Pepito, Alyssa Mae & Soleño, Keziah Antonette 06-21-2022

Group 7

CS171 | BM1

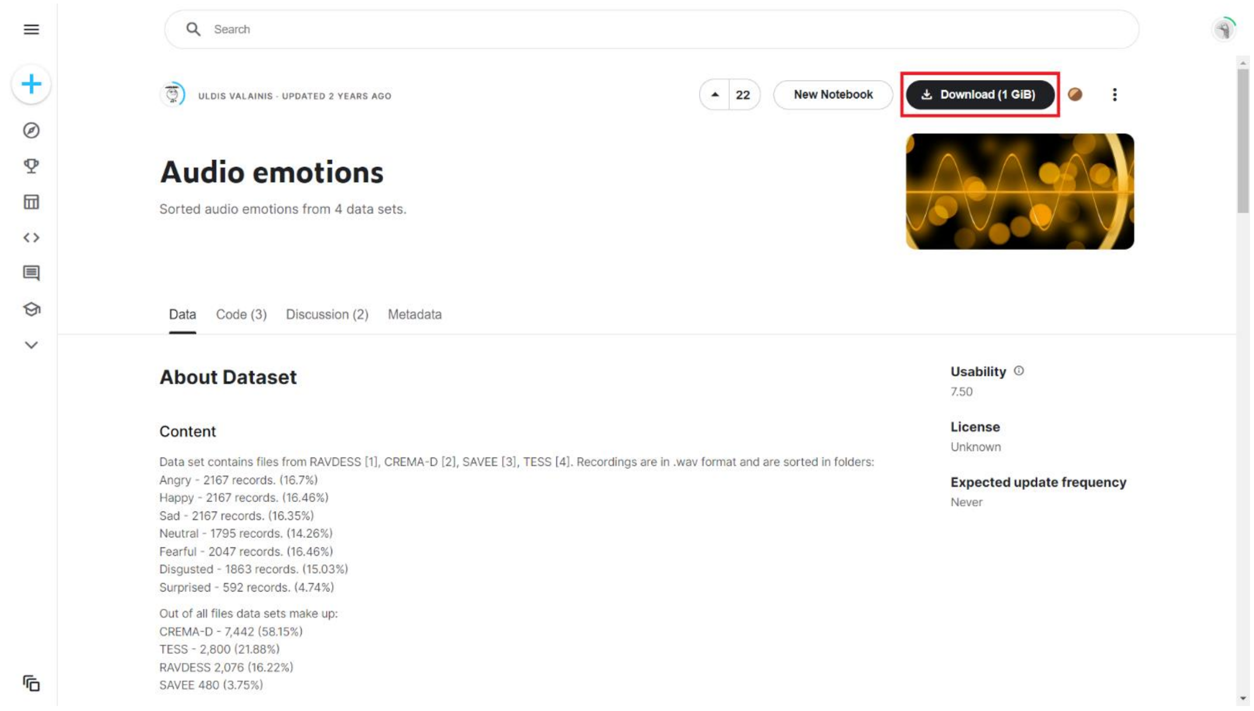
Module 1: Summative Assessment (Documentation)

I. Background and Introduction

Speech Emotion Recognition (SER) is a collection of methodologies that process and detect emotions embedded within speech. SER classifies signals from human speech and uses those signals to identify the underlying emotion within them. The SER system is especially useful in fields where interactive based-assistant or caller-agent conversation analysis is required. Recently, such a system has been primarily utilized in the medical field whenever a patient is being diagnosed over mobile platforms, customer service as conversations between the customer and attendants are analyzed to improve quality of service, and recommender systems as customers' emotions are used to determine which products are most appropriate to recommend to them. (Market Trends, 2020) (Rockikz, 2019)

II. Data Preparation

Since we have selected emotion recognition through voice (speech emotion recognition), we downloaded the provided data set found in this link: https://www.kaggle.com/datasets/uldisvalainis/audio-emotions?resource=download. The data set contained files from RAVDESS, CREMA-D, SAVEE, and TESS, and .wav recordings sorted into seven (angry, happy, sad, neutral, fearful, disgusted, and surprised) emotions. (*Audio Emotions*, 2020)



Audio emotions

Sorted audio emotions from 4 data sets.

About Dataset

Content

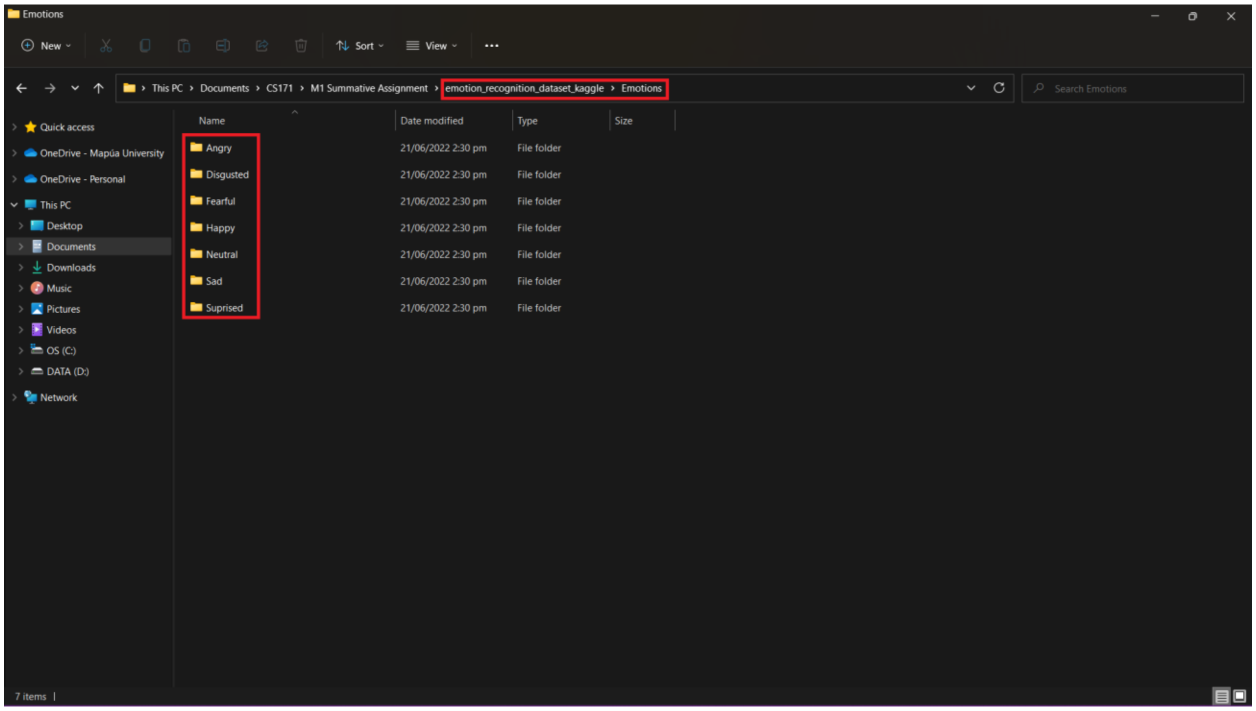
Data set contains files from RAVDESS [1], CREMA-D [2], SAVEE [3], TESS [4]. Recordings are in .wav format and are sorted in folders:
Angry - 2167 records. (16.7%)
Happy - 2167 records. (16.46%)
Sad - 2167 records. (16.35%)
Neutral - 1795 records. (14.26%)
Fearful - 2047 records. (16.46%)
Disgusted - 1863 records. (15.03%)
Surprised - 592 records. (4.74%)

