A Project/Dissertation Report

on

SPEECH EMOTION RECOGNITION

Submitted in partial fulfillment of the
requirement for the award of the degree
of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE

Under The Supervision of
Ms Swati Sharma

Submitted By

DAYANAND KUMAR (19SCSE1010810)

SHAIMA PERWEEN(19SCSE1010416)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
OCTOBER 2021

ABSTRACT

Speech Emotion Recognition (SER) is the task of recognising the emotion from speech irrespective of the semantic contents.However, emotions are subjective and even for humans it is hard to notate them in natural speech communication regardless of the meaning.The ability to automatically conduct it is a very difficult task and still an ongoing subject of research. Our project is about automatic emotion recognition system aim to create efficient , real time methods of detecting the emotions of mobile phon user, call centre, operators and customers, car driver, pilots and many more other human machine communication user. Adding emotions to machines has been recognised as a critical factor in making machines appear and act as a human. The presence of various language, accent, sentences, speaking style, speakers also add another difficulty because these characteristics directly change most of the extracted features include pitch, energy. In the world of telemedicine where patients are evaluated over mobile platforms, the ability for a medical professional to discern what the patient is actually feeling can be useful in the healing process. We are going to use it in therapies like stress therapy anxiety therapy. It will help us to cure there kinds of disorders like stress, anxiety.Can be useful to recommend products to customers based on their emotions towards that product. We will use python for this project through some dependencies using pip like Librosa, Numpy,Soundfile,Scikit learning,Pyaudio. We will use MFCC,Chroma and Mel Frequency Cepstrum as a speech features rather than raw waveform.Furthermore, it highlights the current promising direction for improvement of speech emotion recognition systems.

DECEMBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"SPEECH EMOTION RECOGNITION"** in partial fulfillment of the requirements for the award of the B.TECH submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of Ms Swati Sharma assistant professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida.

The matter presented in the thesis/project/dissertation has not been submitted by

DAYANAND KUMAR(19SCSE1010810)

SHAIMA PARWEEN(19SCSE1010416)

me/us for the award of any other degree of this or any other places.

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor : (Ms Swati Sharma)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of DAYANAND KUMAR (19SCSE1010810) and SHAIMA PERWEEN(19SCSE1010416) has been held on____and his/her work is recommended for the award of B.TECH

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

TABLE OF CONTENT

Title	page no
Abstract :	
Candidate declaration:	
Acknowledgement:	
Introduction:	1
Problem indentification :	
Dataset:	
Litreature survey:	4
Proposed system	
IMPLEMENTATION	18
PROBLEM FORMULATION	21
CONCLUSION :	24

REFRENCES: **25**

CHAPTER 1

1.1 INTRODUCTION

Communication is the key to human existence and more often than not, we have to deal with ambiguous situations. For instance, the phrase "This is awesome" could be said under either happy or sad settings. Humans are able to resolve ambiguity in most cases because we can efficiently comprehend information from multiple domains (henceforth, referred to as modalities), namely, speech, text and visual. With the rise of deep learning algorithms, there have been multiple attempts to tackle the task

1.2 FORMULATION OF PROBLEM

Speech emotion recognition, the best ever python mini project. The best example of it can be seen at call centers. If you ever noticed, call centers employees never talk in the same manner, their way of pitching/talking to the customers changes with customers. Now, this does happen with common people too, but how is this relevant to call centers? Here is your answer, the employees recognize customers' emotions from speech, so they can improve their service and convert more people. In this way, they are using speech emotion recognition. So, let's discuss this project in detail.

