**EMOTION RECOGNITION**

**Project:**

**Code:**

#install kaggle

!pip install -q kaggle

#create a kaggle folder

! mkdir ~/.kaggle/

!kaggle datasets download -d ejlok1/toronto-emotional-speech-set-tess

!unzip toronto-emotional-speech-set-tess.zip

import pandas as pd

import numpy as np

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')

from keras import utils

paths =[]

labels = []

for dirname, \_, filenames in os.walk('/content/tess toronto emotional speech set data'):

    for filename in filenames:

        paths.append(os.path.join(dirname, filename))

        label = filename.split('\_')[-1]

        label = label.split('.')[0]

        labels.append(label.lower())

    if len(paths) == 2800:

        break

print('Dataset is loaded')

len(paths)

paths[:5]

labels[:5]

## Create a dataframe

df = pd.DataFrame()

df['speech'] = paths

df['label'] = labels

df.head()

df['label'].value\_counts()

df['label\_count']= df['label'].value\_counts()

df.drop('label\_count', axis = 1)

df.info()

sns.countplot(data=df, x='label')

df

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=(11,4))

    plt.title(emotion, size=20)

    librosa.display.specshow(xdb, sr=sr, x\_axis='time', y\_axis='hz')

    plt.colorbar()

df.info()

df.drop('label\_count', axis=1, inplace = True)

print(df.head())

print(df['label'].unique())

df['speech'].unique()

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)

emotion = 'angry'

path = np.array(df['speech'][df['label']==emotion])[1]

data, sampling\_rate = librosa.load(path)

waveplot(data, sampling\_rate, emotion)

spectogram(data, sampling\_rate, emotion)

Audio(path)

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)

Audio(path)

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)

Audio(path)

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)

Audio(path)

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)

Audio(path)

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)

Audio(path)

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])

X\_mfcc = df['speech'].apply(lambda x: extract\_mfcc(x))

X\_mfcc

X = [x for x in X\_mfcc]

X = np.array(X)

X.shape

## input split

X = np.expand\_dims(X, -1)

X.shape

from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder()

y = enc.fit\_transform(df[['label']])

y = y.toarray()

y.shape

from sklearn.model\_selection import train\_test\_split

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Create the LSTM Model

from keras.models import Sequential

from keras.layers import Dense, LSTM, Dropout

model = Sequential([

    LSTM(256, return\_sequences=False, input\_shape=(40, 1)),

    Dropout(0.5),  # Add dropout after LSTM layer

    Dense(128, activation='relu'),

    Dropout(0.5),  # Add dropout after dense layer

    Dense(64, activation='relu'),

    Dropout(0.5),  # Add dropout after dense layer

    Dense(7, activation='softmax')

])

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

Model: "sequential"

#Train the model

history = model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=30, batch\_size=64)

Plot the results

epochs = list(range(30))

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()

y\_pred = model.predict(X\_val)

y\_pred\_classes = np.argmax(y\_pred, axis=1)

y\_val\_classes = np.argmax(y\_val, axis=1)

from sklearn.metrics import confusion\_matrix, classification\_report

# Compute confusion matrix

conf\_matrix = confusion\_matrix(y\_val\_classes, y\_pred\_classes)

# Print the confusion matrix

print("Confusion Matrix:")

print(conf\_matrix)

#Print the classification report

target\_names = ['angry', 'disgust', 'fear', 'happy', 'neutral', 'ps', 'sad']

print("Classification Report:")

print(classification\_report(y\_val\_classes, y\_pred\_classes, target\_names=target\_names))

#Correlation HeatMap

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=target\_names, yticklabels=target\_names)

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()