GROUP MEMBERS
NEHA BISWAS
SIMRAN DASGUPTA
RIDDHI PAUL
ZAREEN ARSHEE

SPEECH EMOTION RECOGNITION USING PYTHON LIBRARY LIBROSA

INDEX

PROBLEM STATEMENT

To recognise speech in python

TECHNOLOGIES WE USED

Python and Jupyter Notebook

INTRODUCTION

Speech Emotion Recognition is defined as a collection of methodologies which can prove and categorize speech signals to detect emotions embedded in them through loudness, timbre and pitch irrespective of the semantic contents.

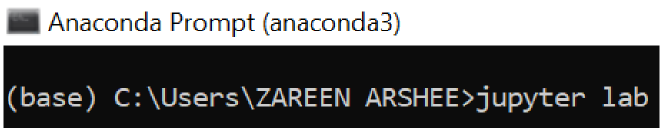
With the advancement of science and technology, Speech Emotion Recognition can be done artificially using programmable devices. It has become a necessity to introduce the world to the services of technology to achieve efficiency like never before as in this advancing Artificial Intelligence (AI) world, Human Computer Interactions (HCI) are of extreme importance.

Python is a programming language which is widely used for this. Its amazing library Librosa and other tools help in achieving the task of speech emotion recognition very accurately and efficiently.

What is librosa?

librosa is a library in Python. It analyses audio and music. It has a standardizes interfaces, flatter package layout, and readable code, backwards compatibility, modular functions, and names.

What is JupyterLab?
JupyterLab is a web-based, open-source user interface for Jupyter Project. It has all basic functionalities of the Jupyter Notebook, such as notebooks, terminals, text editors, file browsers, rich outputs and more.
We will first need to run JupyterLab with the Anaconda prompt to run code in it:



```
(base) C:\Users\ZAREEN ARSHEE>jupyter lab
```

OBJECTIVE

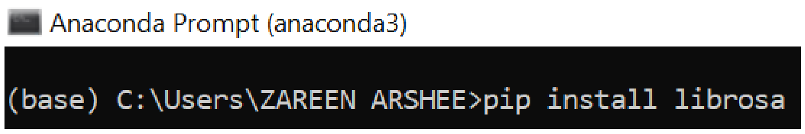
The objective is to discuss about a method to build a model for recognizing emotion from speech using the libraries librosa and sklearn and the dataset RAVDESS. RAVDESS stands for Ryerson Audio-Visual Database of Emotional Speech and Song which is a dataset consisting of 7356 files on emotional validity, genuineness and intensity

PREREQUISITE

Speech Emotion Recognition can be done in Python using its library Librosa. To perform speech emotion recognition, we need to install these libraries from Anaconda Prompt:

Name: Librosa
Version: 0.8.1
Summary: librosa: Python package for music and audio analysis.
Requires: soundfile and audioread

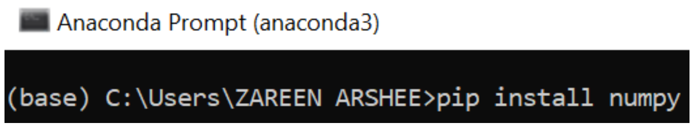
Command for installation:



```
(base) C:\Users\ZAREEN ARSHEE>pip install librosa
```

Name: Numpy
Version: 1.14.1
Summary: NumPy: Python array processing used for strings, numbers, objects and records
Required-by: Scipy, pandas, matplotlib

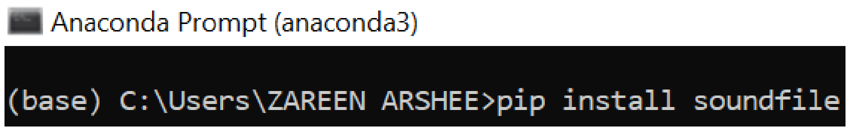
Command for installation:



```
(base) C:\Users\ZAREEN ARSHEE>pip install numpy
```