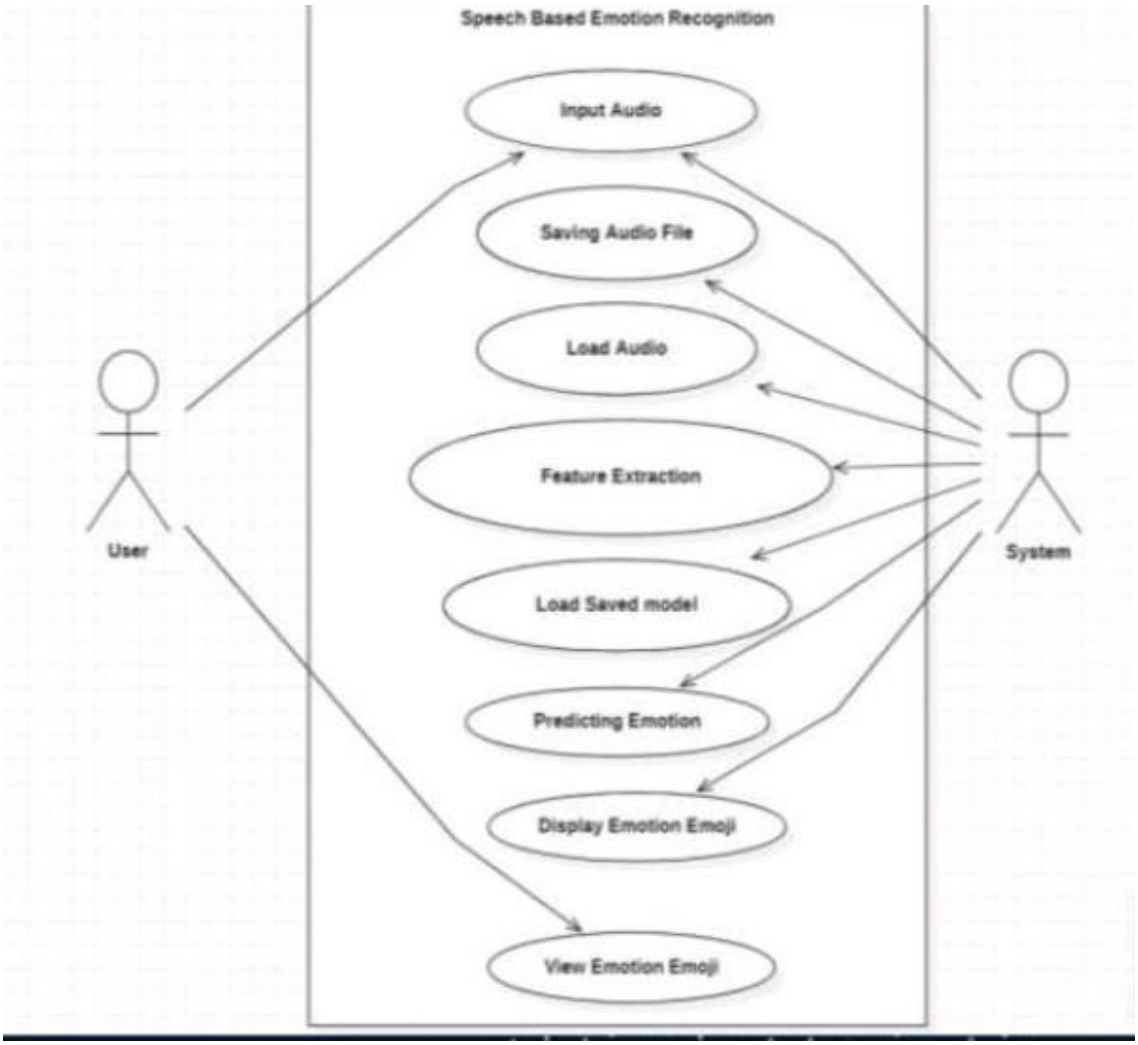
Speech emotion recognition is a simple Python mini-project, which you are going to practice with DataFlair. Before, I explain to you the terms related to this mini python project, make sure you bookmarked the

CHAPTER2

LITERATURE SURVEY/PROJECT DESIGN

In this section, we review some of the work that has been done in the field of speech emotion recognition (SER). The task of SER is not new and has been studied for quite some time in literature. A majority of the early approaches ([6] [7]) used Hidden Markov Models (HMMs) [8] for identifying emotion from speech. Recent introduction of deep neural networks to the domain has also significantly improved the state-of-the-art performance. For instance, [3] and [9] use recurrent autoencoders to solve the task. Recently, methods have also been proposed to efficiently combine features from multiple domains, such as, Tensor Fusion Networks [10] and Low-Rank Matrix Multiplication [11], instead of trivial concatenation. This work aims to provide a comparative study between 1) deep learning based models that are trained end-to-end, and 2) lighter machine learning and deep learning based models trained over handcrafted features. We also investigate the information residing in multiple modalities and how their combination affects the performance.

