions.replace({1: 'neutral', 2: 'calm', 3: 'happy', 4:
'sad', 5: 'angry', 6: 'fear', 7: 'disgust', 8: 'surprise'},
inplace=True)
data_1.head()
```

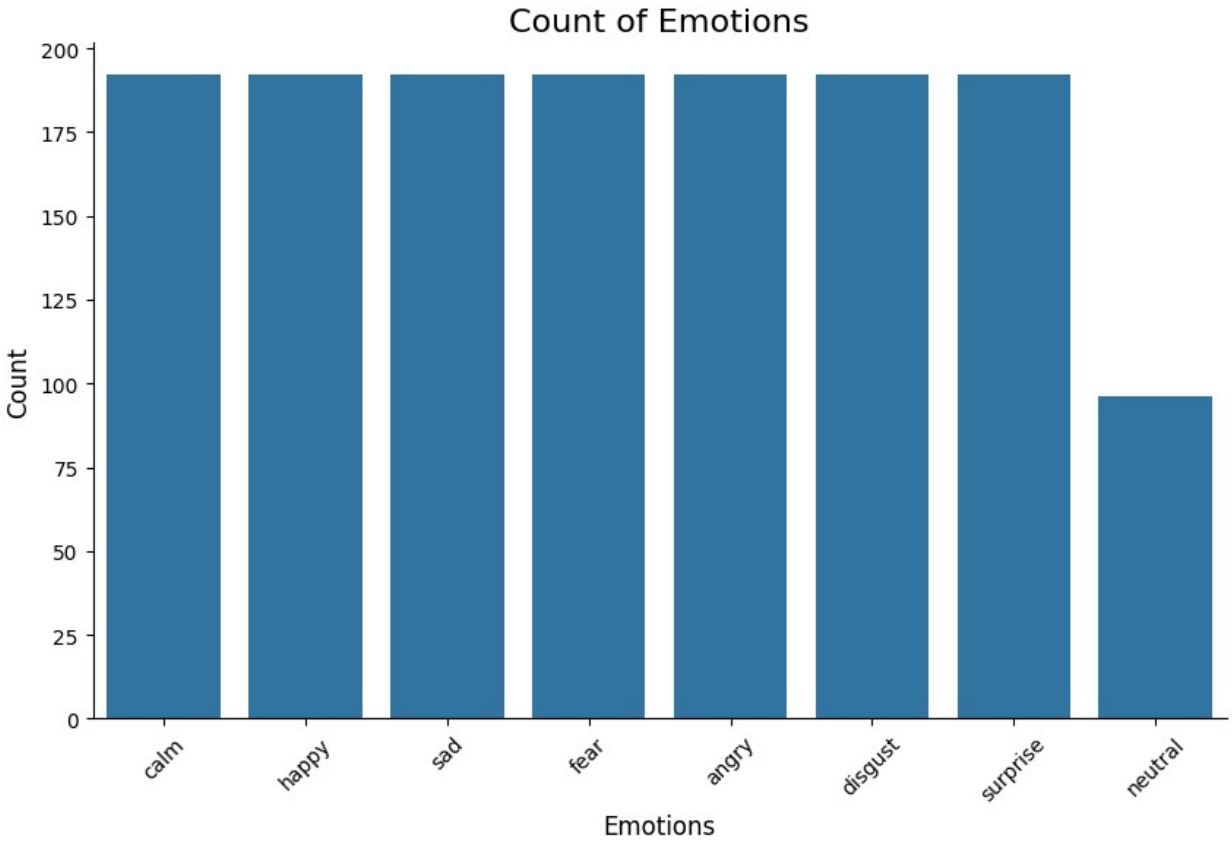
```
  Emotions                                               Path
0  neutral  /content/drive/MyDrive/actoraudiofiles/audio_s...
1     calm  /content/drive/MyDrive/actoraudiofiles/audio_s...
2     calm  /content/drive/MyDrive/actoraudiofiles/audio_s...
3     calm  /content/drive/MyDrive/actoraudiofiles/audio_s...
4     calm  /content/drive/MyDrive/actoraudiofiles/audio_s...
```

```
print(data_1.Emotions.value_counts())
```

```
calm        192
happy       192
sad         192
fear        192
angry       192
disgust     192
surprise    192
neutral      96
Name: Emotions, dtype: int64
```

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