Out of all files data sets make up:
CREMA-D - 7,442 (58.15%)
TESS - 2,800 (21.88%)
RAVDESS 2,076 (16.22%)
SAVEE 480 (3.75%)

Usability
7.50

License
Unknown

Expected update frequency
Never



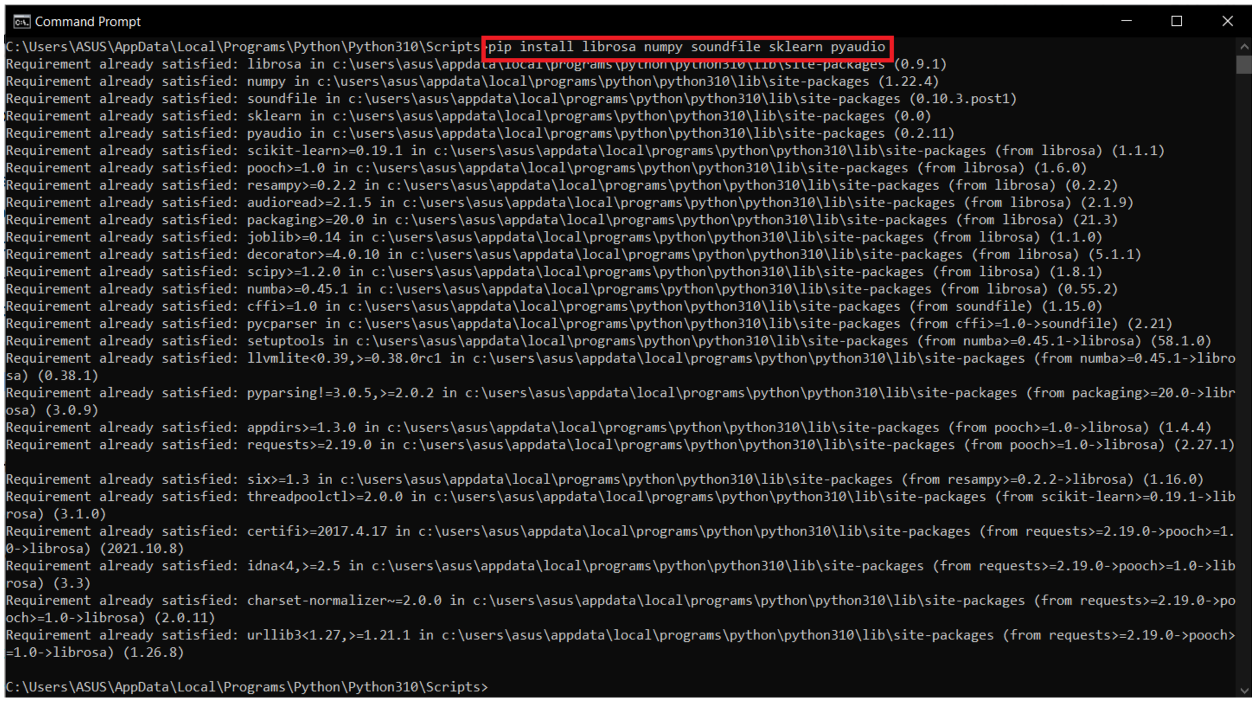
This PC › Documents › CS171 › M1 Summative Assignment › emotion_recognition_dataset_kaggle › Emotions

Name	Date modified	Type	Size
Angry	21/06/2022 2:30 pm	File folder	
Disgusted	21/06/2022 2:30 pm	File folder	
Fearful	21/06/2022 2:30 pm	File folder	
Happy	21/06/2022 2:30 pm	File folder	
Neutral	21/06/2022 2:30 pm	File folder	
Sad	21/06/2022 2:30 pm	File folder	
Suprised	21/06/2022 2:30 pm	File folder	