Name: Soundfile
Version: 0.10.3
Summary: soundfile: Python package to represent audio data as NumPy arrays.
Requires: NumPy and CFFI
Required-by: Librosa

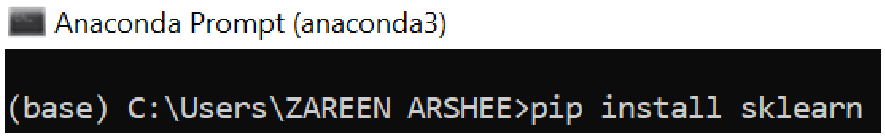
Command for installation:



```
(base) C:\Users\ZAREEN ARSHEE>pip install soundfile
```

Name: sklearn
Version: 1.0.1
Summary: sklearn: Python package for predictive data analysis.
Requires: NumPy, SciPy, and matplotlib

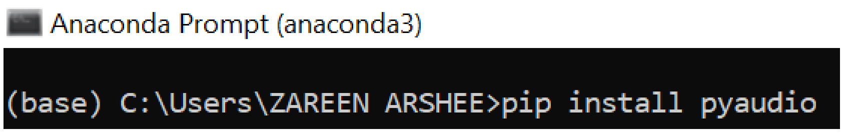
Command for installation:



```
(base) C:\Users\ZAREEN ARSHEE>pip install sklearn
```

Name: pyaudio
Version: 0.2.11
Summary: pyaudio: Python Bindings for PortAudio.
Required-by: MLPClassifier and train_test_split

Command for installation:


Anaconda Prompt (anaconda3)

```
(base) C:\Users\ZAREEN ARSHEE>pip install pyaudio
```

ALGORITHM

STEP 1: START

STEP 2: Make the necessary imports.

STEP 3: Define extract_feature named function to extract the chroma, mel, and mfcc features from a sound file.

STEP 4: To hold the emotions available in the RAVDESS dataset, a list to hold the emotions and numbers, define a dictionary.

STEP 5: Load the data with a function load_data() as parameter to take in the relative size of the test set.

STEP 6: Use the glob() function to get the pathnames for the sound files in dataset.

STEP 7: Observe the shape of the training and testing datasets and get the number of features extracted.

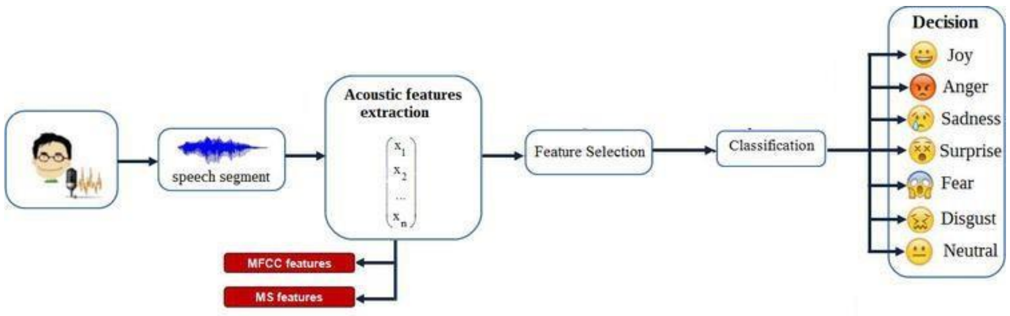
STEP 8: Initialize an MLPClassifier.

STEP 9: Fit/train the model.

STEP 10: Predict the values for the test set. It corresponds to the emotions predicted for the features in the test set.

STEP 11: Calculate the accuracy of the model using function accuracy_score().

