JSSMAHAVIDYAPEETHA

JSSSCIENCEANDTECHNOLOGYUNIVERSITY

“Speech and emotion recognition”

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE

BY

Student Name	USN
Neethushree GN	01JCE21CS131
Bhoomika KS	01JCE21CS019
Nandan KR	01JST21CS089

Under the guidance of

Rakshitha R

Assistant Professor

Dept. of CSE, JSS STU Mysuru-06

2023-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ABSTRACT

Language is human’s most important communication and speech is basic medium of communication. Emotion plays a crucial role in social interaction. Recognizing the emotion in a speech is important as well as challenging because here we are dealing with human machine interaction. Emotion varies from person to person were same person have different emotions all together has different way express it. When a person express his emotion each will be having different energy, pitch and tone variation are grouped together considering upon different subject. Therefore the speech emotion recognition is a future goal of computer vision. The aim of our project is to develop the smart emotion recognition speech based on the convolutional neural network. Which uses different modules for emotion recognition and the classifier are used to differentiate emotion such as happy sad angry surprise. The machine will convert the human speech signals into waveform and process its routine at last it will display the emotion. The data is speech sample and the characteristics are extracted from the speech sample using librosa package. We are using RAVDESS dataset which are used as an experimental dataset. This study shows that for our dataset all classifiers achieve an accuracy of 73%.

Keywords : Emotion, RAVDESS Dataset, Speech Emotion Recognition, Convolutional neuralnetwork.

TABLE OF CONTENT

1. INTRODUCTION

We being a normal person we can easily understand human emotions but it is difficult for a machine to understand. For a machine to understand emotions we use machine learning to teach how it can recognize emotions. Algorithm which build a model by training data in order to make decision or predication without being programmed to do so. Machines are used in multiple area like security, face recognition, email filtering, fruit detection, crop detection and etc. In this project we used convolutional neural network algorithm. CNN takes multiple inputs and find the best output by the required given output. Here we are using a RAVDESS dataset which consist of 1400 audios. Each audio are different from each other and having a different type of emotion in it. We divide the audio by removing the noisy part from the audio and the clean voice is send to the clean folder entire process will be mentioned in further of this document. The audio will go under test and train then at the end display the emotion. Also we done a live demo section where we give a live audio or a human voice and the machine will predict the correct emotion because it is trained to identify the exact emotion that a voice is having. In future this technology can be used in student teacher emotion interaction so that they can find emotions easily by the use of machine.The presence of various language, accent, sentences, speaking style, speakers also add another difficulty because these characteristics directly change most of the extracted features includes pitch, energy, etc. Speech emotion recognition is tough because emotions are subjective and annotating audio is challenging. The idea of creating this project was to build a machine learning model that could detect emotions from speech we have with us all the time. In this we have used librosa and MLP classifier, here librosa is used for analyzing audio and music. It has flatter package layout, standardizes interfaces and names, backwards compatibility, modular functions, and readable code. The MLP-Classifier is used to classify the emotions from the given wave of learning rate to be adaptive. In this study we attempt to dettect underlying emotions in recorded speech by analysing the acoustic features of the audio data of recordings. There are three classes of features in speech namely, lexical , visual and acoustic features. The problem of speech emotion recognition can be solved by analysing one or more of these features.

2. LITERATURE REVIEW

[1] Title: “A comparison of the discrete and dimensional models of the emotion in music“

Author: TuomasEerola and Jonna K. Vuoskoski, Psychology of Music, 1-32, The Author(s) 2010.

The primary aim of the present study is to contribute to the theoretical debate currently occupying music and emotion research by systematically comparing evaluations of perceived emotions using two different theoretical frameworks: the discrete emotion model, and dimensional model of affect. The importance of the comparison lies not only in the prevalence of these models in music and emotion studies, but also in the suggested neurological differences involved in emotion categorization and the evaluation of emotion dimensions, as well as in the categorically constrained affect space the excerpts have represented to date. Moreover, the various alternative formulations of the dimensional model have not been investigated in music and emotion studies before. A secondary aim is to introduce a new, improved set of stimuli – consisting of unfamiliar, thoroughly tested and validated non-synthetic music excerpts – for the study of musicmediated emotions. Moreover, this set of stimuli should not only include the best examples of target emotions but also moderate examples that permit the study of more subtle variations in emotion.

[2] Title: DEAP: A Database for Emotion Analysis using Physiological Signals

Author: Sander Koelstra, Student Member, IEEE, Christian M¨uhl, Mohammad Soleymani, Student Member, IEEE,JongSeok Lee, Member, IEEE, Ashkan Yazdani, Touradj Ebrahimi, Member, IEEE,Thierry Pun, Member, IEEE, Anton Nijholt, Member, IEEE, Ioannis Patras, Member,IEEE.

In this work, we have presented a database for the analysis of spontaneous emotions. The database contains physiological signals of 32 participants, where each participant watched and rated their emotional response to 40 music videos along the scales of arousal, valence, and dominance, as well as their liking of and familiarity with the videos. We presented a novel semiautomatic stimuli selection method using affective tags, which was validated by an analysis of the ratings participants gave during the experiment. Significant correlates were found between the participant ratings and EEG frequencies. Single-trial classification was performed for the scales of arousal, valence and liking using features extracted from the EEG, peripheral and MCA modalities. The results were shown to be significantly better than random classification. Finally, decision fusion of these results yielded a modest increase in the performance, indicating at least some complementarity to the modalities.The database is made publicly available and it is our hope that other researchers will try their methods and algorithms on this highly challenging database.

[3].Title: “Introduction to Digital Speech Processing"

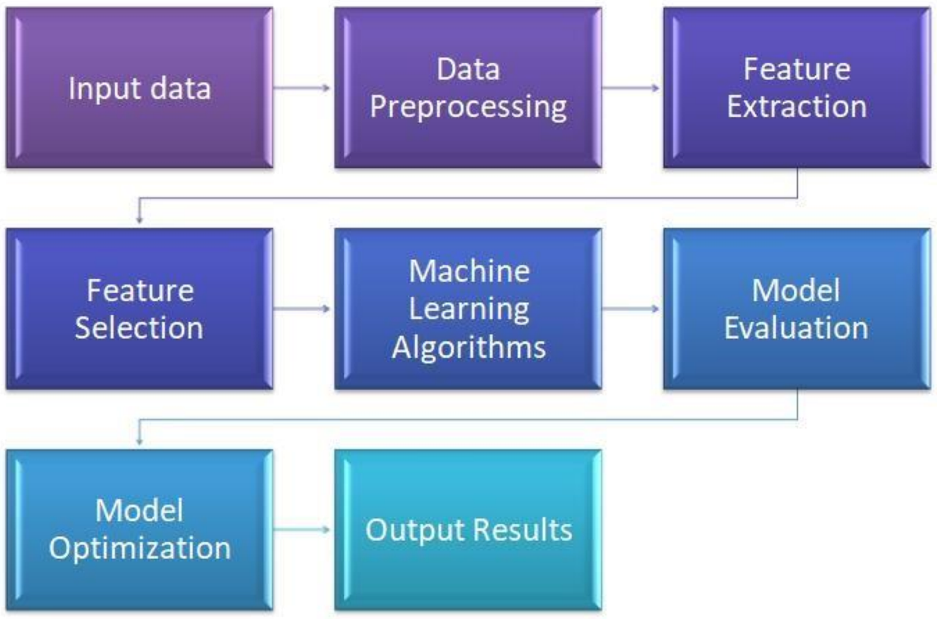
Authors: Lawrence R Rabiner Ronald W Schafer, , Vol. 1, Nos. 1–2 (2007) 1–194, 2007 L. R. Rabiner and R. W... Schafer.

Associative advances in incorporated circuit innovation and PC design have adjusted to make a mechanical domain with for all intents and purposes boundless open doors for development in speech communication applications. In this content, The initial phase in many uses of digital speech processing is to change over the acoustic waveform to a grouping of numbers. Most present day A-to-D converters work by inspecting at an exceptionally high rate, applying digital low pass filter with cutoff set to protect an endorsed data transmission, and after that lessening the sampling rate to the ideal testing rate, which can be as low as double the cutoff frequency of the sharp-cutoff digital filter. This discrete-time representation is the beginning stage for generally applications.

Starting here, different representations are gotten by digital processing.