7 items

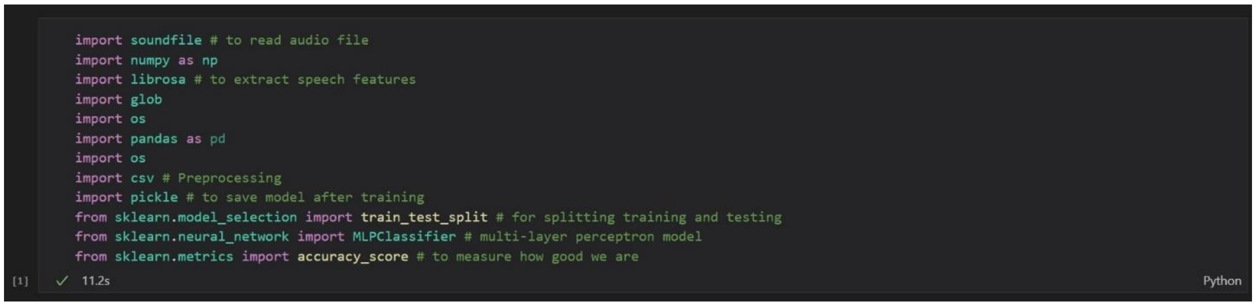
III. Python Libraries Installation

Before performing the necessary codes for speech emotion recognition, the Librosa Python library and its dependencies had to be installed by entering the following command in the terminal:



```
C:\Users\ASUS\AppData\Local\Programs\Python\Python310\Scripts>pip install librosa numpy soundfile sklearn pyaudio
Requirement already satisfied: librosa in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (0.9.1)
Requirement already satisfied: numpy in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (1.22.4)
Requirement already satisfied: soundfile in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (0.10.3.post1)
Requirement already satisfied: sklearn in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (0.0)
Requirement already satisfied: pyaudio in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (0.2.11)
Requirement already satisfied: scikit-learn>=0.19.1 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from librosa) (1.1.1)
Requirement already satisfied: pooch>=1.0 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from librosa) (1.6.0)
Requirement already satisfied: resampy>=0.2.2 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from librosa) (0.2.2)
Requirement already satisfied: audioread>=2.1.5 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from librosa) (2.1.9)
Requirement already satisfied: packaging>=20.0 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from librosa) (21.3)
Requirement already satisfied: joblib>=0.14 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from librosa) (1.1.0)
Requirement already satisfied: decorator>=4.0.10 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from librosa) (5.1.1)
Requirement already satisfied: scipy>=1.2.0 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from librosa) (1.8.1)
Requirement already satisfied: numba>=0.45.1 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from librosa) (0.55.2)
Requirement already satisfied: cffi>=1.0 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from soundfile) (1.15.0)
Requirement already satisfied: pycparser in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from cffi>=1.0->soundfile) (2.21)
Requirement already satisfied: setuptools in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from numba>=0.45.1->librosa) (58.1.0)
Requirement already satisfied: llvmlite<0.39,>=0.38.0rc1 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from numba>=0.45.1->libro
sa) (0.38.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from packaging>=20.0->libr
osa) (3.0.9)
Requirement already satisfied: appdirs>=1.3.0 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from pooch>=1.0->librosa) (1.4.4)
Requirement already satisfied: requests>=2.19.0 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from pooch>=1.0->librosa) (2.27.1)

Requirement already satisfied: six>=1.3 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from resampy>=0.2.2->librosa) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn>=0.19.1->lib
rosa) (3.1.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from requests>=2.19.0->pooch>=1.
0->librosa) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from requests>=2.19.0->pooch>=1.0->lib
rosa) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from requests>=2.19.0->po
och>=1.0->librosa) (2.0.11)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\asus\appdata\local\programs\python\python310\lib\site-packages (from requests>=2.19.0->pooch>
=1.0->librosa) (1.26.8)

C:\Users\ASUS\AppData\Local\Programs\Python\Python310\Scripts>
```

Once all the libraries and their dependencies have been installed, these libraries were then imported by typing the following lines of code:



```
import soundfile # to read audio file
import numpy as np
import librosa # to extract speech features
import glob
import os
import pandas as pd
import os
import csv # Preprocessing
import pickle # to save model after training
from sklearn.model_selection import train_test_split # for splitting training and testing
from sklearn.neural_network import MLPClassifier # multi-layer perceptron model
from sklearn.metrics import accuracy_score # to measure how good we are
[1]  ✓  11.2s                                                              Python
```