STEP 12: STOP



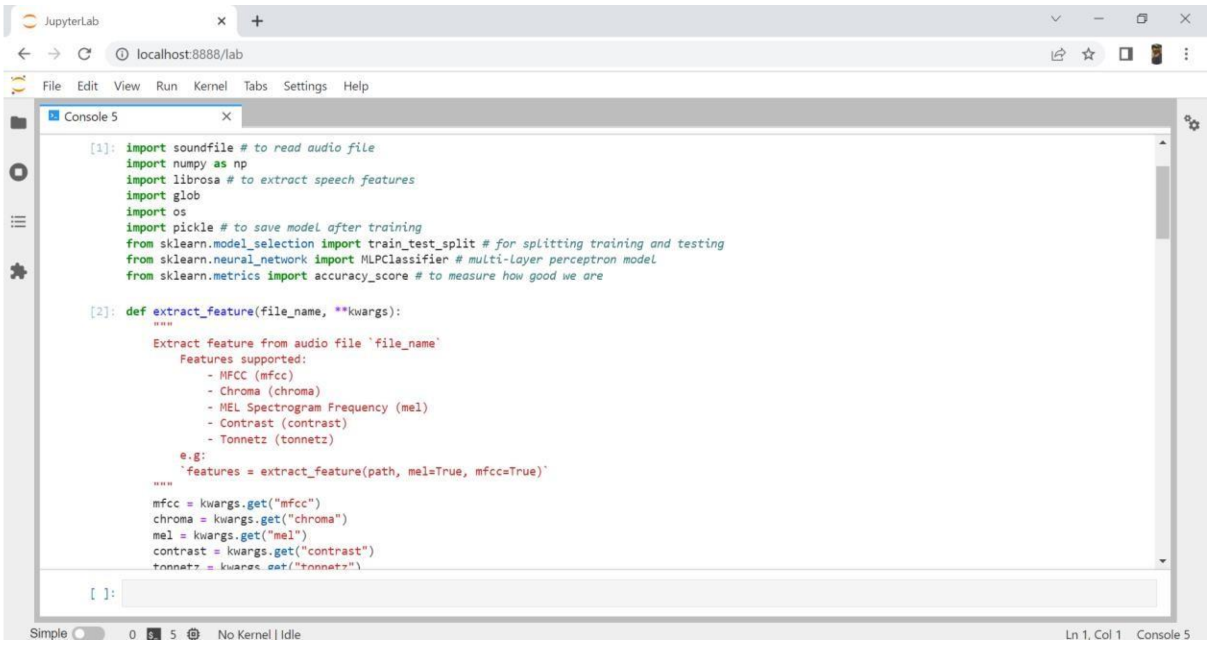
CODE

Optimized code for SER:

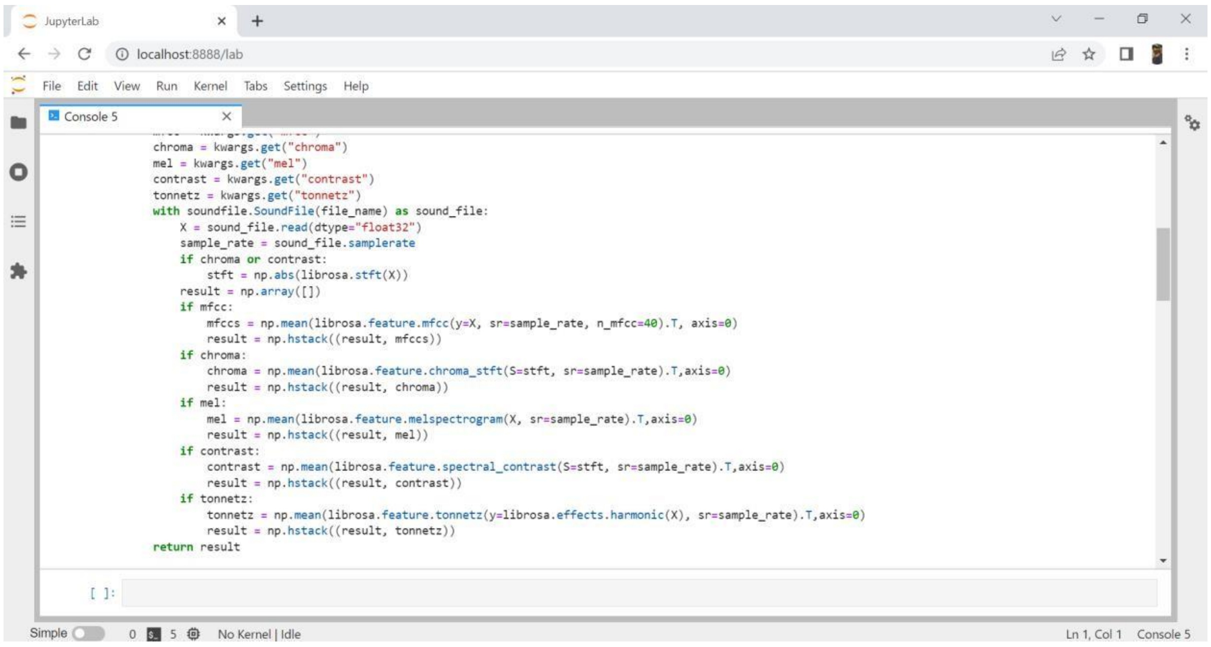
```
import soundfile                                    #to read audio file

import numpy as np

import librosa                                      #to extract speech features

import glob

import os

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


def extract_feature(file_name, **kwargs):

  [ """Extract feature from audio file `file_name`

        Features supported:

            - MFCC (mfcc)

            - Chroma (chroma)

            - MEL Spectrogram Frequency (mel)

            - Contrast (contrast)

            - Tonnetz (tonnetz)

        e.g:

        `features = extract_feature(path, mel=True, mfcc=True)`""" ]

    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


#All emotions on RAVDESS dataset


int2emotion = {
    "01": "neutral",
    "02": "calm",
    "03": "happy",
    "04": "sad",
    "05": "angry",
    "06": "fearful",
    "07": "disgust",
    "08": "surprised"
}

#we allow only these emotions (feel free to tune this on your need)
AVAILABLE_EMOTIONS = {
    "angry",
    "sad",
    "happy",
    "surprised"
}

def load_data(test_size=0.2):
    X, y = [], []