```
emotion_counts = data_1['Emotions'].value_counts()
plt.figure(figsize=(10, 6))
plt.title('Count of Emotions', size=16)
sns.barplot(x=emotion_counts.index, y=emotion_counts.values)
plt.ylabel('Count', size=12)
plt.xlabel('Emotions', size=12)
plt.xticks(rotation=45)
sns.despine(top=True, right=True, left=False, bottom=False)
```



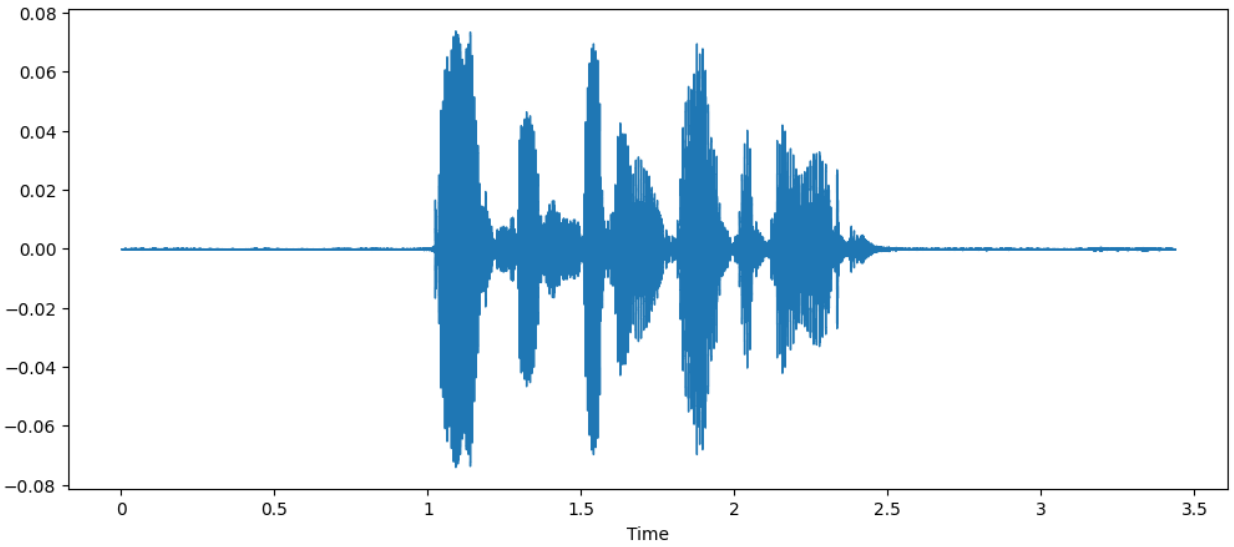

```
data_array, sample_rate = librosa.load(data_1['Path'].iloc[0])
sample_rate
```

```
22050
```

```
def noise(data):
    noise_amp = 0.045*np.random.uniform()*np.amax(data)
    data = data + noise_amp*np.random.normal(size=data.shape[0])
    return data


def stretch(data, rate=0.8):
    return librosa.effects.time_stretch(data,rate=rate)

def shift(data):
```

```
    shift_range = int(np.random.uniform(low=-5, high = 5)*1000)
    return np.roll(data, shift_range)

def pitch(data, sampling_rate, pitch_factor=0.7):
    return librosa.effects.pitch_shift(data, sr=sampling_rate,
n_steps=pitch_factor)

plt.figure(figsize=(12, 5))
librosa.display.waveshow(y=data_array, sr=sample_rate)
ipd.Audio(data_array,rate=sample_rate)
```

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




```
def zcr(data,frame_length,hop_length):

zcr=librosa.feature.zero_crossing_rate(y=data,frame_length=frame_lengt
h,hop_length=hop_length)
    return np.squeeze(zcr)
def rmse(data,frame_length=2048,hop_length=512):

rmse=librosa.feature.rms(y=data,frame_length=frame_length,hop_length=h
op_length)
    return np.squeeze(rmse)
#mfcc
def mfcc(data,sr,frame_length=2048,hop_length=512,flatten:bool=True):
    mfcc=librosa.feature.mfcc(y=data,sr=sr)
    return np.squeeze(mfcc.T)if not flatten else np.ravel(mfcc.T)

def extract_features(data,sr=22050,frame_length=2048,hop_length=512):
    result=np.array([])

    result=np.hstack((result,
```

```
                      zcr(data,frame_length,hop_length),
                      rmse(data,frame_length,hop_length),
                      mfcc(data,sr,frame_length,hop_length)
                     ))
    return result

def get_features(path,duration=2.5, offset=0.6):
    data,sr=librosa.load(path = path,duration=duration,offset=offset)
    aud=extract_features(data)
    audio=np.array(aud)
    noised_audio=noise(data)
    aud2=extract_features(noised_audio)
    audio=np.vstack((audio,aud2))