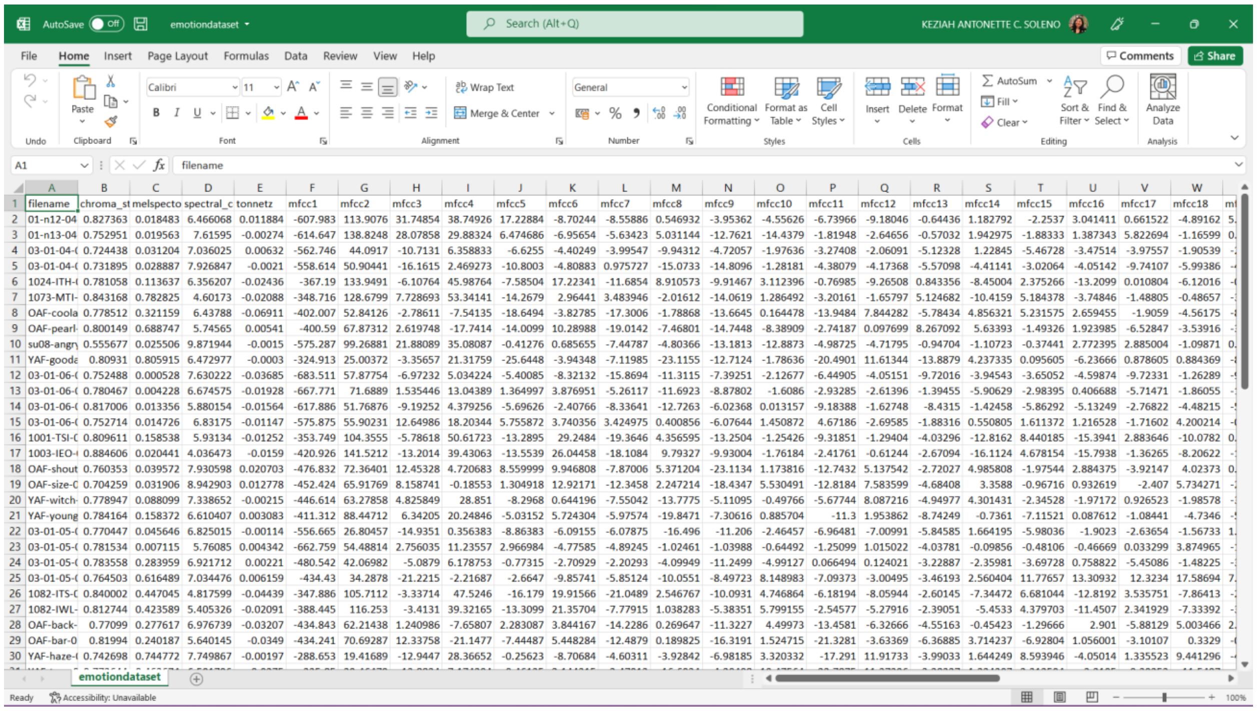
IV. Speech Feature Extraction

Now that the data set, libraries, and dependencies have been downloaded and installed, feature extraction may finally be performed. To prepare the audio files, the kwargs function had to be created to handle the features being extracted. The following are the features supported by kwargs: MFCC, Chroma, MEL Spectogram Frequency, Contrast, and Tonnetz.

```
def extract_feature(file_name, **kwargs):

    mfcc = kwargs.get("mfcc")
    chroma = kwargs.get("chroma")
    mel = kwargs.get("mel")
    contrast = kwargs.get("contrast")
    tonnetz = kwargs.get("tonnetz")
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate = sound_file.samplerate
        if chroma or contrast:
            stft = np.abs(librosa.stft(X))
        result = np.array([])
        if mfcc:
            mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate,
n_mfcc=40).T, axis=0)
            result = np.hstack((result, mfccs))
        if chroma:
            chroma = np.mean(librosa.feature.chroma_stft(S=stft,
sr=sample_rate).T,axis=0)
            result = np.hstack((result, chroma))
        if mel:
            mel = np.mean(librosa.feature.melspectrogram(X,
sr=sample_rate).T,axis=0)
            result = np.hstack((result, mel))
        if contrast:
            contrast = np.mean(librosa.feature.spectral_contrast(S=stft,
sr=sample_rate).T,axis=0)
            result = np.hstack((result, contrast))
        if tonnetz:
            tonnetz =
np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X),
sr=sample_rate).T,axis=0)
            result = np.hstack((result, tonnetz))
    return result
```