3. METHODOLOGY

System implementation is a stage where models are converted into a working system, entirely new applications are built by replacing the old one using all the freshly implemented design. Dataset, feature extraction, testing and training are the important stages of this project.Speech emotion recognition (SER) begins with the acquisition of speech signals, typically through microphones or telecommunication systems. The first step involves preprocessing the raw speech signals to remove noise and normalize the data. Following preprocessing, relevant features are extracted from the speech signals, such as pitch, energy, spectral features, and prosodic features. Feature selection techniques may then be applied to reduce dimensionality and enhance the discriminative power of the feature set. Once the feature set is prepared, classification algorithms, such as support vector machines (SVM), artificial neural networks (ANN), or hidden Markov models (HMM), are trained using labeled data to recognize emotional states. Evaluation of the SER system is conducted using metrics like accuracy, precision, recall, and F1score, often through cross-validation techniques to ensure robustness and generalization. Finally, the performance of the SER system is assessed on unseen data to measure its effectiveness in real-world scenarios.



3.1 PROPOSED SYSTEM

In this current study, we presented an automatic speech emotion recognition (SER) system using machine learning algorithms to classify the emotions.The performance of the emotion detection system can greatly influence the overall performance of the application in many ways and can provide many advantages over the functionalities of these applications. This research presents a speech emotion detection system with improvements over an existing system in terms of data, feature selection, and methodology that aims at classifying speech percepts based on emotions, more accurately.

3.2 SCOPE

The scope of this SRSdocument persists for the entire life cycle of the project. This document defines the final state of the software requirements agreed upon by the customers and designers. Finally at the end of the project execution all the functionalities may be traceable from the SRSto the product. The document describes the functionality, performance, constraints, interface and reliability for the entire life cycle of the project.

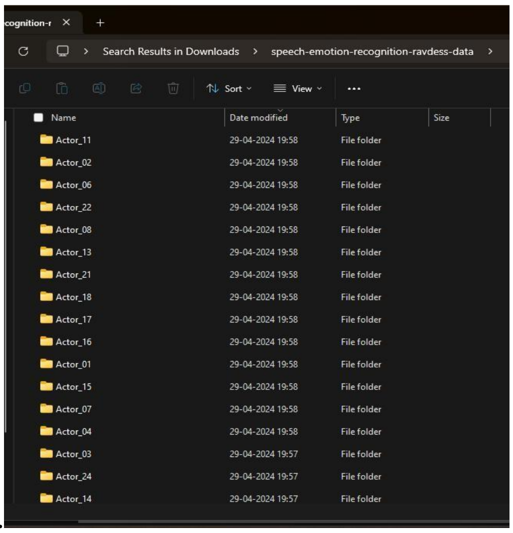
3.3 EXISTING SYSTEM

The speech emotion detection system is implemented as a Machine Learning (ML) model. The steps of implementation are comparable to any other ML project, with additional fine-tuning procedures to make the model function better. The flowchart represents a pictorial overview of the process (see Figure 1). The first step is data collection, which is of prime importance. The model being developed will learn from the data provided to it and all the decisions and results that a developed model will produce is guided by the data. The second step, called feature engineering, is a collection of several machine learning tasks that are executed over the collected data. These procedures address the several data representation and data quality issues. The third step is often considered the core of an ML project where an algorithmic based model is developed. This model uses an ML algorithm to learn about the data and train itself to respond to any new data it is exposed to. The final step is to evaluate the functioning of the built model. Very often, developers repeat the steps of developing a model and evaluating it to compare the performance of different algorithms. Comparison results help to choose the appropriate ML algorithm most relevant to the problem.

3.4 DATASET

RAVDESS dataset has recordings of 24 actors, 12 male actors, and 12 female actors, the actors are numbered from 01 to 24. The male actors are odd in number and female actors are even in number. The emotions contained in the dataset are as sad, happy, neural, angry, disgust, surprised, fearful and calm expressions. The dataset contains all expressions in three formats, those are: Only Audio, AudioVideo and Only Video. Since our focus is on recognising emotions from speech, this model is trained on Audio-only data.

A Multi-Layer perceptron(MLP) is a network made up of perceptrons. It has an input layer that receives the input signal, an output layer that makes predictions or decisions for a given input, and the layers present in between the input and output layer are called hidden layers. In the proposed methodology for Speech Emotion Recognition, the MLP network will have one input layer, 300,40,80,40 hidden layers and one output layer. The hidden layers will be large numbers and the number of hidden layers can be changed as per requirements.

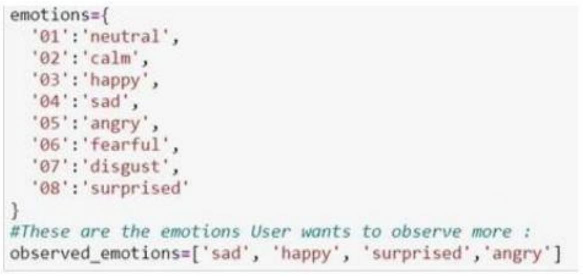

cognition-r
Search Results in Downloads > speech-emotion-recognition-ravdess-data >
Sort
View

Name	Date modified	Type	Size
Actor_11	29-04-2024 19:58	File folder	
Actor_02	29-04-2024 19:58	File folder	
Actor_06	29-04-2024 19:58	File folder	
Actor_22	29-04-2024 19:58	File folder	
Actor_08	29-04-2024 19:58	File folder	
Actor_13	29-04-2024 19:58	File folder	
Actor_21	29-04-2024 19:58	File folder	
Actor_18	29-04-2024 19:58	File folder	
Actor_17	29-04-2024 19:58	File folder	
Actor_16	29-04-2024 19:58	File folder	
Actor_01	29-04-2024 19:58	File folder	
Actor_15	29-04-2024 19:58	File folder	
Actor_07	29-04-2024 19:58	File folder	
Actor_04	29-04-2024 19:58	File folder	
Actor_03	29-04-2024 19:57	File folder	
Actor_24	29-04-2024 19:57	File folder	
Actor_14	29-04-2024 19:57	File folder	

.

Fig 1. Folder containing audio files

There are 8 emotions basically but here we are dealing with only 4 that are happy, sad, surprised, and angry.



```
emotions={
  '01':'neutral',
  '02':'calm',
  '03':'happy',
  '04':'sad',
  '05':'angry',
  '06':'fearful',
  '07':'disgust',
  '08':'surprised'
}
#These are the emotions User wants to observe more :
observed_emotions=['sad', 'happy', 'surprised','angry']
```

Fig 2. Emotion in dataset

3.5 FEATURE EXTRACTION

It is the function where we extract the mfcc, chroma, and mel features from a sound file. This will take 4 parameters, the file name and three Boolean parameters.

The three features are mfcc : mel frequency cepstral coefficient, represent the power spectrum of a sound.

Chroma: pertains to the 12 pitch classes mel: mel spectrum frequency Sound file will be opened and it will be readied and result will be saved to array. For each of three, if it exists then a call will be made to the corresponding function from librosa. Mean value will be noted and result along with feature value and storing it in a file.

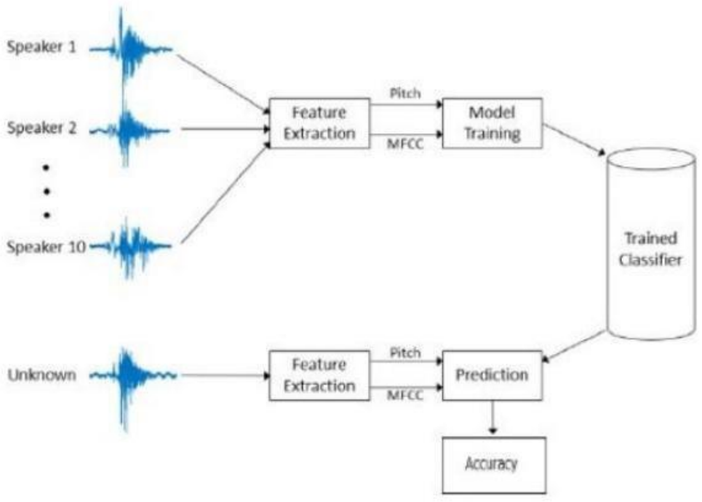
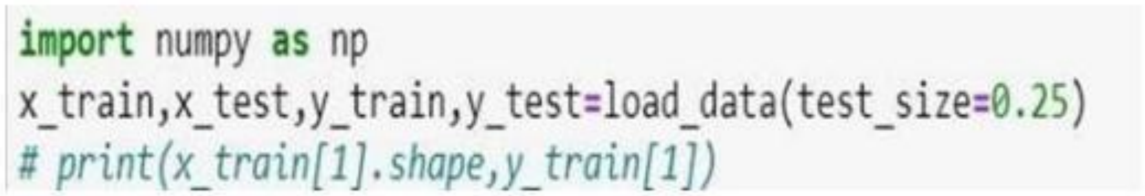