    pitched_audio=pitch(data,sr)
    aud3=extract_features(pitched_audio)
    audio=np.vstack((audio,aud3))

    pitched_audio1=pitch(data,sr)
    pitched_noised_audio=noise(pitched_audio1)
    aud4=extract_features(pitched_noised_audio)
    audio=np.vstack((audio,aud4))

    return audio

import multiprocessing as mp
print("Number of processors: ", mp.cpu_count())
```

```
Number of processors:  2
```

```
from joblib import Parallel, delayed
import timeit
start = timeit.default_timer()

def process_feature(path, emotion):
    features = get_features(path)
    X = []
    Y = []
    for ele in features:
        X.append(ele)
        Y.append(emotion)
    return X, Y

paths = data_1.Path
emotions = data_1.Emotions


results = Parallel(n_jobs=-1)(delayed(process_feature)(path, emotion)
for (path, emotion) in zip(paths, emotions))

X = []
```

```
Y = []
for result in results:
    x, y = result
    X.extend(x)
    Y.extend(y)


stop = timeit.default_timer()

print('Time: ', stop - start)
```

```
Time:  541.104870043
```

```
len(X), len(Y), data_1.Path.shape
```

```
(5760, 5760, (1440,))
```

```
emotions = pd.DataFrame(X)
emotions['Emotions'] = Y
emotions.to_csv('emotion.csv', index=False)
emotions.head()
```

```
          0         1         2         3         4         5
6  \
0  0.338379  0.467773  0.537109  0.395508  0.198242  0.148438
0.096191
1  0.255371  0.379395  0.503906  0.493164  0.494141  0.500977
0.509766
2  0.107910  0.132812  0.149902  0.078125  0.041992  0.037598
0.034180
3  0.247559  0.358887  0.479004  0.479980  0.478516  0.491211
0.496582
4  0.136230  0.170410  0.197266  0.115234  0.108887  0.077148
0.078125

          7         8         9  ...      2367      2368      2369
2370  \
0  0.145996  0.169434  0.134766  ...  6.049611  6.316730  6.572581
6.700691
1  0.514648  0.518066  0.513672  ... -2.960067 -4.683760 -2.000774 -
0.628860
2  0.038086  0.044922  0.044434  ...  6.358764  6.698202  6.968064
7.069247
3  0.496582  0.505859  0.507324  ...  5.280717  1.664261  0.182290
4.184744
4  0.118164  0.175293  0.217773  ...  2.674970 -0.359755  2.863761
6.995654

       2371      2372      2373      2374      2375  Emotions
0  6.629885  6.335938  5.834177  5.166978  4.390205   neutral
1  1.281268  1.641462  1.121243  3.666843 -2.836057   neutral
```

```
2  6.931980  6.524342  5.855868  4.976034  3.968057   neutral
3 -3.835645 -1.157363 -2.632855 -0.202608 -0.869386   neutral
4  6.144086  0.868822 -1.191844  5.002143  7.147192      calm

[5 rows x 2377 columns]
```

```
Emotions = pd.read_csv('emotion.csv')
Emotions.head()
```

```
          0         1         2         3         4         5
6  \
0  0.338379  0.467773  0.537109  0.395508  0.198242  0.148438
0.096191
1  0.255371  0.379395  0.503906  0.493164  0.494141  0.500977
0.509766
2  0.107910  0.132812  0.149902  0.078125  0.041992  0.037598
0.034180
3  0.247559  0.358887  0.479004  0.479980  0.478516  0.491211
0.496582
4  0.136230  0.170410  0.197266  0.115234  0.108887  0.077148
0.078125

          7         8         9  ...      2367      2368      2369
2370  \
0  0.145996  0.169434  0.134766  ...  6.049611  6.316730  6.572581
6.700691
1  0.514648  0.518066  0.513672  ... -2.960067 -4.683760 -2.000774 -
0.628860
2  0.038086  0.044922  0.044434  ...  6.358764  6.698202  6.968064
7.069247
3  0.496582  0.505859  0.507324  ...  5.280717  1.664261  0.182290
4.184744
4  0.118164  0.175293  0.217773  ...  2.674970 -0.359755  2.863761
6.995654

       2371      2372      2373      2374      2375  Emotions
0  6.629885  6.335938  5.834177  5.166978  4.390205   neutral
1  1.281268  1.641462  1.121243  3.666843 -2.836057   neutral
2  6.931980  6.524342  5.855868  4.976034  3.968057   neutral
3 -3.835645 -1.157363 -2.632855 -0.202608 -0.869386   neutral
4  6.144086  0.868822 -1.191844  5.002143  7.147192      calm