After applying these features to the .wav files, the speech waveform of each audio will change to a form of parametric representation at a lesser data rate that can be later converted into a .csv file. (Rockikz, 2019)

```
# Saving the datset into a CSV file

header = 'filename chroma_stft melspectogram spectral_contrast tonnetz'
for i in range(1, 41):
    header += f' mfcc{i}'
header += ' label'
header = header.split()

file = open('emotiondataset.csv', 'w', newline='')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
emotions = 'Angry Disgusted Fearful Happy Neutral Sad Surprised'.split()

for m in emotions:
    for filename in os.listdir(f'Emotions/{m}'):
        audioname = f'Emotions/{m}/{filename}'
        y, sr = librosa.load(audioname, mono=True, duration=30)
        stft = np.abs(librosa.stft(y))
        chroma_stft = librosa.feature.chroma_stft(y=stft, sr=sr)
        mel_spec = librosa.feature.melspectrogram(y=y, sr=sr)
        spec_cont = librosa.feature.spectral_contrast(y=stft, sr=sr)
        tonnetz = librosa.feature.tonnetz(y=librosa.effects.harmonic(y), sr=sr)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
        to_append = f'{filename} {np.mean(chroma_stft)} {np.mean(mel_spec)}
{np.mean(spec_cont)} {np.mean(tonnetz)}'
        for e in mfcc:
            to_append += f' {np.mean(e)}'
        to_append += f' {m}'
        file = open('emotiondataset.csv', 'a', newline='')
        with file:
            writer = csv.writer(file)
            writer.writerow(to_append.split())
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	filename	chroma_st	melspecto	spectral_c	tonnetz	mfcc1	mfcc2	mfcc3	mfcc4	mfcc5	mfcc6	mfcc7	mfcc8	mfcc9	mfcc10	mfcc11	mfcc12	mfcc13	mfcc14	mfcc15	mfcc16	mfcc17	mfcc18
2	01-n12-04	0.827363	0.018483	6.466068	0.011884	-607.983	113.9076	31.74854	38.74926	17.22884	-8.70244	-8.55886	0.546932	-3.95362	-4.55626	-6.73966	-9.18046	-0.64436	1.182792	-2.2537	3.041411	0.661522	-4.89162
3	01-n13-04	0.752951	0.019563	7.61595	-0.00274	-614.647	138.8248	28.07858	29.88324	6.474686	-6.95654	-5.63423	5.031144	-12.7621	-14.4379	-1.81948	-2.64656	-0.57032	1.942975	-1.88333	1.387343	5.822694	-1.16599
4	03-01-04-(	0.724438	0.031204	7.036025	0.00632	-562.746	44.0917	-10.7131	6.358833	-6.6255	-4.40249	-3.99547	-9.94312	-4.72057	-1.97636	-3.27408	-2.06091	-5.12328	1.22845	-5.46728	-3.47514	-3.97557	-1.90539
5	03-01-04-(	0.731895	0.028887	7.926847	-0.0021	-558.614	50.90441	-16.1615	2.469273	-10.8003	-4.80883	0.975727	-15.0733	-14.8096	-1.28181	-4.38079	-4.17368	-5.57098	-4.41141	-3.02064	-4.05142	-9.74107	-5.99386
6	1024-ITH-(	0.781058	0.113637	6.356207	-0.02436	-367.19	133.9491	-6.10764	45.98764	-7.58504	17.22341	-11.6854	8.910573	-9.91467	3.112396	-0.76985	-9.26508	0.843356	-8.45004	2.375266	-13.2099	0.010804	-6.12016
7	1073-MTI-	0.843168	0.782825	4.60173	-0.02088	-348.716	128.6799	7.728693	53.34141	-14.2679	2.96441	3.483946	-2.01612	-14.0619	1.286492	-3.20161	-1.65797	5.124682	-10.4159	5.184378	-3.74846	-1.48805	-0.48657
8	OAF-coola	0.778512	0.321159	6.43788	-0.06911	-402.007	52.84126	-2.78611	-7.54135	-18.6494	-3.82785	-17.3006	-1.78868	-13.6645	0.164478	-13.9484	7.844282	-5.78434	4.856321	5.231575	2.659455	-1.9059	-4.56175
9	OAF-pearl-	0.800149	0.688747	5.74565	0.00541	-400.59	67.87312	2.619748	-17.7414	-14.0099	10.28988	-19.0142	-7.46801	-14.7448	-8.38909	-2.74187	0.097699	8.267092	5.63393	-1.49326	1.923985	-6.52847	-3.53916
10	su08-angr	0.555677	0.025506	9.871944	-0.0015	-575.287	99.26881	21.88089	35.08087	-0.41276	0.685655	-7.44787	-4.80366	-13.1813	-12.8873	-4.98725	-4.71795	-0.94704	-1.10723	-0.37441	2.772395	2.885004	-1.09871
11	YAF-gooda	0.80931	0.805915	6.472977	-0.0003	-324.913	25.00372	-3.35657	21.31759	-25.6448	-3.94348	-7.11985	-23.1155	-12.7124	-1.78636	-20.4901	11.61344	-13.8879	4.237335	0.095605	-6.23666	0.878605	0.884369
12	03-01-06-(	0.752488	0.000528	7.630222	-0.03685	-683.511	57.87754	-6.97232	5.034224	-5.40085	-8.32132	-15.8694	-11.3115	-7.39251	-2.12677	-6.44905	-4.05151	-9.72016	-3.94543	-3.65052	-4.59874	-9.72331	-1.26289
13	03-01-06-(	0.780467	0.004228	6.674575	-0.01928	-667.771	71.6889	1.535446	13.04389	1.364997	3.876951	-5.26117	-11.6923	-8.87802	-1.6086	-2.93285	-2.61396	-1.39455	-5.90629	-2.98395	0.406688	-5.71471	-1.86055
14	03-01-06-(	0.817006	0.013356	5.880154	-0.01564	-617.886	51.76876	-9.19252	4.379256	-5.69626	-2.40766	-8.33641	-12.7263	-6.02368	0.013157	-9.18388	-1.62748	-8.4315	-1.42458	-5.86292	-5.13249	-2.76822	-4.48215
15	03-01-06-(	0.752714	0.014726	6.83175	-0.01147	-575.875	55.90231	12.64986	18.20344	5.755872	3.740356	3.424975	0.400856	-6.07644	1.450872	4.67186	-2.69585	-1.88316	0.550805	1.611372	1.216528	-1.71602	4.200214
16	1001-TSI-(	0.809611	0.158538	5.93134	-0.01252	-353.749	104.3555	-5.78618	50.61723	-13.2895	29.2484	-19.3646	4.356595	-13.2504	-1.25426	-9.31851	-1.29404	-4.03296	-12.8162	8.440185	-15.3941	2.883646	-10.0782
17	1003-IEO-(	0.884606	0.020441	4.036473	-0.0159	-420.926	141.5212	-13.2014	39.43063	-13.5539	26.04458	-18.1084	9.79327	-9.93004	-1.76184	-2.41761	-0.61244	-2.67094	-16.1124	4.678154	-15.7938	-1.36265	-8.20622
18	OAF-shout	0.760353	0.039572	7.930598	0.020703	-476.832	72.36401	12.45328	4.720683	8.559999	9.946808	-7.87006	5.371204	-23.1134	1.173816	-12.7432	5.137542	-2.72027	4.985808	-1.97544	2.884375	-3.92147	4.02373
19	OAF-size-(	0.704259	0.031906	8.942903	0.012778	-452.424	65.91769	8.158741	-0.18553	1.304918	12.92171	-12.3458	2.247214	-18.4347	5.530491	-12.8184	7.583599	-4.68408	3.3588	-0.96716	0.932619	-2.407	5.734271
20	YAF-witch-	0.778947	0.088099	7.338652	-0.00215	-446.614	63.27858	4.825849	28.851	-8.2968	0.644196	-7.55042	-13.7775	-5.11095	-0.49766	-5.67744	8.087216	-4.94977	4.301431	-2.34528	-1.97172	0.926523	-1.98578
21	YAF-young	0.784164	0.158372	6.610407	0.003083	-411.312	88.44712	6.34205	20.24846	-5.03152	5.724304	-5.97574	-19.8471	-7.30616	0.885704	-11.3	1.953862	-8.74249	-0.7361	-7.11521	0.087612	-1.08441	-4.7346
22	03-01-05-(	0.770447	0.045646	6.825015	-0.00114	-556.665	26.80457	-14.9351	0.356383	-8.86383	-6.09155	-6.07875	-16.496	-11.206	-2.46457	-6.96481	-7.00991	-5.84585	1.664195	-5.98036	-1.9023	-2.63654	-1.56733
23	03-01-05-(	0.781534	0.007115	5.76085	0.004342	-662.759	54.48814	2.756035	11.23557	2.966984	-4.77585	-4.89245	-1.02461	-1.03988	-0.64492	-1.25099	1.015022	-4.03781	-0.09856	-0.48106	-0.46669	0.033299	3.874965
24	03-01-05-(	0.783558	0.283959	6.921712	0.00221	-480.542	42.06982	-5.0879	6.178753	-0.77315	-2.70929	-2.20293	-4.09949	-11.2499	-4.99127	0.066494	0.124021	-3.22887	-2.35981	-3.69728	0.758822	-5.45086	-1.48225
25	03-01-05-(	0.764503	0.616489	7.034476	0.006159	-434.43	34.2878	-21.2215	-2.21687	-2.6647	-9.85741	-5.85124	-10.0551	-8.49723	8.148983	-7.09373	-3.00495	-3.46193	2.560404	11.77657	13.30932	12.3234	17.58694
26	1082-ITS-(	0.840002	0.447045	4.817599	-0.04439	-347.886	105.7112	-3.33714	47.5246	-16.179	19.91566	-21.0489	2.546767	-10.0931	4.746864	-6.18194	-8.05944	-2.60145	-7.34472	6.681044	-12.8192	3.535751	-7.86413
27	1082-IWL-	0.812744	0.423589	5.405326	-0.02091	-388.445	116.253	-3.4131	39.32165	-13.3099	21.35704	-7.77915	1.038283	-5.38351	5.799155	-2.54577	-5.27916	-2.39051	-5.4533	4.379703	-11.4507	2.341929	-7.33392
28	OAF-back-	0.77099	0.277617	6.976739	-0.03207	-434.843	62.21438	1.240986	-7.65807	2.283087	3.844167	-14.2286	0.269647	-11.3227	4.49973	-13.4581	-6.32666	-4.55163	-0.45423	-1.29666	2.901	-5.88129	5.003466
29	OAF-bar-0	0.81994	0.240187	5.640145	-0.0349	-434.241	70.69287	12.33758	-21.1477	-7.44487	5.448284	-12.4879	0.189825	-16.3191	1.524715	-21.3281	-3.63369	-6.36885	3.714237	-6.92804	1.056001	-3.10107	0.3329
30	YAF-haze-(	0.742698	0.744772	7.749867	-0.00197	-288.653	19.41689	-12.9447	28.36652	-0.25623	-8.70684	-4.60311	-3.92842	-6.98185	3.320332	-17.291	11.91733	-3.99033	1.644249	8.593946	-4.05014	1.335523	9.441296

V. Read Data from the Kaggle Data Set

After declaring a function for feature extraction, we declared two additional functions that will read an audio file by associating the numbers on the .wav files with a corresponding emotion from the data set.