Figure 3. Speech Emotion Recognition System

3.6 TRAINING AND TESTING

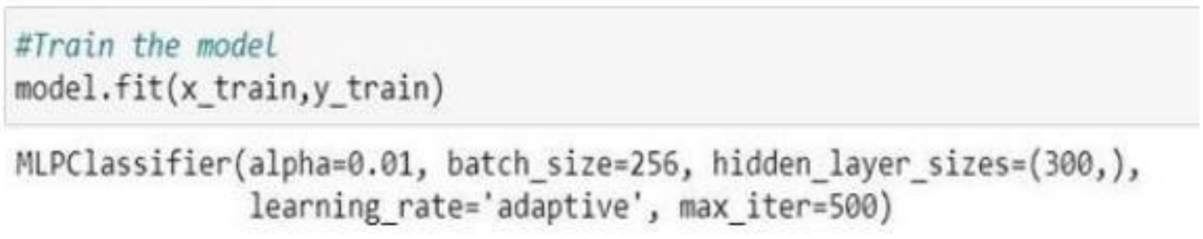
We are loading the data where it takes in the relative size of the test set as a parameter. X and Y are empty lists, functions will check whether the emotions are in the list of observed emotions. The feature will be sent to X and emotions to Y. Now the testing and training function will be called. 75% of audio will be tested at the same time 25% of audio will be trained. For classification we are using MLP Classifier.



```
import numpy as np
x_train,x_test,y_train,y_test=load_data(test_size=0.25)
# print(x_train[1].shape,y_train[1])
```

Fig 4. Training and testing a speech

Finally the system will be ready and it will trained using fit/train model



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

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

Fig 5. Training the model

3.7 SYSTEM DESIGN

The model is trained using several user speech inputs made by different actors. The audio data is analysed and then its feature is extracted using Librosa library. In the above diagram the flow is shown. After extracting the features it is labelled with appropriate labels. The trained model is saved using the pickle library and then loaded whenever it is required. The prediction of emotion is done by taking the audio data from the microphone and preprocessing it i.e., extracting its features and feeding it to the model.The model will predict the emotion output as it takes the input as shown in Figure 6.

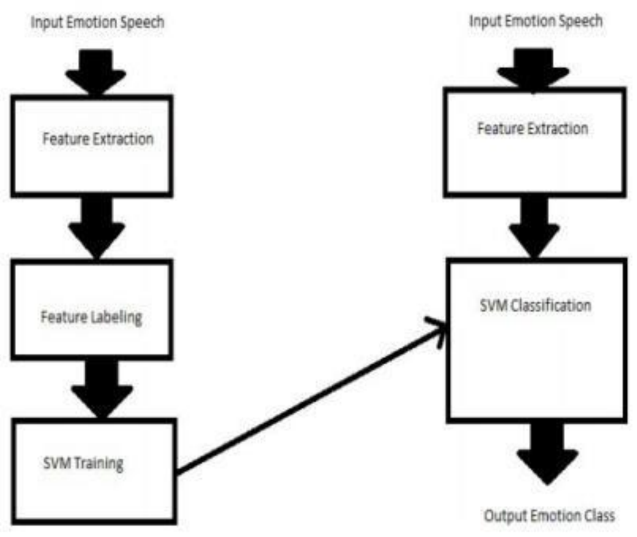


Figure 6. Block Diagram of the System

3.8 ALGORITHMS USED

CLASSIFIERS

Classification is defined as the process of recognition, understanding, and grouping of objects and ideas into preset categories a.k.a ―sub- populations.‖ With the help of these pre-categorized training datasets, classification in machine learning programs leverage a wide range of algorithms to classify future datasets into respective and relevant categories. Classification algorithms used in machine learning utilize input training data for the purpose of predicting the likelihood or probability that the data that follows will fall into one of the predetermined categories. One of the most common applications of classification is for filtering emails into ―spam‖ or ―non-spam‖, as used by today‗s top email service providers.In short, classification is a form of―pattern recognition,‖. Here, classification algorithms applied to the training data find the same pattern (similar number sequences, words or sentiments, and the like) in future data sets.

We will explore classification algorithms in detail, and discover how a text analysis software can perform actions like sentiment analysis – used for categorizing unstructured text by opinion polarity (positive, negative, neutral, and the like).

MLP CLASSIFIER

MLPClassifier stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. Unlike other classification algorithms such as Support Vectors or Naive Bayes Classifier, MLPClassifier relies on an underlying Neural Network to perform the task of classification.

A multi-layer rather than a single layer network is required since a single layer perceptron (SLP) can only compute a linear decision boundary, which is not flexible enough for most realistic learning problems. For a problem that is linearly separable, (that is capable of being perfectly separated by linear decision boundary), the perceptron convergence theorem guarantees convergence. In its simplest form, SLP training is based on the simple idea of adding or subtracting a pattern from the current weights when the target and predicted class disagrees, otherwise the weights are unchanged. For a non-linearly separable problem, this simple algorithm can go on cycling indefinitely. The modification known as least mean square (LMS) algorithm uses a mean squared error cost function to overcome this difficulty, but since there is only a single perceptron, the decision boundary is still linear. An MLP is a universal approximator that typically uses the same squared error function as LMS. However, the main difficulty with the MLP is that the learning algorithm has a complex error surface, which can become stuck in local minima. There does not exist any MLP learning algorithm that is guaranteed to converge, as with SLP. The popular MLP back-propagation algorithm has two phases, the first being a forward pass, which is a forward simulation for the current training pattern and enables the error to be calculated. It is followed by a backward pass, that calculates for each weight in the network how a small change will affect the error function. The derivative calculation is based on the application of the chain rule, and training typically proceeds by changing the weights proportional to the derivative.

REGRESSORS

A statistical approach that estimates the relationships among variables and predicts future outcomes or items in a continuous data set by solving for the pattern of past inputs, such as linear regression in statistics. Regression is foundational to machine learning and artificial intelligence. It predicts a real numbered label given an unlabeled example.

Regression analysis is the process of estimating the relationship between a dependent variable and independent variables. In simpler words, it means fitting a function from a selected family of functions to the sampled data under some error function. Regression analysis is one of the most basic tools in the area of machine learning used for prediction. Using regression you fit a function on the available data and try to predict the outcome for the future or hold-out datapoints. This fitting of function serves two purposes.

You can estimate missing data within your data range (Interpolation) You can estimate future data outside your data range (Extrapolation) Some real-world examples for regression analysis include predicting the price of a house given house features, predicting the impact of SAT/GRE scores on college admissions, predicting the sales based on input parameters, predicting the weather, etc.

MLP Regressor

MLP Regressor implements a multi-layer perceptron (MLP) that trains using backpropagation with no activation function in the output layer, which can also be seen as using the identity function as activation function. Therefore, it uses the square error as the loss function, and the output is a set of continuous values. When it comes to regression, there is a little more to say about the MLP. As it turns out, the only thing that changes is the activation function for the final nodes in the network that produces predictions. They allow for a wide range of outputs, not just the output from a set of classes. All the issues and

hyperparameters are the same, as in the case of classification. Of course, in the regression context, you may end up making different choices than for classification. As well as MLPClassifier, there is its regressor sibling,MLPRegressor. The two share an almost identical interface.The main difference between the two is the loss functions used by each of them and the activation functions of the output layer. The regressor optimizes a squared loss, and the last layer is activated by an identity function. All other hyperparameters are the same, including the four activation options for the hidden layers.Both estimators have a partial_fit()method. You can use it to update the model once you get a hold of additional training data after the estimator has already been fitted.score()in MLPRegressorcalculates the regressor'sR2, as opposed to the
classifier's accuracy, which is calculated byMLPClassifier.