USE CASE DIAGRAM:



Software Tools

The Dataset

For this Python project, we'll use the RAVDESS dataset; this is the Ryerson Audio-Visual Database of Emotional Speech and Song dataset, and is free to download. This dataset has 7356 files rated by 247 individuals 10 times

on emotional validity, intensity, and genuineness. The entire dataset is 24.8GB from 24 actors

Procedures:

1. Make the necessary imports:

```
import librosa
import soundfile
import os, glob, pickle
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

Screenshot:

2. Define a function extract_feature to extract the mfcc, chroma, and mel features from a sound file. This function takes 4 parameters- the file name and three Boolean parameters for the three features:

- **mfcc:** Mel Frequency Cepstral Coefficient, represents the short-term power spectrum of a sound
- **chroma:** Pertains to the 12 different pitch classes
- **mel:** Mel Spectrogram Frequency

Learn more about Python Sets and Booleans

Open the sound file with soundfile.SoundFile using with-as so it's automatically closed once we're done. Read from it and call it X. Also, get the sample rate. If chroma is True, get the Short-Time Fourier Transform of X.

Let result be an empty numpy array. Now, for each feature of the three, if it exists, make a call to the corresponding function from librosa.feature (eg- librosa.feature.mfcc for mfcc), and get the mean value. Call the function

hstack() from numpy with result and the feature value, and store this in result. hstack() stacks arrays in sequence horizontally (in a columnar fashion). Then, return the result.

```
#DataFlair - Extract features (mfcc, chroma, mel) from a sound file
def extract_feature(file_name, mfcc, chroma, mel):
with soundfile.SoundFile(file_name) as sound_file:
X = sound_file.read(dtype="float32")
sample_rate=sound_file.samplerate
if chroma:
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))
return result
```

Screenshot:

```
[2]: #DataFlair - Extract features (mfcc, chroma, mel) from a sound file
     def extract_feature(file_name, mfcc, chroma, mel):
         with soundfile.SoundFile(file_name) as sound_file:
             X = sound_file.read(dtype="float32")
             sample_rate=sound_file.samplerate
             if chroma:
                 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))
         return result
```

3. Now, let's define a ***dictionary*** to hold numbers and the emotions available in the RAVDESS dataset, and a list to hold those we want- calm, happy, fearful, disgust.