    for file in glob.glob("C:/Users/ZAREEN ARSHEE/data/Actor_*/*.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)

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



#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])

#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 best parameters (so far)
model = MLPClassifier(**model_params)

#Train the model model.fit(X_train,y_train)


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



#Calculate the accuracy of our model accuracy=accuracy_score(y_true=y_test,
y_pred=y_pred)

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

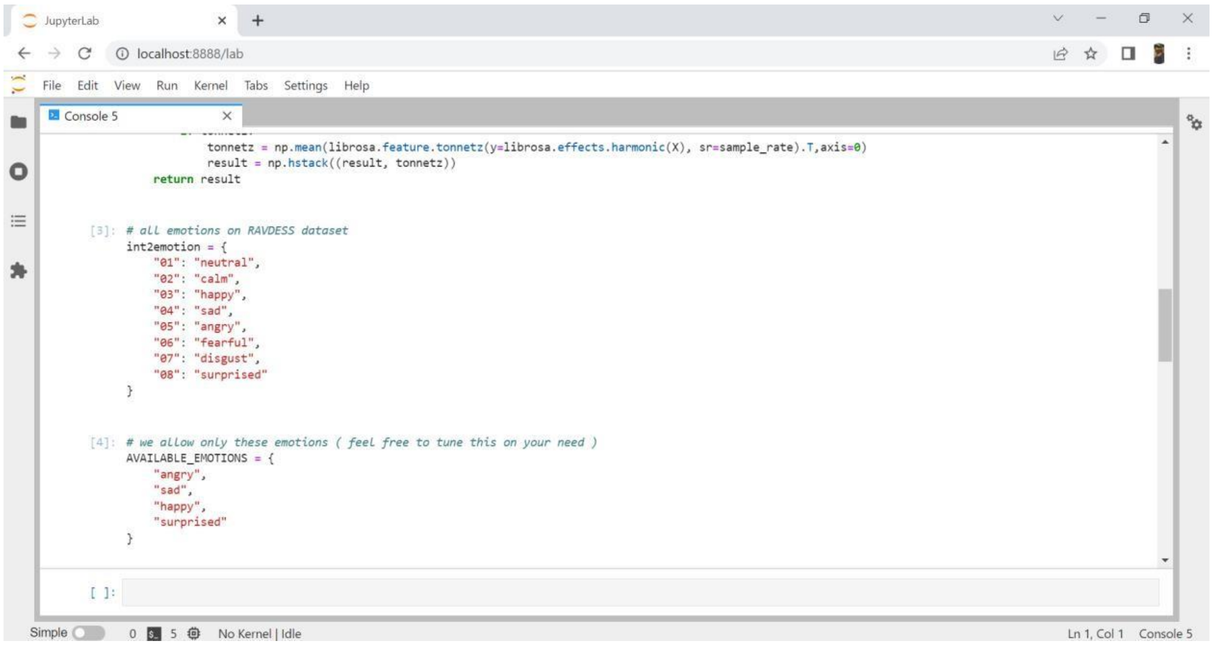
OUTPUT WITH CODE IN JUPYTER LAB:


JupyterLab
localhost:8888/lab
File Edit View Run Kernel Tabs Settings Help
Console 5
Simple 0 5 No Kernel | Idle
Ln 1, Col 1 Console 5

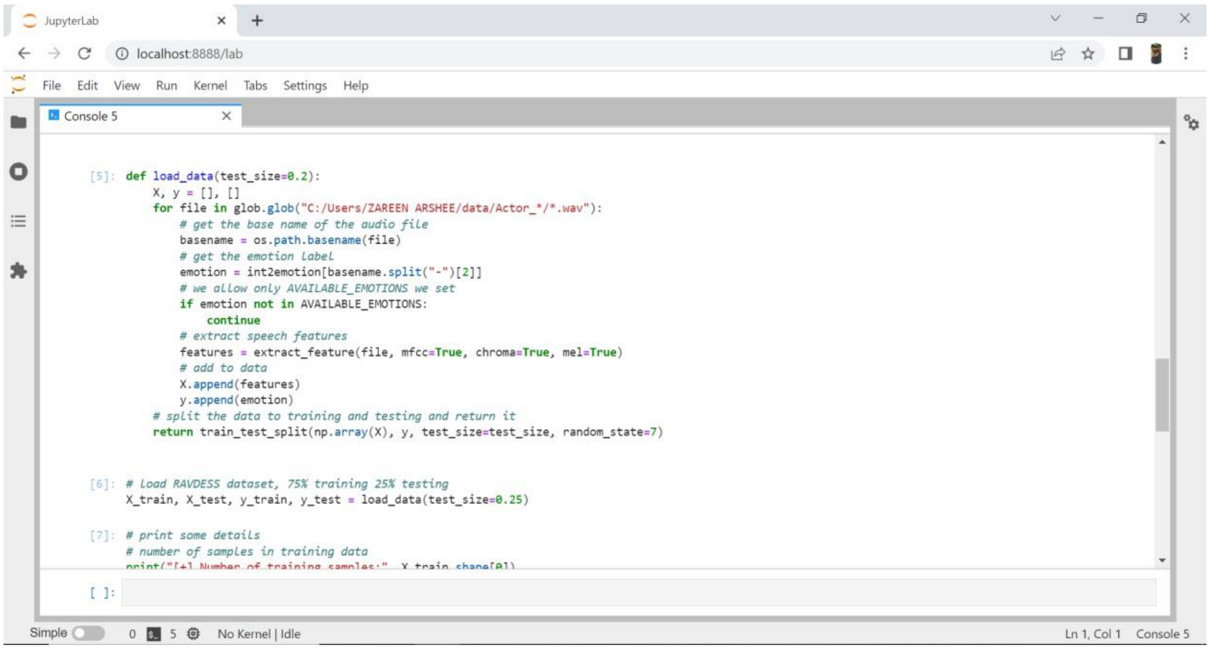
```
[1]: import soundfile # to read audio file
     import numpy as np
     import librosa # to extract speech features
     import glob
     import os
     import pickle # to save model after training
     from sklearn.model_selection import train_test_split # for splitting training and testing
   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