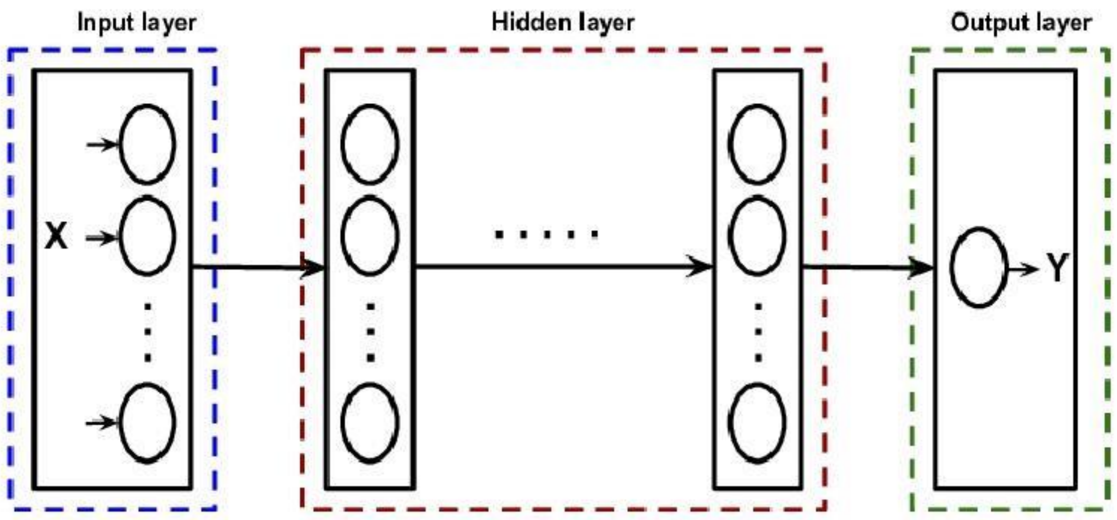


Figure -MLP Regression

4. DATA FLOW DIAGRAM

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

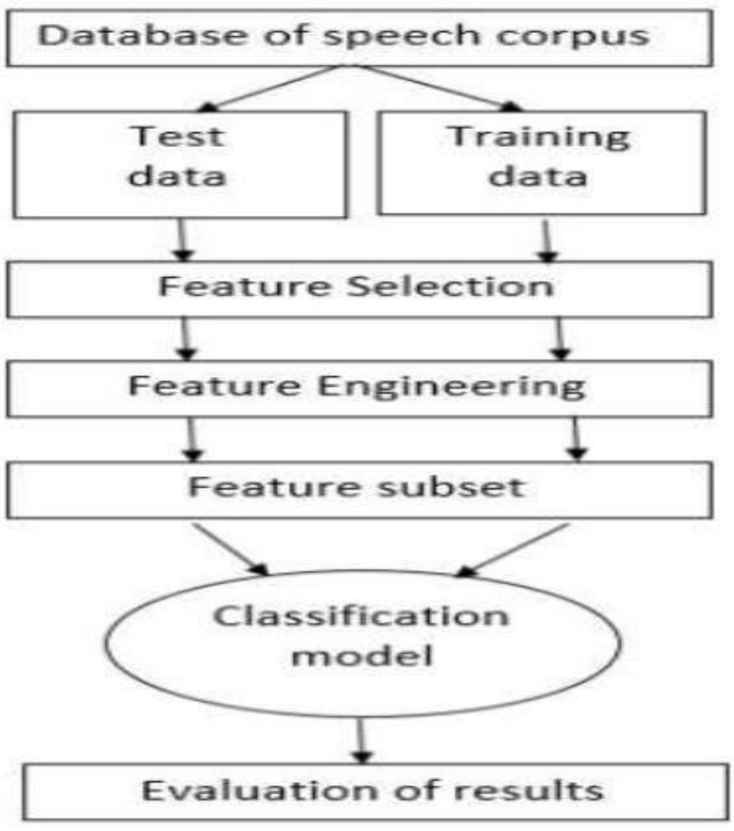


Figure – Data Flow Diagram

5. MODULES

- ➢ Speech input Module
- ➢ Feature extraction and selection
- ➢ Classification
- ➢ Recognized emotional output

MODULE DESCRIPTION 1. Speech input Module

Input to the system is speech taken with the help of audio. Then equivalent digital representation of received audio is produced through sound file.

2. Feature extraction and selection

There are so many emotional states of emotion and emotion relevance is used to select the extracted speech features. For speech feature extraction to selection corresponding to emotions all procedure revolves around the speech signal.

3. Classification Module

Finding a set of significant emotions for classification is the main concern in speech emotion recognition system. There are various emotional states contains in a typical set of emotions that makes classification a complicated task.

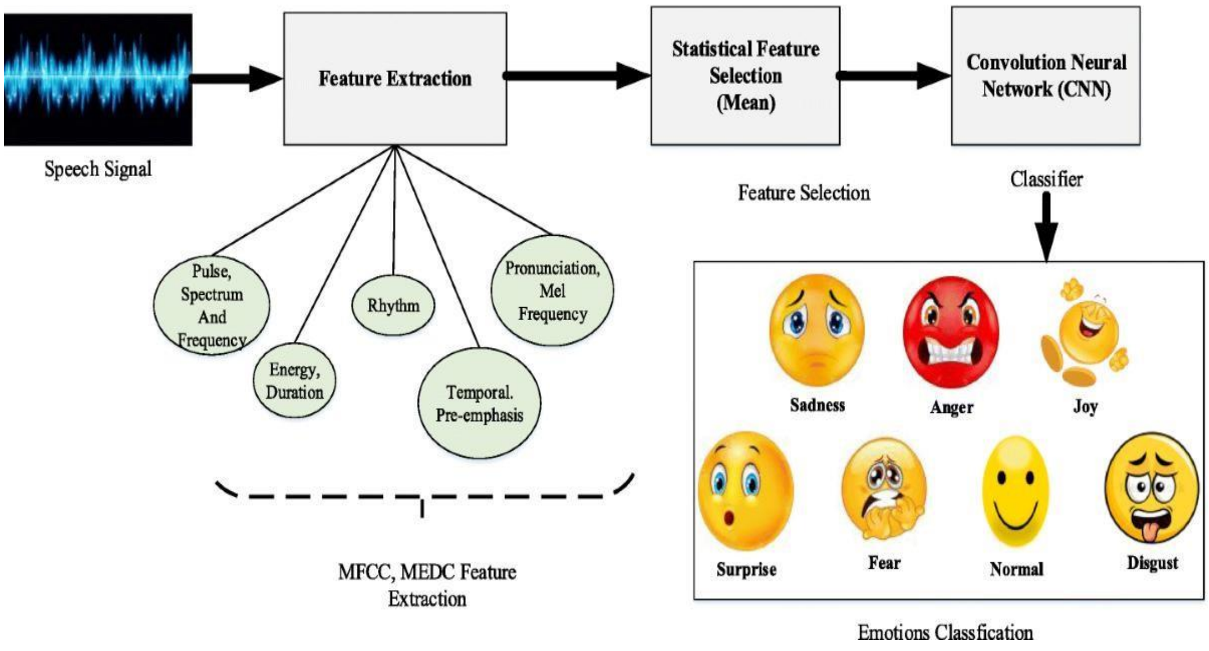
4. Recognized emotional output

Fear, surprise, anger, joy, disgust and sadness are primary emotions and naturalness of database level is the basis for speech emotion recognition system evaluation.

6. SYSTEM ARCHITECTURE

Describing the overall features of the software involves defining both the functional and non-functional requirements and establishing a high-level overview of the system. This stage sets the groundwork for architectural design, where the various web pages and their interconnections are identified and designed. The major software components are broken down into processing modules and conceptual data structures, with the interconnections among these modules clearly outlined.

The interconnections among these modules are critical for the system's cohesive functioning. The UI Module interacts with the Authentication, Content Management, Search, and User Interaction Modules to display relevant content and features. The Authentication Module works with the Security Module to manage user access and protect data. The Content Management Module communicates with the Database Management and Search Modules for content storage and retrieval. The User Interaction Module relies on the Backend Processing Module for managing user-generated data and interactions. Finally, the Security Module integrates with all other modules to enforce security policies and protect sensitive information.By defining these modules and their interconnections, the architectural design phase lays a solid foundation for detailed system development, ensuring that all components work together seamlessly to meet the specified requirements.