```
# all emotions on Audio Emotions dataset
int2emotion = {
    "01": "Neutral",
    "02": "Happy",
    "03": "Sad",
    "04": "Angry",
    "05": "Fearful",
    "06": "Disgusted",
    "07": "Surprised"
}

# we allow only these emotions
AVAILABLE_EMOTIONS = {
    "Neutral",
    "Happy",
    "Sad"
}
```

The following code snippet is the main process for extracting features from the data. First, the computer will read the file name and retrieve the emotion label to identify what emotion the .wav file is expressing. If the retrieved file is not in the list of AVAILABLE_EMOTIONS, the process will continue to the next step of having its speech features extracted. These features will then be added to the data, which will be split into a training and testing set.

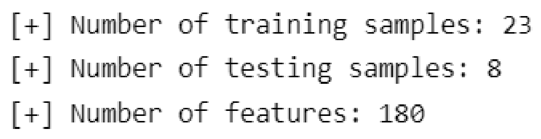
```
def load_data(test_size=0.2):
    X, y = [], []

    for m in emotions:
        for file in glob.glob(f'Emotions/{m}/*.wav'):

            # get the base name of the audio file
            basename = os.path.basename(file)

            # get the emotion label
            emotion = int2emotion[basename.split("-")[2]]

            # we allow only AVAILABLE_EMOTIONS we set
            if emotion not in AVAILABLE_EMOTIONS:
                continue

            # extract speech features
            features = extract_feature(file, mfcc=True, chroma=True, mel=True)

            # add to data
            X.append(features)
            y.append(emotion)

            # split the data to training and testing and return it
            return train_test_split(np.array(X), y, test_size=test_size,
            random_state=7)
```

VI. Load and Log Information about the Data Set in 75:25 Proportions

After splitting the data into a training and testing set, the following code was performed to divide them into a 75:25-percent ratio.

```
# load Audio emotion dataset, 75% training 25% testing
X_train, X_test, y_train, y_test = load_data(test_size=0.25)

# number of samples in training data
print("[+] Number of training samples:", X_train.shape[0])

# number of samples in testing data
print("[+] Number of testing samples:", X_test.shape[0])

# number of features used
# this is a vector of features extracted
# using extract_features() function
print("[+] Number of features:", X_train.shape[1])
```

Output:



```
[+] Number of training samples: 23
[+] Number of testing samples: 8
[+] Number of features: 180
```

VII. Initialize Model with Given Parameters

The `Multi Layer Perceptron` classifier was initialized using the following parameters.

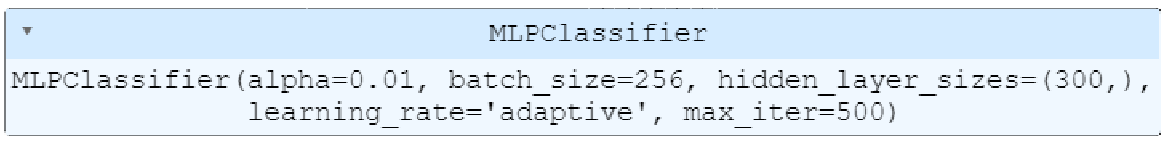
```
# best model, determined by a grid search
model_params = {
    'alpha': 0.01,
    'batch_size': 256,
    'epsilon': 1e-08,
    'hidden_layer_sizes': (300,),
    'learning_rate': 'adaptive',
    'max_iter': 500,
}

# initialize Multi Layer Perceptron classifier
# with the parameters declared above
model = MLPClassifier(**model_params)
```

VIII. Train Model with Loaded Data Set

After loading the data set, we trained the model by applying the codes below.