[5 rows x 2377 columns]
```

```
print(Emotions.isna().any())
```

```
0           False
1           False
2           False
3           False
4           False
```

```
            ...
2372         True
2373         True
2374         True
2375         True
Emotions    False
Length: 2377, dtype: bool
```

```
Emotions=Emotions.fillna(0)
print(Emotions.isna().any())
Emotions.shape
```

```
0           False
1           False
2           False
3           False
4           False
            ...
2372        False
2373        False
2374        False
2375        False
Emotions    False
Length: 2377, dtype: bool
```

```
(5760, 2377)
```

```
np.sum(Emotions.isna())
```

```
0           0
1           0
2           0
3           0
4           0
           ..
2372        0
2373        0
2374        0
2375        0
Emotions    0
Length: 2377, dtype: int64
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

```
sequence_data = Emotions['2375'].values
target_data = Emotions['Emotions'].values

encoder = OneHotEncoder(sparse=False)
target_data = encoder.fit_transform(target_data.reshape(-1, 1))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/
_encoders.py:868: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(
```

```
X_train, X_test, y_train, y_test = train_test_split(sequence_data,
target_data, test_size=0.2, random_state=42)

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train.reshape(-1, 1))
X_test = scaler.transform(X_test.reshape(-1, 1))

model = Sequential()
model.add(LSTM(64, input_shape=(X_train.shape[1], 1),
activation='relu'))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')

history = model.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_test, y_test))
```

```
Epoch 1/100
144/144 [==============================] - 2s 5ms/step - loss: 0.1105
- val_loss: 0.1094
Epoch 2/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 3/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 4/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 5/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 6/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 7/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
```

```
Epoch 8/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 9/100
144/144 [==============================] - 1s 4ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 10/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 11/100
144/144 [==============================] - 1s 4ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 12/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 13/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 14/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 15/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 16/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 17/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 18/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 19/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 20/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 21/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 22/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 23/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 24/100
```

```
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 25/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 26/100
144/144 [==============================] - 1s 4ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 27/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 28/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 29/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 30/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 31/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 32/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 33/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 34/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 35/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 36/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 37/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 38/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 39/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 40/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
```

```
- val_loss: 0.1094
Epoch 41/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 42/100
144/144 [==============================] - 1s 4ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 43/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 44/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 45/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 46/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 47/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 48/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 49/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 50/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 51/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 52/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 53/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 54/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 55/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 56/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
```

```
Epoch 57/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 58/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 59/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 60/100
144/144 [==============================] - 1s 4ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 61/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 62/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 63/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 64/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 65/100
144/144 [==============================] - 1s 4ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 66/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 67/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 68/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 69/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 70/100
144/144 [==============================] - 1s 7ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 71/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 72/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 73/100
```

```
144/144 [==============================] - 1s 4ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 74/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 75/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 76/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 77/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 78/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 79/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 80/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 81/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 82/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 83/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 84/100
144/144 [==============================] - 1s 4ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 85/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 86/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 87/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 88/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 89/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
```