[2]: def extract_feature(file_name, **kwargs):
         """
         Extract feature from audio file `file_name`
             Features supported:
                 - MFCC (mfcc)
                 - Chroma (chroma)
                 - MEL Spectrogram Frequency (mel)
                 - Contrast (contrast)
                 - Tonnetz (tonnetz)
             e.g:
             `features = extract_feature(path, mel=True, mfcc=True)`
         """
         mfcc = kwargs.get("mfcc")
         chroma = kwargs.get("chroma")
         mel = kwargs.get("mel")
         contrast = kwargs.get("contrast")
         tonnetz = kwargs.get("tonnetz")

[ ]:
```


JupyterLab
localhost:8888/lab
File Edit View Run Kernel Tabs Settings Help
Console 5
Simple 0 5 No Kernel | Idle
Ln 1, Col 1 Console 5

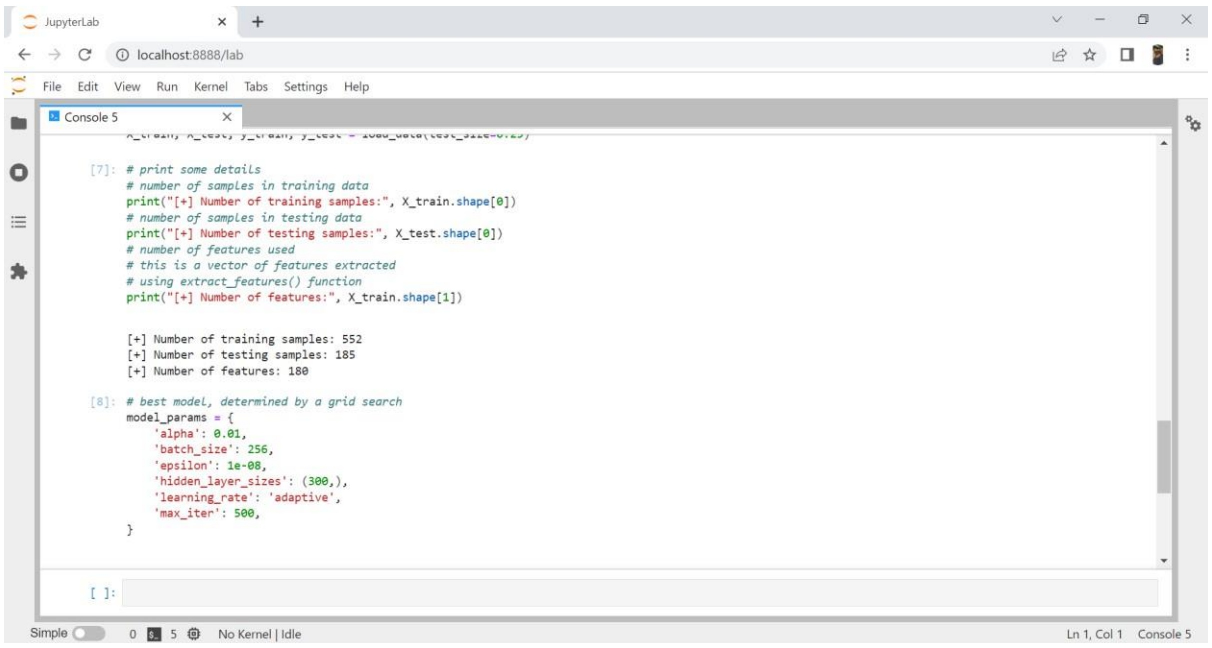
```
         chroma = kwargs.get("chroma")
         mel = kwargs.get("mel")
         contrast = kwargs.get("contrast")
         tonnetz = kwargs.get("tonnetz")
         with soundfile.SoundFile(file_name) as sound_file:
             X = sound_file.read(dtype="float32")
             sample_rate = sound_file.samplerate
             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