```
#DataFlair - Emotions in the RAVDESS dataset
emotions={
'01':'neutral',
```

```
'02':'calm',
'03':'happy',
'04':'sad',
'05':'angry',
'06':'fearful',
'07':'disgust',
'08':'surprised'
}
#DataFlair - Emotions to observe
observed_emotions=['calm', 'happy', 'fearful', 'disgust']
```

Screenshot:

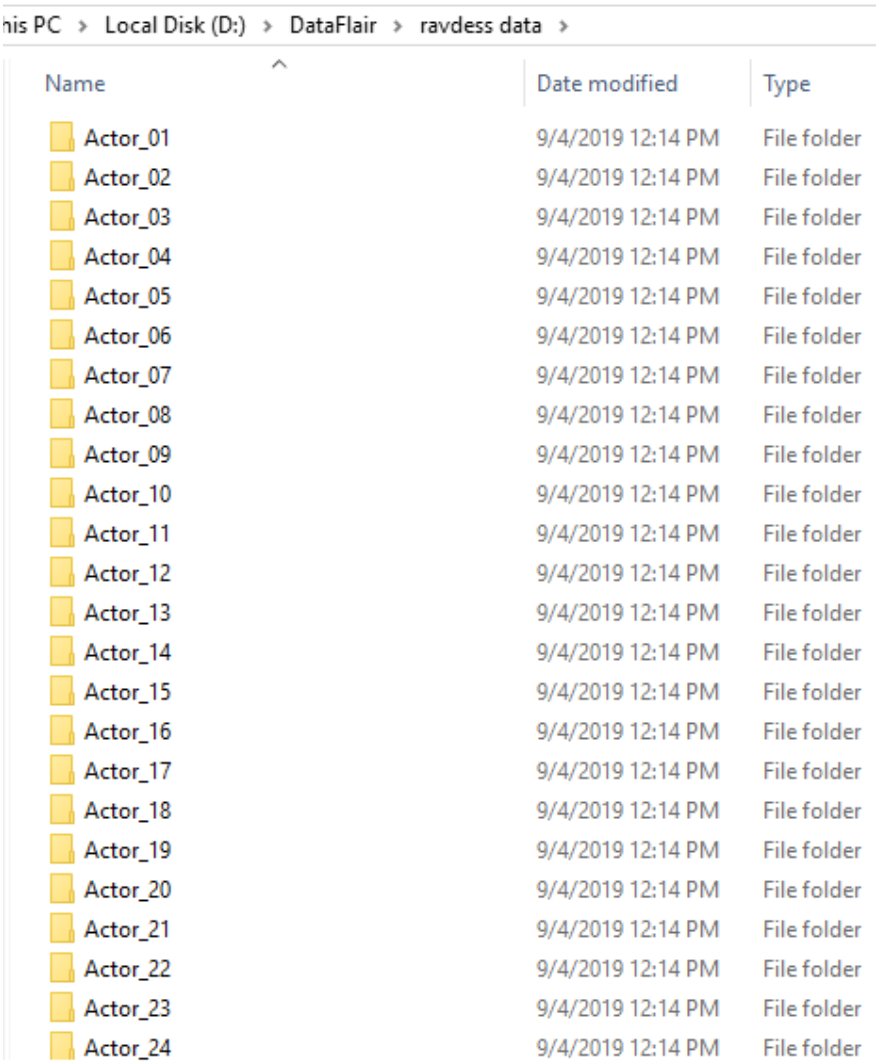
```
[3]: #DataFlair - Emotions in the RAVDESS dataset
     emotions={
         '01':'neutral',
         '02':'calm',
         '03':'happy',
         '04':'sad',
         '05':'angry',
         '06':'fearful',
         '07':'disgust',
         '08':'surprised'
     }

     #DataFlair - Emotions to observe
     observed_emotions=['calm', 'happy', 'fearful', 'disgust']
```

Prepare with DataFlair – *Frequently Asked Python Interview Questions*

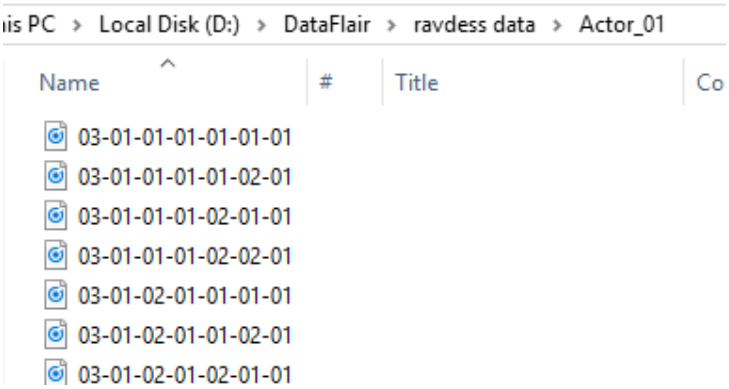
4. Now, let's load the data with a function load_data() – this takes in the relative size of the test set as parameter. x and y are empty lists; we'll use the glob() function from the glob module to get all the pathnames for the sound files in our dataset. The pattern we use for this is: "D:\\DataFlair\\ravdess data\\Actor_**.wav". This is because our dataset looks like this:

Screenshot:



So, for each such path, get the basename of the file, the emotion by splitting the name around '-' and extracting the third value:

Screenshot:



Using our emotions dictionary, this number is turned into an emotion, and our function checks whether this emotion is in our list of observed_emotions; if

not, it continues to the next file. It makes a call to extract_feature and stores what is returned in 'feature'. Then, it appends the feature to x and the emotion to y. So, the list x holds the features and y holds the emotions. We call the function train_test_split with these, the test size, and a random state value, and return that.

