**Prototyping documentation for the Emotion Detection**

Emotion Detection is the process of predicting human emotions from audio signals using artificial intelligence (AI) techniques.

To perform this first we need to know about its approach and its preprocessing functionalities.

Step by step approach:

* We use Librosa library to extract the audio features such as MFCCs.
* We train the basic classifier using scikit-learn.
* First we need to import some libraries such as

import os

import librosa

import librosa.display

import numpy as mp

import pandas as pd

1. **Inputting audio files**:

* First create a directory to organize a audio files and that directory structure should have a sub directories for “happy” and “sad”.
* Ensure that audio files are in one of the accepted form, which includes:

WAV(.wav) , MP3(.mp3), FLAC(.flac) etc

1. **Running the emotion detection in system**:

* Open the command prompt
* Install the required libraray pip install librosa scikit-learn
* Organize the dataset by creating the directory structure where we have subdirectories for “happy” and “sad”.
* To get ouput based on happy or sad we need to draw a grap. For drawing graph we require pandas and matplotlib to create the line charts to visualize the predicted emotions.

**Libraries Used**:

The main library that we will be using is Librosa. Apart from that we will also be using Soundfile and Pyaudio .

* Emotion Prediction from Audio:
* First we need to train or use the pre-defined model for audio emotion recognition.
* Common libraries for this includes TessorFlow, PyTorch or some specialized libraries like librosa for audio processing.
* The output of this model would predict emotion for each audio file.
* Create a data table
* Use the Pandas library to create a data table with two columns:

“Audio File Name” and “Predicted Emotion” .

* Populate this table with the file names and their respective predicted emotions.
* Smooth Line Chart :
* Use the matplotlib library to create a smooth line chart. To create a chart where the positive y-axis represents “happy” and the negative y-axis represents “sad”, we need to map our predicated emotions to numerical values.
* Loading Audio dataset
* By using load data-set function we can download and prepare a dataset.
* To get the current directory that we are using just give os.cwd() , it shows our current working directory.
* To see the list of directories we give os.listdir() .
* After getting current directory now we have to give a path and assign any dataset in directory to file.
* By using ospath.join() , join both file name and new path.
* Now, to read datasets we use pandas we give pd.read\_csv()
* Now we got the first dataset to travel inside that data set again we are going to do same method as we did before,

path= getos.cwd()

file\_name = “ data\_set”

new\_path = os.path.join(path, file\_name)

os.chdir(new\_path)

* By giving os.listdir() we get all the audio files within the directory.
* And to get any specific file we give indexing

Ex: os.listdir()[0]

* Now to convert this audio into numpy array we use

y, sr = librosa.load(os.listdir()[0], duration = 1.2)

ps = librosa.feature.melspectogram(y=y, sr=sr)

**Here in this Emotion detection model ,I have downloaded the audio files from envato elements, and loaded thoseaudio files into this model.**

**CODE for emotion detection**:

import os

import librosa

import librosa.display

import numpy as np

import pandas as pd

from sklearn.svm import SVC

import matplotlib.pyplot as plt

# Define a function to extract audio features

def extract\_features(audio\_path):

y, sr = librosa.load(audio\_path)

# Extract relevant features (e.g., MFCCs, pitch)

mfccs = librosa.feature.mfcc(y=y, sr=sr, n\_mfcc=13)

pitch, \_ = librosa.piptrack(y=y, sr=sr)

# Aggregate features

features = np.mean(mfccs, axis=1)

features = np.append(features, np.mean(pitch))

return features

# Create a list to store features and corresponding labels

data = []

labels = []

# Process each audio file in the dataset

for emotion in ["happy", "sad"]:

emotion\_dir = os.path.join("mixkit-ending-show-audience-clapping-478” ,”mixkit-lost-kid-sobbing-474” ,emotion)

for audio\_file in os.listdir(emotion\_dir):

audio\_path = os.path.join(emotion\_dir, audio\_file)

features = extract\_features(audio\_path)

data.append(features)

labels.append(emotion)

# Convert lists to numpy arrays

X = np.array(data)

y = np.array(labels)

# Train a simple classifier (SVM)

classifier = SVC(kernel='linear')

classifier.fit(X, y)

# Create a DataFrame to store results

results = pd.DataFrame(columns=["Audio File", "Predicted Emotion"])

# Predict emotion for new audio files and store in the DataFrame

new\_data = []

for audio\_file in os.listdir("new\_audio"):

audio\_path = os.path.join("new\_audio", audio\_file)

features = extract\_features(audio\_path)

new\_data.append(features)

predicted\_emotion = classifier.predict([features])[0]

results = results.append({"Audio File": , "Predicted Emotion": predicted\_emotion}, ignore\_index=True)

# Create a line chart

positive = (results["Predicted Emotion"] == "happy").astype(int)

negative = (results["Predicted Emotion"] == "sad").astype(int)

plt.plot(results["Audio File"], positive, label="Happy", marker='o', color='g')

plt.plot(results["Audio File"], -negative, label="Sad", marker='o', color='r')

plt.xlabel("Audio File")

plt.ylabel("Emotion")

plt.title("Emotion Prediction")

plt.xticks(rotation=45)

plt.grid(True)

plt.legend()

plt.show()

# Print the results as a table

print(results)