7. Design and Implementation

i. Frontend Code

index.html

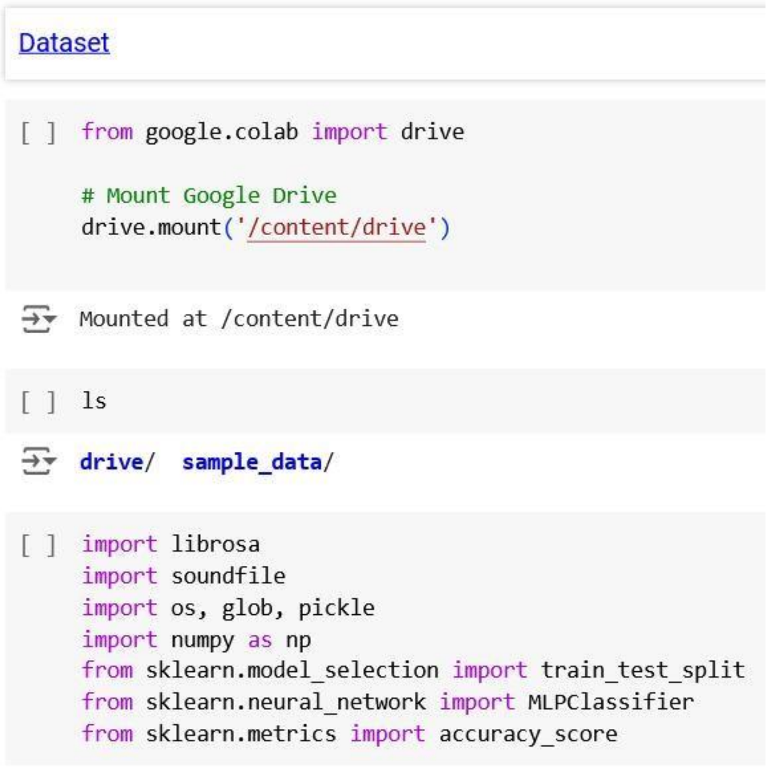
```
<!DOCTYPE html>
<html>
<head>
   <title>Emotion Prediction</title>
   <style>
      body {
         font-family: Arial, sans-serif; background-color: #f8f9fa; color: #333;
         margin: 0; display: flex; flex-direction: column; align-items: center;
         justify-content: flex-start; height: 100vh; padding-top: 40px; /* Added
         padding to move the content down a bit */
      } h1
      {
         color:    #007bff;    text-
         align: center;
      } form
      {
         background: #ffffff; padding: 20px; border-radius: 10px; box-shadow: 0 0
         10px rgba(0, 0, 0, 0.1); text-align: center; margin-top: 20px; /* Added
         margin to separate the form from the heading */
      }
      input[type="file"] {
      display: block;
      margin: 20px auto;
      font-size: 16px; }
      input[type="submit"]
      {
         background-color: #007bff;
      color: white; border: none;
      padding: 10px 20px; border-
```

```
        radius: 5px; cursor: pointer;
        font-size: 16px; }
        input[type="submit"]:hover {
          background-color: #0056b3;
        }
        #result { margin-
          top: 20px; font-
          size: 18px; color:
          #28a745;
        }
      </style>
    </head>
    <body>
      <h1>Emotion Prediction</h1>
      <form id="uploadForm" enctype="multipart/form-data">
        <input type="file" name="file" accept=".wav" required/>
        <br><br>
        <input type="submit" value="Upload and Predict"/>
      </form>
      <br>
      <div id="result"></div>
      <script> const form =
        document.getElementById('uploadForm'); const
        resultDiv = document.getElementById('result');

        form.addEventListener('submit', async (event) =>
          { event.preventDefault(); const formData = new
          FormData(form); const response = await
          fetch('/predict', {
            method:         'POST',
            body: formData
          }); const data = await response.json(); resultDiv.innerHTML =
          <h2>Prediction: ${data.prediction}</h2>;
        });
      </script>
    </body>
    </html>
```

app.py

```
from flask import Flask, request, jsonify, send_from_directory
import pickle import numpy as np import librosa

# Load the model
filename            =            'modelForPrediction1.sav'
loaded_model = pickle.load(open(filename, 'rb'))

# Function to extract features (you need to define this based on your
implementation) def extract_feature(file_name, mfcc, chroma, mel): X, sample_rate
= librosa.load(file_name) result = np.array([]) if mfcc:
        mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13).T, axis=0)
    result = np.hstack((result, mfccs)) if chroma:
        stft = np.abs(librosa.stft(X)) 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(y=X,   sr=sample_rate).T,   axis=0)
        result = np.hstack((result, mel))
    return result

# Emotion labels according to the RAVDESS dataset emotions
= {
    '01': 'neutral',
    '02': 'calm',
    '03': 'happy',
    '04': 'sad',
    '05': 'angry',
    '06': 'fearful',
    '07': 'disgust',
    '08': 'surprised'
}

app = Flask(_name_)

@app.route('/')          def          home():          return
send_from_directory('views', 'index.html')

@app.route('/predict',      methods=['POST'])
def predict():
    file = request.files['file']
    file_path = "temp.wav"
    file.save(file_path)

    # Extract features from the uploaded file feature =
    extract_feature(file_path, mfcc=True, chroma=True, mel=True) feature =
    feature.reshape(1, -1)
    # Make prediction prediction =
    loaded_model.predict(feature)
    print(prediction)

    # Map prediction to label emotion_label
    = prediction
    #emotion_label = emotions.get(str(prediction[0]).zfill(2), "Unknown emotion")
    prediction = prediction.tolist() return jsonify({'prediction': prediction})

if _name_ == "_main_": app.run(debug=True)
```

ii. Backend



Dataset

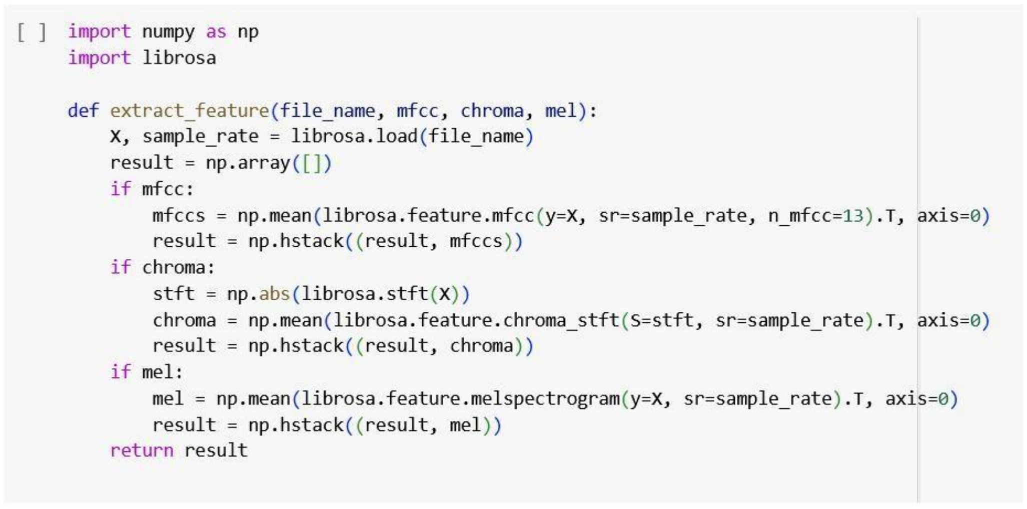
```
[ ] from google.colab import drive

    # Mount Google Drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

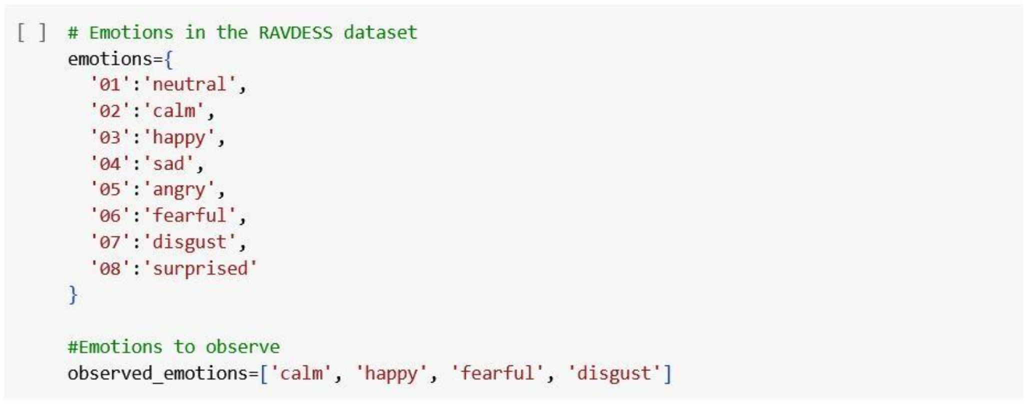
```
[ ] ls
```

```
drive/  sample_data/
```

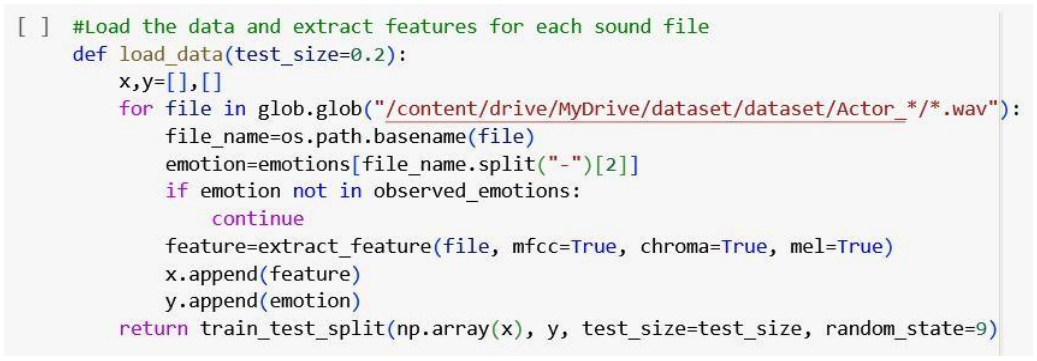
```
[ ] import librosa
    import soundfile
    import os, glob, pickle
    import numpy as np
    from sklearn.model_selection import train_test_split
    from sklearn.neural_network import MLPClassifier
    from sklearn.metrics import accuracy_score
```



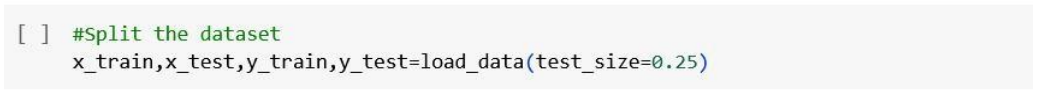
```
import numpy as np
import librosa