```
#DataFlair - Load the data and extract features for each sound file
def load_data(test_size=0.2):
x,y=[],[]
for file in glob.glob("D:\\DataFlair\\ravdess data\\Actor_*\\*.wav"):
file_name=os.path.basename(file)
emotion=emotions[file_name.split("-")[2]]
if emotion not in observed_emotions:
continue
feature=extract_feature(file, mfcc=True, chroma=True, mel=True)
x.append(feature)
y.append(emotion)
return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
```

Screenshot:

```
[4]: #DataFlair - Load the data and extract features for each sound file
     def load_data(test_size=0.2):
         x,y=[],[]
         for file in glob.glob("D:\\DataFlair\\ravdess data\\Actor_*\\*.wav"):
             file_name=os.path.basename(file)
             emotion=emotions[file_name.split("-")[2]]
             if emotion not in observed_emotions:
                 continue
             feature=extract_feature(file, mfcc=True, chroma=True, mel=True)
             x.append(feature)
             y.append(emotion)
         return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
```

5. Time to split the dataset into training and testing sets! Let's keep the test set 25% of everything and use the load_data function for this.

```
#DataFlair - Split the dataset
x_train,x_test,y_train,y_test=load_data(test_size=0.25)
```

Screenshot:

```
[5]: #DataFlair - Split the dataset
     x_train,x_test,y_train,y_test=load_data(test_size=0.25)
```

6. Observe the shape of the training and testing datasets:

```
#DataFlair - Get the shape of the training and testing datasets
print((x_train.shape[0], x_test.shape[0]))
```

Screenshot:

```
[6]: #DataFlair - Get the shape of the training and testing datasets
     print((x_train.shape[0], x_test.shape[0]))

     (576, 192)
```

7. And get the number of features extracted.

```
#DataFlair - Get the number of features extracted
print(f'Features extracted: {x_train.shape[1]}')
```

Output Screenshot:

```
[7]: #DataFlair - Get the number of features extracted
     print(f'Features extracted: {x_train.shape[1]}')

     Features extracted: 180
```

8. Now, let's initialize an MLPClassifier. This is a Multi-layer Perceptron Classifier; it optimizes the log-loss function using LBFGS or stochastic gradient descent. Unlike SVM or ***Naive Bayes***, the MLPClassifier has an internal neural network for the purpose of classification. This is a feedforward ANN model.

```
#DataFlair - Initialize the Multi Layer Perceptron Classifier
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,),
learning_rate='adaptive', max_iter=500)
```

Screenshot:

```
[8]: #DataFlair - Initialize the Multi Layer Perceptron Classifier
     model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
```

9. Fit/train the model.

```
#DataFlair - Train the model
model.fit(x_train,y_train)
```

Output Screenshot:

```
[9]: #DataFlair - Train the model
     model.fit(x_train,y_train)

[9]: MLPClassifier(activation='relu', alpha=0.01, batch_size=256, beta_1=0.9,
                   beta_2=0.999, early_stopping=False, epsilon=1e-08,
                   hidden_layer_sizes=(300,), learning_rate='adaptive',
                   learning_rate_init=0.001, max_iter=500, momentum=0.9,
                   n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
                   random_state=None, shuffle=True, solver='adam', tol=0.0001,
                   validation_fraction=0.1, verbose=False, warm_start=False)
```

10. Let's predict the values for the test set. This gives us y_pred (the predicted emotions for the features in the test set).

```
#DataFlair - Predict for the test set
y_pred=model.predict(x_test)
```

Screenshot:

```
[10]: #DataFlair - Predict for the test set
      y_pred=model.predict(x_test)
```

11. To calculate the accuracy of our model, we'll call up the accuracy_score() function we imported from sklearn. Finally, we'll round the accuracy to 2 decimal places and print it out.