[ ]:
```


JupyterLab
localhost:8888/lab
File Edit View Run Kernel Tabs Settings Help
Console 5
Simple 0 5 No Kernel | Idle Ln 1, Col 1 Console 5

```
            tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X), sr=sample_rate).T,axis=0)
            result = np.hstack((result, tonnetz))
        return result


[3]: # all emotions on RAVDESS dataset
     int2emotion = {
         "01": "neutral",
         "02": "calm",
         "03": "happy",
         "04": "sad",
         "05": "angry",
         "06": "fearful",
         "07": "disgust",
         "08": "surprised"
     }


[4]: # we allow only these emotions ( feel free to tune this on your need )
     AVAILABLE_EMOTIONS = {
         "angry",
         "sad",
         "happy",
         "surprised"
     }

[ ]:
```

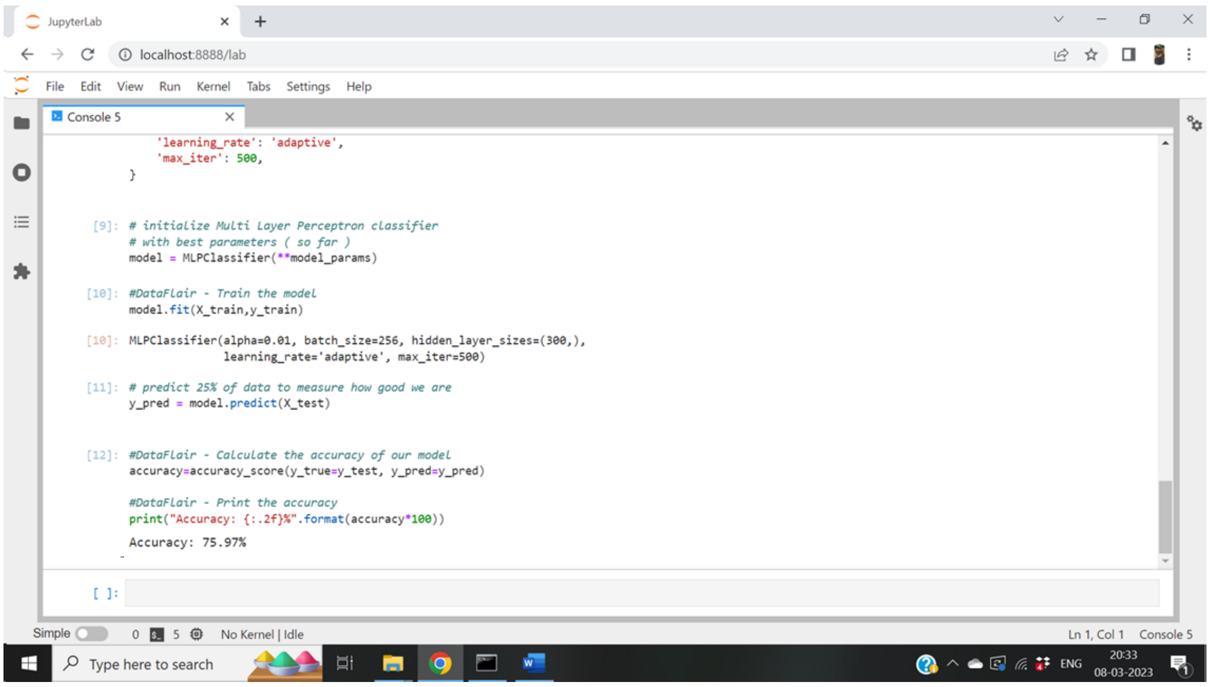

JupyterLab
localhost:8888/lab
File Edit View Run Kernel Tabs Settings Help
Console 5
Simple 0 5 No Kernel | Idle Ln 1, Col 1 Console 5