```
# train the model
print("[*] Training the model...")
model.fit(X_train, y_train)
```

Since MLPClassifier was initialized with the given parameters from the previous step, the output below was generated.



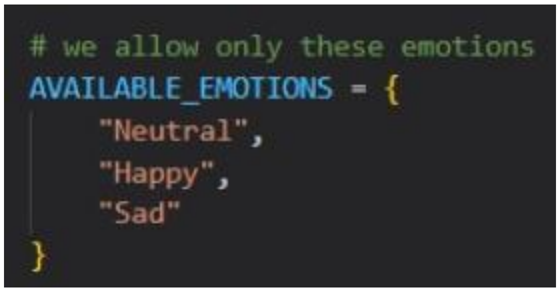
```
▾                         MLPClassifier
MLPClassifier(alpha=0.01, batch_size=256, hidden_layer_sizes=(300,),
              learning_rate='adaptive', max_iter=500)
```

IX. Print Accuracy Score

After training our model with our dataset, we can now calculate the accuracy score and print it to measure how good we are.

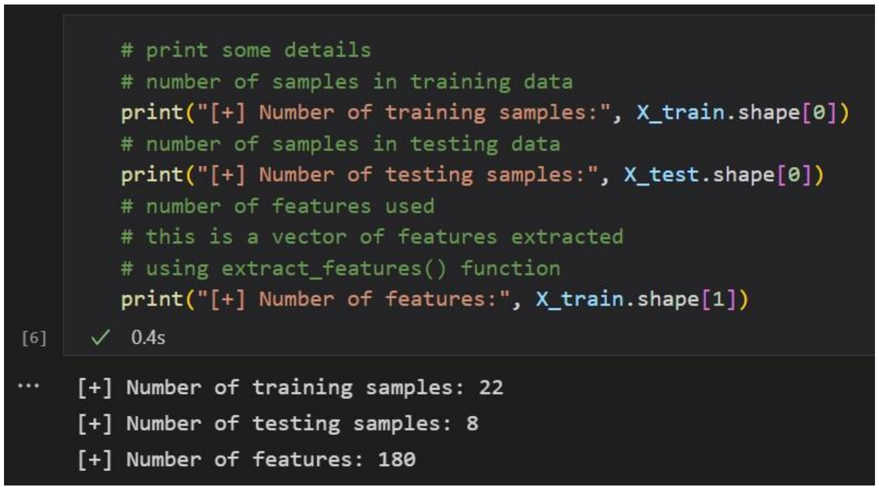
```
# predict 25% of data to measure how good we are
y_pred = model.predict(X_test)

# calculate the accuracy
accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
```

The accuracy score depends on how many emotions we want to classify. For example, let's say we only want to classify neutral, happy, and sad emotions.



```
# we allow only these emotions
AVAILABLE_EMOTIONS = {
    "Neutral",
    "Happy",
    "Sad"
}
```

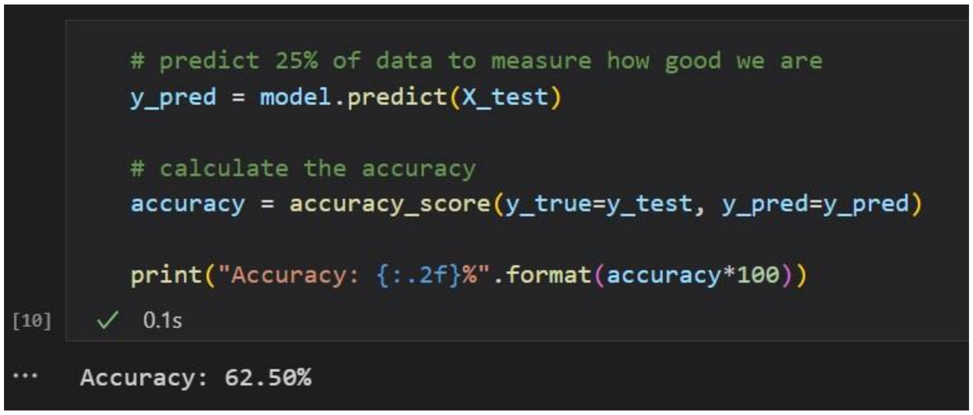
Since we have a total of 10 audio files for each emotion, we have 30 audio files all in all. It is then divided into training and testing samples. We notice that we only have a small sample size to work with.



```
# print some details
# number of samples in training data
print("[+] Number of training samples:", X_train.shape[0])
# number of samples in testing data
print("[+] Number of testing samples:", X_test.shape[0])
# number of features used
# this is a vector of features extracted
# using extract_features() function
print("[+] Number of features:", X_train.shape[1])
```

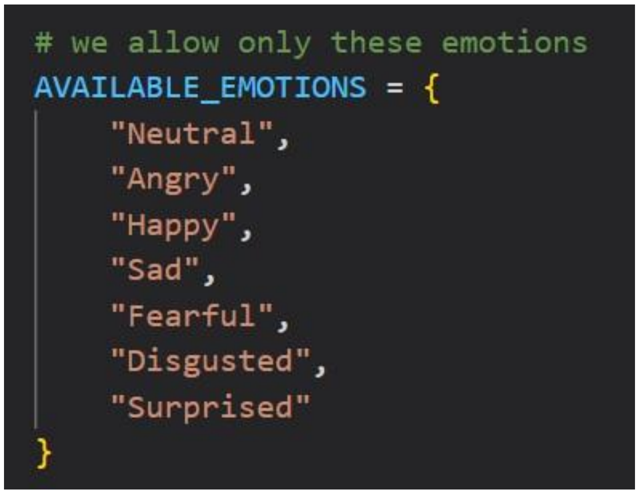
```
[6]  ✓ 0.4s