def extract_feature(file_name, mfcc, chroma, mel):
    X, sample_rate = librosa.load(file_name)
    result = np.array([])
    if mfcc:
        mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13).T, axis=0)
        result = np.hstack((result, mfccs))
    if chroma:
        stft = np.abs(librosa.stft(X))
        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(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, mel))
    return result
```



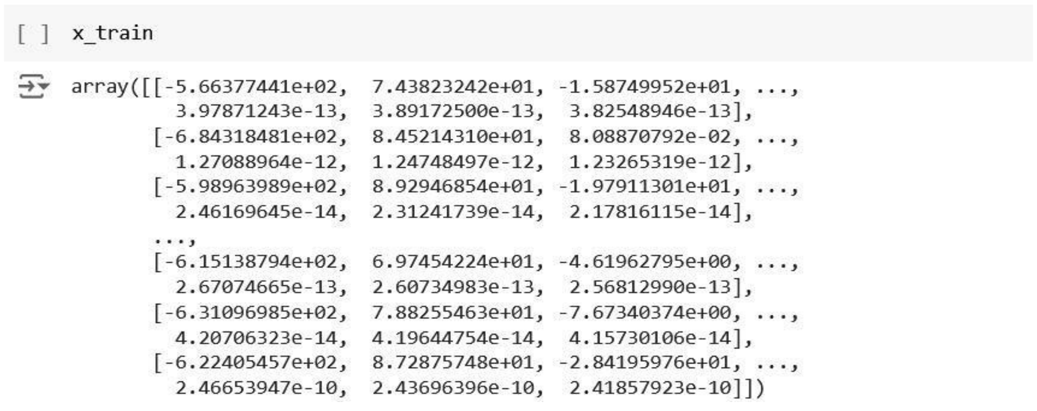
```
# Emotions in the RAVDESS dataset
emotions={
  '01':'neutral',
  '02':'calm',
  '03':'happy',
  '04':'sad',
  '05':'angry',
  '06':'fearful',
  '07':'disgust',
  '08':'surprised'
}

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



```
#Load the data and extract features for each sound file
def load_data(test_size=0.2):
    x,y=[],[]
    for file in glob.glob("/content/drive/MyDrive/dataset/dataset/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)
```

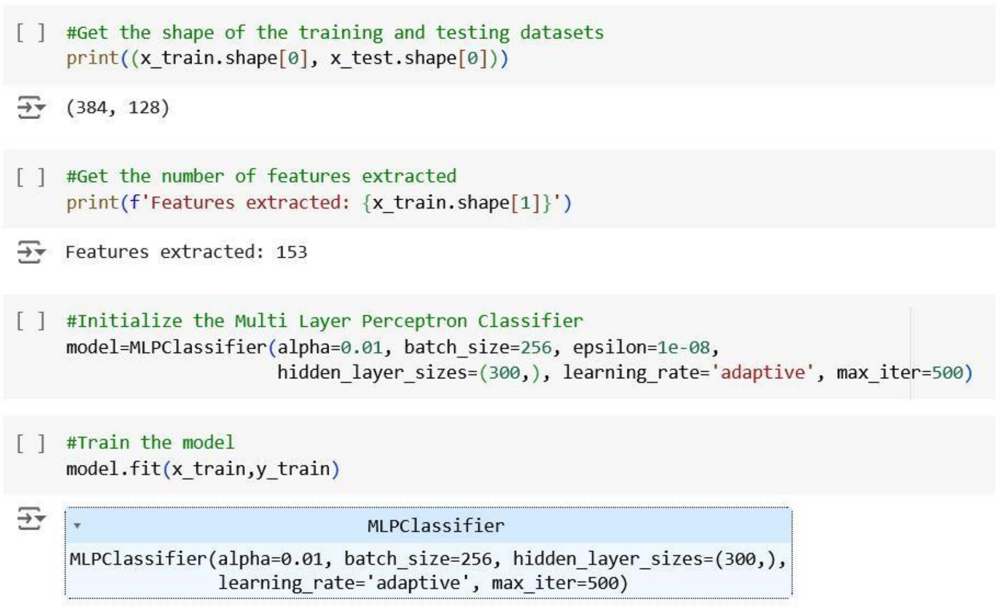


```
#Split the dataset
x_train,x_test,y_train,y_test=load_data(test_size=0.25)
```



```
x_train
```

```
array([[-5.66377441e+02,  7.43823242e+01, -1.58749952e+01, ...,
         3.97871243e-13,  3.89172500e-13,  3.82548946e-13],
       [-6.84318481e+02,  8.45214310e+01,  8.08870792e-02, ...,
         1.27088964e-12,  1.24748497e-12,  1.23265319e-12],
       [-5.98963989e+02,  8.92946854e+01, -1.97911301e+01, ...,
         2.46169645e-14,  2.31241739e-14,  2.17816115e-14],
       ...,
       [-6.15138794e+02,  6.97454224e+01, -4.61962795e+00, ...,
         2.67074665e-13,  2.60734983e-13,  2.56812990e-13],
       [-6.31096985e+02,  7.88255463e+01, -7.67340374e+00, ...,
         4.20706323e-14,  4.19644754e-14,  4.15730106e-14],
       [-6.22405457e+02,  8.72875748e+01, -2.84195976e+01, ...,
         2.46653947e-10,  2.43696396e-10,  2.41857923e-10]])
```



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

```
(384, 128)
```

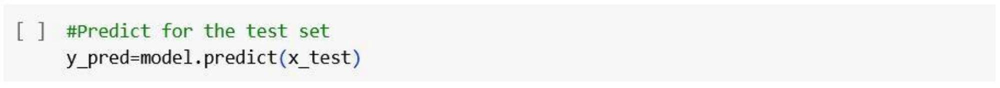
```
[ ] #Get the number of features extracted
    print(f'Features extracted: {x_train.shape[1]}')
```

```
Features extracted: 153
```

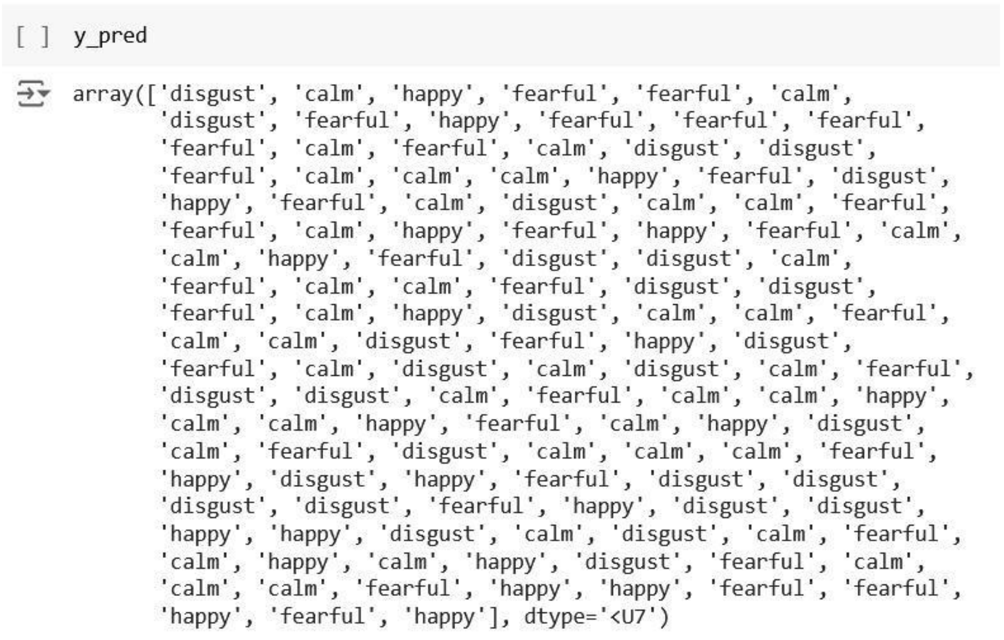
```
[ ] #Initialize the Multi Layer Perceptron Classifier
    model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08,
                        hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
```

```
[ ] #Train the model
    model.fit(x_train,y_train)
```

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

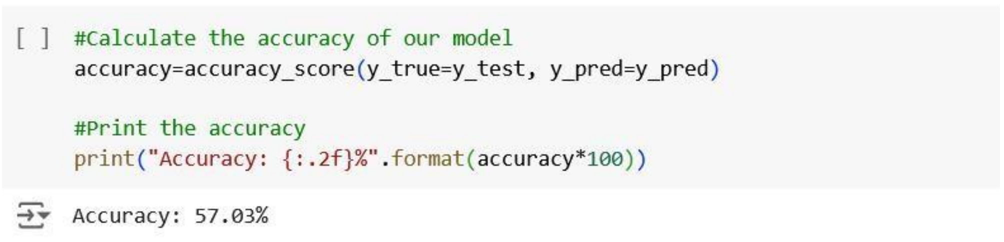


```
[ ] #Predict for the test set
    y_pred=model.predict(x_test)
```