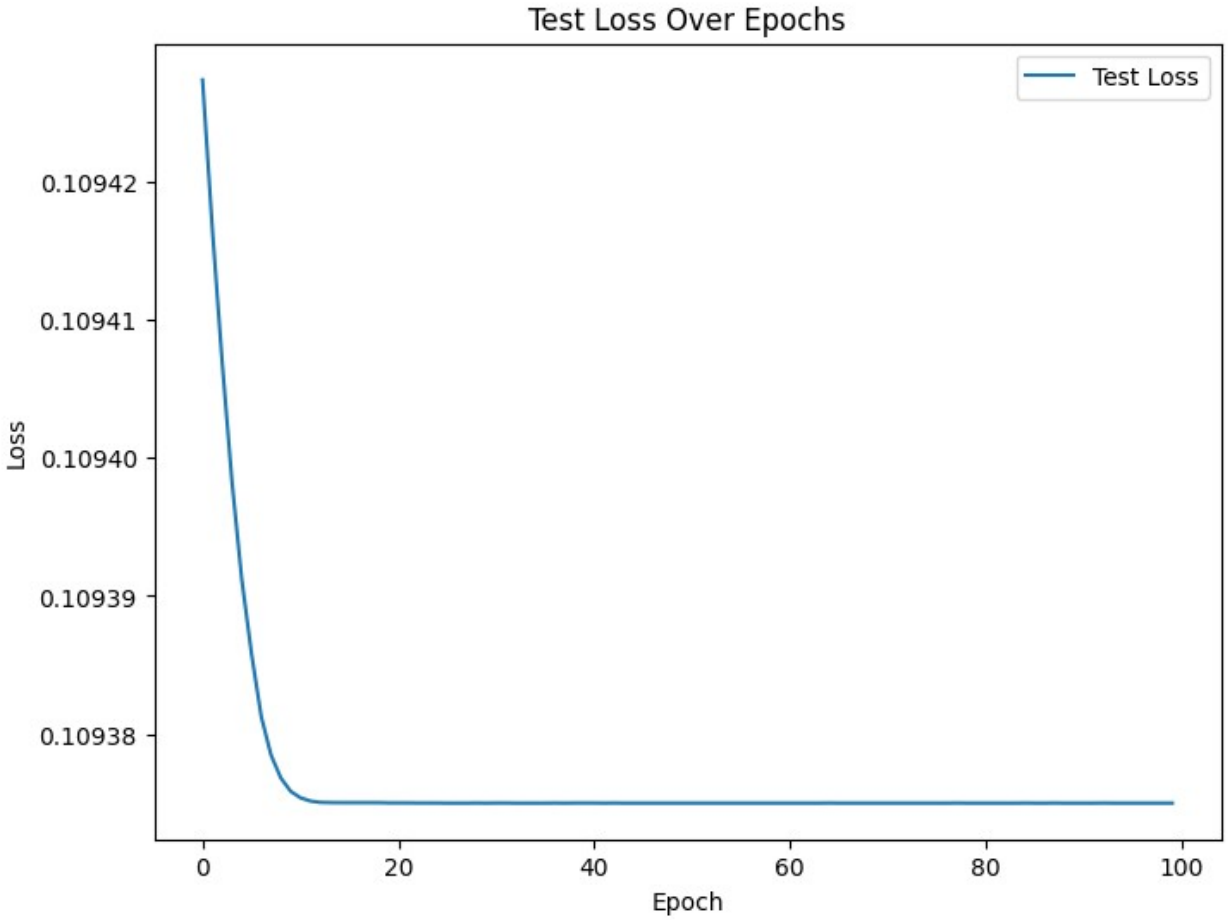
```
- val_loss: 0.1094
Epoch 90/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 91/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 92/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 93/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 94/100
144/144 [==============================] - 0s 3ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 95/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 96/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 97/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 98/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 99/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
Epoch 100/100
144/144 [==============================] - 1s 5ms/step - loss: 0.1094
- val_loss: 0.1094
```

```
test_loss = model.evaluate(X_test, y_test)
print(f'Test loss: {test_loss}')
test_accuracy = model.evaluate(X_test, y_test)
print(f'Test accuracy: {test_accuracy}')
```

```
36/36 [==============================] - 0s 2ms/step - loss: 0.1094
Test loss: 0.109375
36/36 [==============================] - 0s 2ms/step - loss: 0.1094
Test accuracy: 0.109375
```

```
plt.figure(figsize=(8, 6))
plt.plot(history.history['val_loss'], label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

```
plt.title('Test Loss Over Epochs')
plt.show()
```




```
new_dataframe = np.array([0.5])
scaled_dataframe = scaler.transform(new_dataframe.reshape(-1, 1))
prediction = model.predict(scaled_dataframe)
print(f'Predicted probabilities: {prediction}')
```

```
1/1 [==============================] - 0s 231ms/step
Predicted probabilities: [[0.12498863]]
```

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense

movie_recommendations = {
    "angry": ["The Dark Knight", "Fight Club", "Mad Max: Fury Road"],
    "happy": ["Toy Story", "The LEGO Movie", "Up"],
    "sad": ["The Shawshank Redemption", "Titanic", "The Green Mile"],
```

```
    "excited": ["Avengers: Endgame", "Jurassic Park", "Inception"],
    "neutral": ["Forrest Gump", "Pirates of the Caribbean", "Finding
Nemo"],
}


user_emotion = input("Enter your current emotion (e.g., angry, happy,
sad, excited, neutral): ").lower()


if user_emotion in movie_recommendations:
    recommended_movies = movie_recommendations[user_emotion]
    print(f"Recommended movies for {user_emotion} emotion:")
    for i, movie in enumerate(recommended_movies, start=1):
        print(f"{i}. {movie}")
else:
    print("Sorry, we don't have recommendations for that emotion.")
```

```
Enter your current emotion (e.g., angry, happy, sad, excited,
neutral): neutral
Recommended movies for neutral emotion:
1. Forrest Gump
2. Pirates of the Caribbean
3. Finding Nemo
```