```
#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))
```

Output Screenshot:

```
[11]: #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: 72.40%
```

```
[ ]:
```

Summary

In this project, we learned to recognize emotions from speech. We used an MLPClassifier for this and made use of the soundfile library to read the sound file, and the librosa library to extract features from it. As you'll see, the model delivered an accuracy of 72.4%. That's good enough for us yet.

IMPLEMENTATION

Fit/train the model.

```
#DataFlair - Train the model
model.fit(x_train,y_train)
```

Output Screenshot:

```
[9]: #DataFlair - Train the model
     model.fit(x_train,y_train)

[9]: MLPClassifier(activation='relu', alpha=0.01, batch_size=256, beta_1=0.9,
                   beta_2=0.999, early_stopping=False, epsilon=1e-08,
                   hidden_layer_sizes=(300,), learning_rate='adaptive',
                   learning_rate_init=0.001, max_iter=500, momentum=0.9,
                   n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
                   random_state=None, shuffle=True, solver='adam', tol=0.0001,
                   validation_fraction=0.1, verbose=False, warm_start=False)
```

Let's predict the values for the test set. This gives us y_pred (the predicted emotions for the features in the test set).

```
#DataFlair - Predict for the test set
y_pred=model.predict(x_test)
```

Screenshot:

```
[10]: #DataFlair - Predict for the test set
      y_pred=model.predict(x_test)
```

To calculate the accuracy of our model, we'll call up the accuracy_score() function we imported from sklearn. Finally, we'll round the accuracy to 2 decimal places and print it out.

```
#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))
```

Output Screenshot:

```
[11]: #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: 72.40%
```

```
[ ]:
```

Conclusion:

In this project, we learned to recognize emotions from speech. We used an MLPClassifier for this and made use of the soundfile library to read the sound file, and the librosa library to extract features from it. As you'll see, the model delivered an accuracy of 72.4%. That's good enough for us yet.

REFRENCES:

[1] Patil, V. C., Al-Gaadi, K. A., Biradar, D. P., & Rangaswamy, M. (2012). Internet of things (IoT) and cloud computing for agriculture: An overview. *Proceedings of agro-informatics and precision agriculture (AIPA 2012), India*, 292-296.

[2] Patil, K. A., & Kale, N. R. (2016, December). A model for smart agriculture using IoT. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 543- 545.IEEE.

[3] Veena, S., Mahesh, K., Rajesh, M., & Salmon, S. (2018). The survey on smart agriculture using IOT. *Int J Innov Res EngManag (IJRIREM)*, *5*(2), 63-66.

[4] Chauhan, N., Krishnakanth, M., Kumar, G. P., Jotwani, P., Tandon, U., Gosh, A., ...& Santhi, V. (2019, March). Crop Shop–An application to maximize profit for farmers. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, 1-7. IEEE.

[5] Barcelo-Ordinas, J. M., Chanet, J. P., Hou, K. M., &García-Vidal, J. (2013). A survey of wireless sensor technologies applied to precision agriculture. In *Precision agriculture'13*, 801-808.Wageningen Academic Publishers, Wageningen.

[6] Mallick, C., & Satpathy, S. (2018). Challenges and Design Goals of Wireless Sensor Networks: A State-of-the-art Review. *International Journal of Computer Applications*, *179*(28), 42-47.

[7] Hung, M. C., Wu, J., Chang, J. H., & Yang, D. L. (2005). An efficient k-means clustering algorithm using simple partitioning. *Journal of information science and engineering*, *21*(6), 1157-1177.

[8] Nalini, N., & Suvithavani, P. (2017). A Study on Data Analytics: Internet of Things &Health- Care. *International Journal of Computer Science and Mobile Computing (IJCSMC)*, *6*(3), 20-27.

[9] Sawaitul, S. D., Wagh, K. P., & Chatur, P. N. (2012). Classification and prediction of future weather by using back propagation algorithm-an approach. *International Journal of Emerging Technology and*