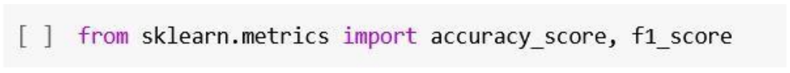
```
[ ] y_pred
```

```
array(['disgust', 'calm', 'happy', 'fearful', 'fearful', 'calm',
       'disgust', 'fearful', 'happy', 'fearful', 'fearful', 'fearful',
       'fearful', 'calm', 'fearful', 'calm', 'disgust', 'disgust',
       'fearful', 'calm', 'calm', 'calm', 'happy', 'fearful', 'disgust',
       'happy', 'fearful', 'calm', 'disgust', 'calm', 'calm', 'fearful',
       'fearful', 'calm', 'happy', 'fearful', 'happy', 'fearful', 'calm',
       'calm', 'happy', 'fearful', 'disgust', 'disgust', 'calm',
       'fearful', 'calm', 'calm', 'fearful', 'disgust', 'disgust',
       'fearful', 'calm', 'happy', 'disgust', 'calm', 'calm', 'fearful',
       'calm', 'calm', 'disgust', 'fearful', 'happy', 'disgust',
       'fearful', 'calm', 'disgust', 'calm', 'disgust', 'calm', 'fearful',
       'disgust', 'disgust', 'calm', 'fearful', 'calm', 'calm', 'happy',
       'calm', 'calm', 'happy', 'fearful', 'calm', 'happy', 'disgust',
       'calm', 'fearful', 'disgust', 'calm', 'calm', 'calm', 'fearful',
       'happy', 'disgust', 'happy', 'fearful', 'disgust', 'disgust',
       'disgust', 'disgust', 'fearful', 'happy', 'disgust', 'disgust',
       'happy', 'happy', 'disgust', 'calm', 'disgust', 'calm', 'fearful',
       'calm', 'happy', 'calm', 'happy', 'disgust', 'fearful', 'calm',
       'calm', 'calm', 'fearful', 'happy', 'happy', 'fearful', 'fearful',
       'happy', 'fearful', 'happy'], dtype='<U7')
```

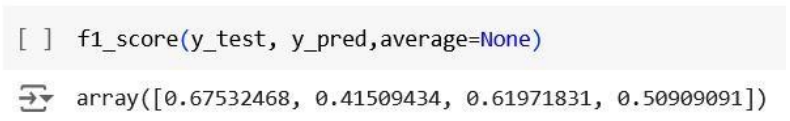


```
[ ] #Calculate the accuracy of our model
    accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

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

```
Accuracy: 57.03%
```

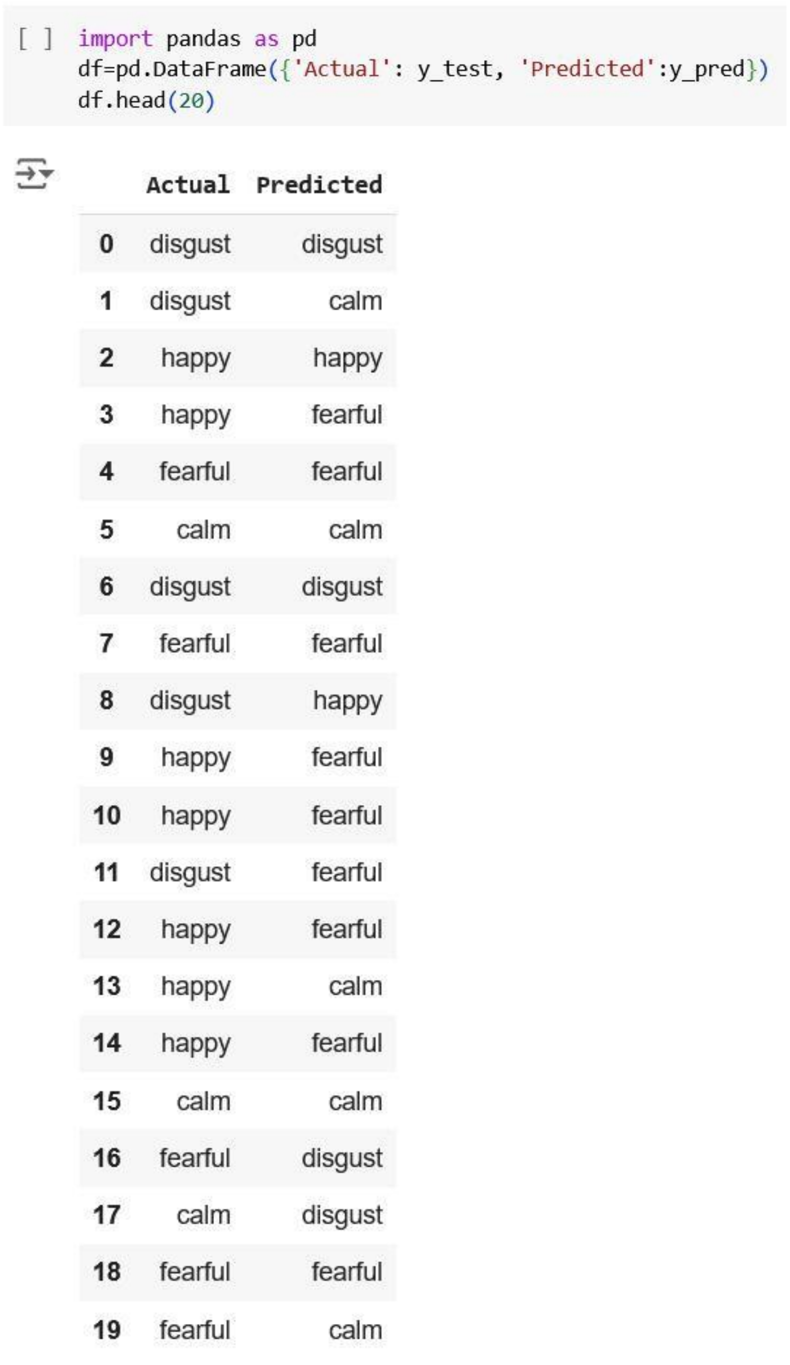


```
[ ] from sklearn.metrics import accuracy_score, f1_score
```



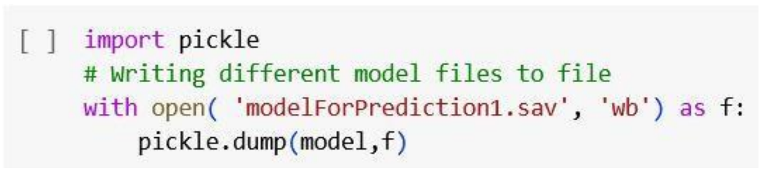
```
[ ] f1_score(y_test, y_pred,average=None)
```