[+] Number of training samples: 22
[+] Number of testing samples: 8
[+] Number of features: 180
```

However, we get a 62.50% accuracy for classifying neutral, happy, and sad emotions. This is quite a good score since the accuracy rate is more than 50%.



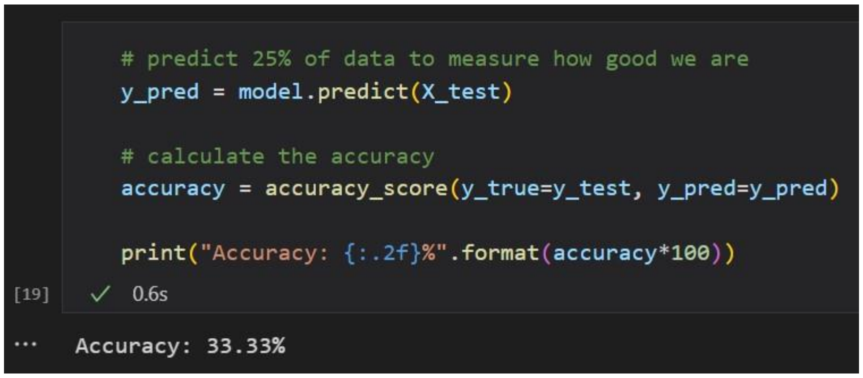
```
# predict 25% of data to measure how good we are
y_pred = model.predict(X_test)

# calculate the accuracy
accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
[10]  ✓ 0.1s

...  Accuracy: 62.50%
```

If we classify all emotions, it will indicate an inaccurate model due to the sample size. We are aware that this could cause a problem in dealing with more than two classes.



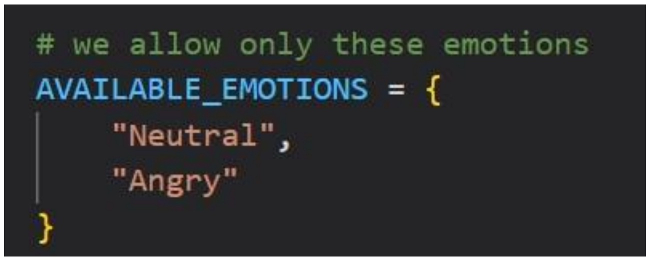
```
# we allow only these emotions
AVAILABLE_EMOTIONS = {
    "Neutral",
    "Angry",
    "Happy",
    "Sad",
    "Fearful",
    "Disgusted",
    "Surprised"
}
```

As we predicted, it becomes more inaccurate if the model tries to classify more than two.



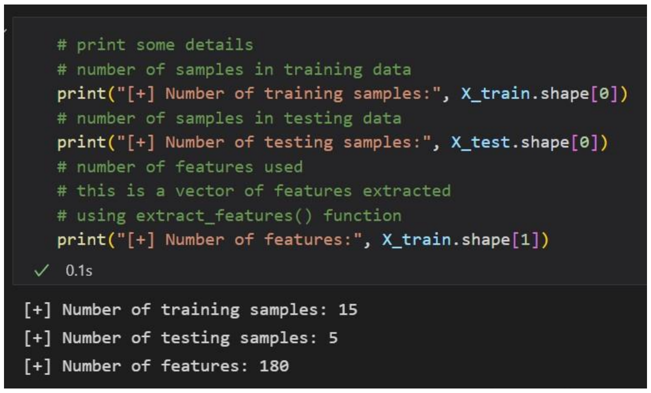
```
# predict 25% of data to measure how good we are
y_pred = model.predict(X_test)

# calculate the accuracy
accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
[19]  ✓ 0.6s

...  Accuracy: 33.33%
```

In another situation, if we only classify two emotions, such as Neutral and Angry.



```
# we allow only these emotions
AVAILABLE_EMOTIONS = {
    "Neutral",
    "Angry"
}
```

As there are only two, fewer samples were being used to train the model with.

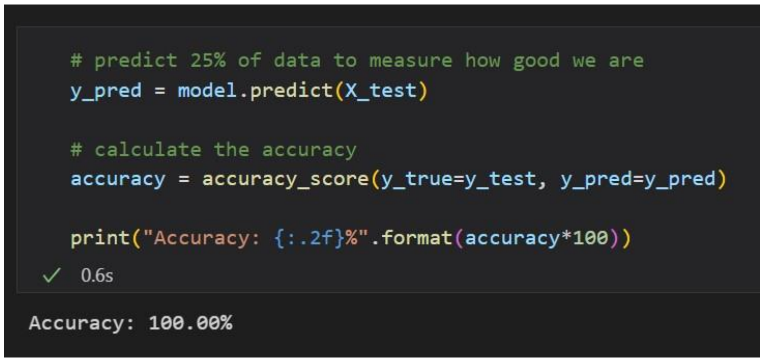


```
# print some details
# number of samples in training data
print("[+] Number of training samples:", X_train.shape[0])
# number of samples in testing data
print("[+] Number of testing samples:", X_test.shape[0])
# number of features used
# this is a vector of features extracted
# using extract_features() function
print("[+] Number of features:", X_train.shape[1])
```

✓ 0.1s

```
[+] Number of training samples: 15
[+] Number of testing samples: 5
[+] Number of features: 180
```

We believe that the model finds it easier to classify two emotions due to the sample size and how it will only differentiate between two emotions.



```
# predict 25% of data to measure how good we are
y_pred = model.predict(X_test)

# calculate the accuracy
accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
```

✓ 0.6s

```
Accuracy: 100.00%
```

X. Save the Model

We created a result directory named "result," where we saved our model `mlp_classifier.model`.

```
if not os.path.isdir("result"):
    os.mkdir("result")

pickle.dump(model, open("result/mlp_classifier.model", "wb"))
```

References

Audio emotions. (2020, June 9). [Dataset]. https://www.kaggle.com/datasets/uldisvalainis/audio-emotions?resource=download

M.T. (2020, July 25). *Speech emotion recognition (SER) through machine learning*. Analytics Insight. Retrieved June 21, 2022, from https://www.analyticsinsight.net/speech-emotion-recognition-ser-through-machine-learning/

Rockikz, A. (2019, July 29). *How to make a speech emotion recognizer using python and scikit-learn - python code*. PythonCode. Retrieved June 21, 2022, from https://www.thepythoncode.com/article/building-a-speech-emotion-recognizer-using-sklearn