```
[5]: def load_data(test_size=0.2):
         X, y = [], []
         for file in glob.glob("C:/Users/ZAREEN ARSHEE/data/Actor_*/*.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)


[6]: # load RAVDESS dataset, 75% training 25% testing
     X_train, X_test, y_train, y_test = load_data(test_size=0.25)

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

[ ]:
```



```
X_train, X_test, y_train, y_test = load_data(test_size=0.25)

[7]: # 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])

     [+] Number of training samples: 552
     [+] Number of testing samples: 185
     [+] Number of features: 180

[8]: # 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,
     }

[ ]:
```

Simple 0 5 No Kernel | Idle Ln 1, Col 1 Console 5



JupyterLab
localhost:8888/lab
File Edit View Run Kernel Tabs Settings Help
Console 5

```
         'learning_rate': 'adaptive',
         'max_iter': 500,
     }

[9]: # initialize Multi Layer Perceptron classifier
     # with best parameters ( so far )
     model = MLPClassifier(**model_params)

[10]: #DataFlair - Train the model
      model.fit(X_train,y_train)

[10]: MLPClassifier(alpha=0.01, batch_size=256, hidden_layer_sizes=(300,),
                    learning_rate='adaptive', max_iter=500)

[11]: # predict 25% of data to measure how good we are
      y_pred = model.predict(X_test)

[12]: #DataFlair - Calculate the accuracy of our model
      accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

      #DataFlair - Print the accuracy
      print("Accuracy: {:.2f}%".format(accuracy*100))

      Accuracy: 75.97%

[ ]:
```

APPLICATION

Some of the fields where speech emotion recognition is found are mentioned below: -

- **Emotional behavior in academics**
  With the change in our system of education from offline to mostly online based, it is difficult to interpret what a student all the way cross internet has to express. This can lead to miscommunication. To avoid this SER can be used to track the change in emotion state of a student and adjust teaching strategies.

- **Customer service**
  Voice recognition is an important AI application in customer service. Voice recognition is an effective call center service solution, available 24/7, and is cheaper than live reps. The common use cases of speech recognition in customer service are:
  - ✓ Interactive Voice Response (IVR)
  - ✓ Analytics

- **Healthcare**
  - ✓ MD note-taking - During patient examinations, doctors shouldn't worry about taking notes of patients' symptoms. Medical transcription (MD) software uses speech recognition to capture patient diagnosis notes.
  - ✓ Diagnosis - Depression speech recognition technology listens to a patient's voice to identify the existence, or lack thereof, of depression undertones through words such as "unhappy," "overwhelmed," "bored," "feeling void," etc.

- **Voice biometrics for security**
  Speech biometrics can be used in Fintech to authorize transactions and to guarantee they are genuine and consented from the account owner. Besides, speech biometrics can restrict access to authorized personnel in healthcare, where maintaining patient confidentiality is of utmost importance.

OBSERVATION

SER could also be applied in observing companies' interactions with customers through call centers. A human specialist with limited capacities has to be included currently to analyze emotions in such conversations. But if machines are employed to do the task, then it will be cheaper with a more consistent output and efficient working.

CONCLUSION

At some point in the future, speech recognition may become speech understanding.

The statistical methods that allow computers to decide what a person just said may someday allow them to grasp the meaning behind those words.

Although it needs to tackle various challenges in terms of computational power and software sophistication, some researchers argue that speech recognition development offers the most direct line from the computers of today to true artificial intelligence.