```
array([0.67532468, 0.41509434, 0.61971831, 0.50909091])
```

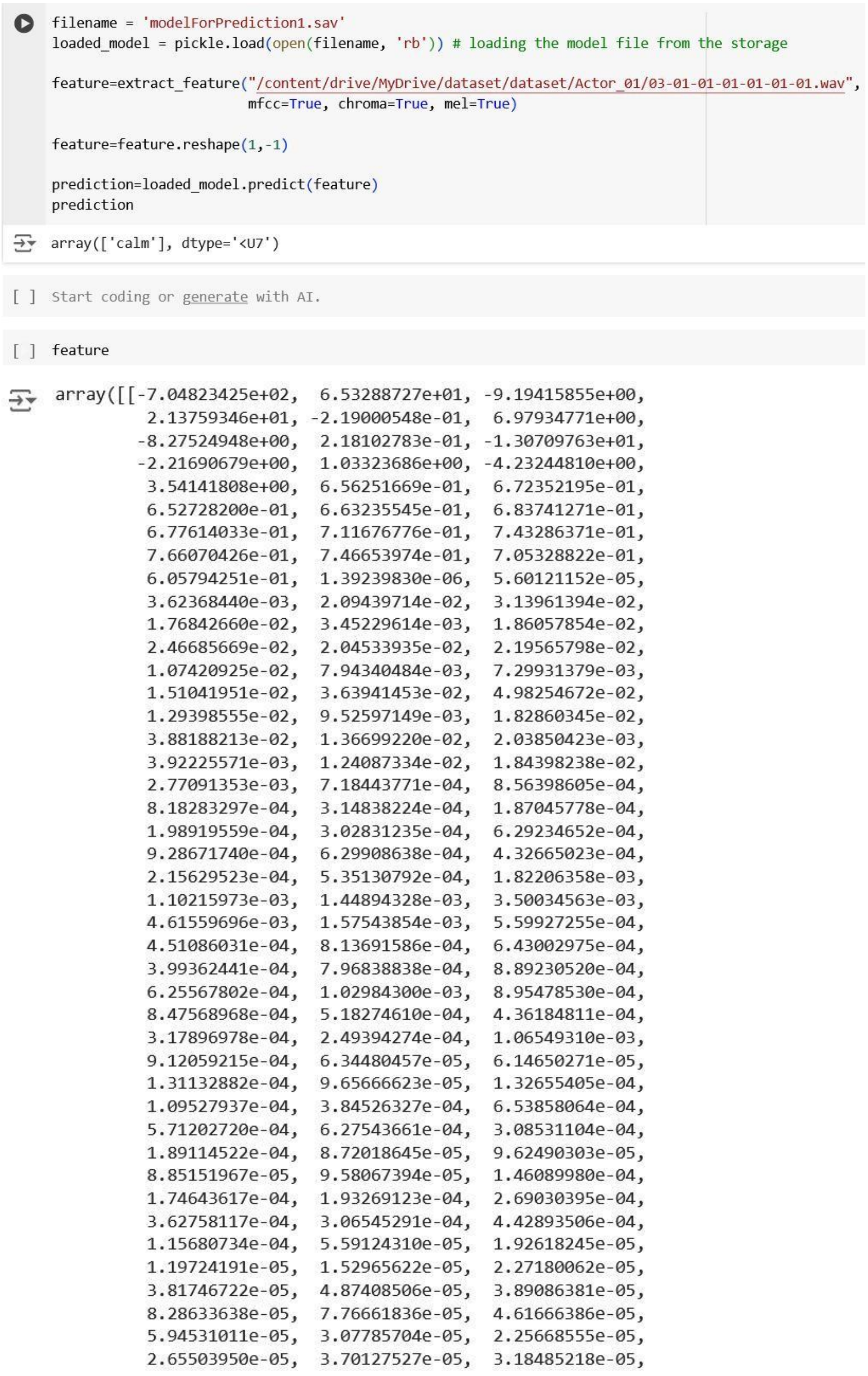


```
[ ] import pandas as pd
    df=pd.DataFrame({'Actual': y_test, 'Predicted':y_pred})
    df.head(20)
```

	Actual	Predicted
0	disgust	disgust
1	disgust	calm
2	happy	happy
3	happy	fearful
4	fearful	fearful
5	calm	calm
6	disgust	disgust
7	fearful	fearful
8	disgust	happy
9	happy	fearful
10	happy	fearful
11	disgust	fearful
12	happy	fearful
13	happy	calm
14	happy	fearful
15	calm	calm
16	fearful	disgust
17	calm	disgust
18	fearful	fearful
19	fearful	calm



```
[ ] import pickle
    # Writing different model files to file
    with open( 'modelForPrediction1.sav', 'wb') as f:
        pickle.dump(model,f)
```



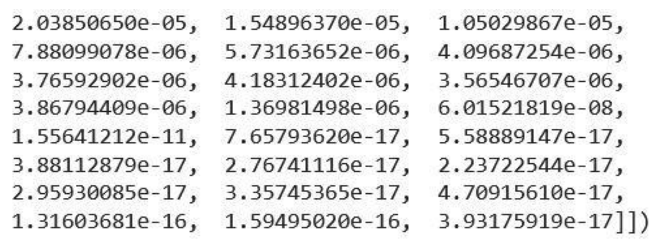
```
filename = 'modelForPrediction1.sav'
loaded_model = pickle.load(open(filename, 'rb')) # loading the model file from the storage

feature=extract_feature("/content/drive/MyDrive/dataset/dataset/Actor_01/03-01-01-01-01-01-01.wav",
                        mfcc=True, chroma=True, mel=True)

feature=feature.reshape(1,-1)

prediction=loaded_model.predict(feature)
prediction
```

```
array(['calm'], dtype='<U7')
```

```
[ ] Start coding or generate with AI.
```

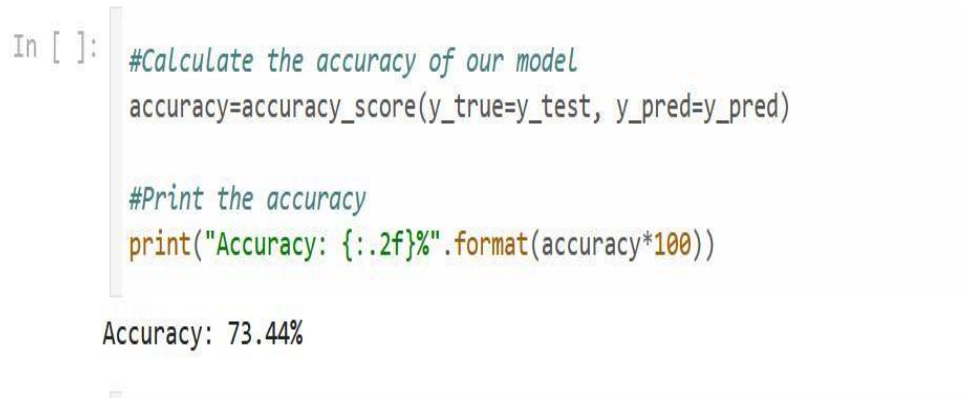
```
[ ] feature
```

```
array([[-7.04823425e+02,  6.53288727e+01, -9.19415855e+00,
         2.13759346e+01, -2.19000548e-01,  6.97934771e+00,
        -8.27524948e+00,  2.18102783e-01, -1.30709763e+01,
        -2.21690679e+00,  1.03323686e+00, -4.23244810e+00,
         3.54141808e+00,  6.56251669e-01,  6.72352195e-01,
         6.52728200e-01,  6.63235545e-01,  6.83741271e-01,
         6.77614033e-01,  7.11676776e-01,  7.43286371e-01,
         7.66070426e-01,  7.46653974e-01,  7.05328822e-01,
         6.05794251e-01,  1.39239830e-06,  5.60121152e-05,
         3.62368440e-03,  2.09439714e-02,  3.13961394e-02,
         1.76842660e-02,  3.45229614e-03,  1.86057854e-02,
         2.46685669e-02,  2.04533935e-02,  2.19565798e-02,
         1.07420925e-02,  7.94340484e-03,  7.29931379e-03,
         1.51041951e-02,  3.63941453e-02,  4.98254672e-02,
         1.29398555e-02,  9.52597149e-03,  1.82860345e-02,
         3.88188213e-02,  1.36699220e-02,  2.03850423e-03,
         3.92225571e-03,  1.24087334e-02,  1.84398238e-02,
         2.77091353e-03,  7.18443771e-04,  8.56398605e-04,
         8.18283297e-04,  3.14838224e-04,  1.87045778e-04,
         1.98919559e-04,  3.02831235e-04,  6.29234652e-04,
         9.28671740e-04,  6.29908638e-04,  4.32665023e-04,
         2.15629523e-04,  5.35130792e-04,  1.82206358e-03,
         1.10215973e-03,  1.44894328e-03,  3.50034563e-03,
         4.61559696e-03,  1.57543854e-03,  5.59927255e-04,
         4.51086031e-04,  8.13691586e-04,  6.43002975e-04,
         3.99362441e-04,  7.96838838e-04,  8.89230520e-04,
         6.25567802e-04,  1.02984300e-03,  8.95478530e-04,
         8.47568968e-04,  5.18274610e-04,  4.36184811e-04,
         3.17896978e-04,  2.49394274e-04,  1.06549310e-03,
         9.12059215e-04,  6.34480457e-05,  6.14650271e-05,
         1.31132882e-04,  9.65666623e-05,  1.32655405e-04,
         1.09527937e-04,  3.84526327e-04,  6.53858064e-04,
         5.71202720e-04,  6.27543661e-04,  3.08531104e-04,
         1.89114522e-04,  8.72018645e-05,  9.62490303e-05,
         8.85151967e-05,  9.58067394e-05,  1.46089980e-04,
         1.74643617e-04,  1.93269123e-04,  2.69030395e-04,
         3.62758117e-04,  3.06545291e-04,  4.42893506e-04,
         1.15680734e-04,  5.59124310e-05,  1.92618245e-05,
         1.19724191e-05,  1.52965622e-05,  2.27180062e-05,
         3.81746722e-05,  4.87408506e-05,  3.89086381e-05,
         8.28633638e-05,  7.76661836e-05,  4.61666386e-05,
         5.94531011e-05,  3.07785704e-05,  2.25668555e-05,
         2.65503950e-05,  3.70127527e-05,  3.18485218e-05,
```

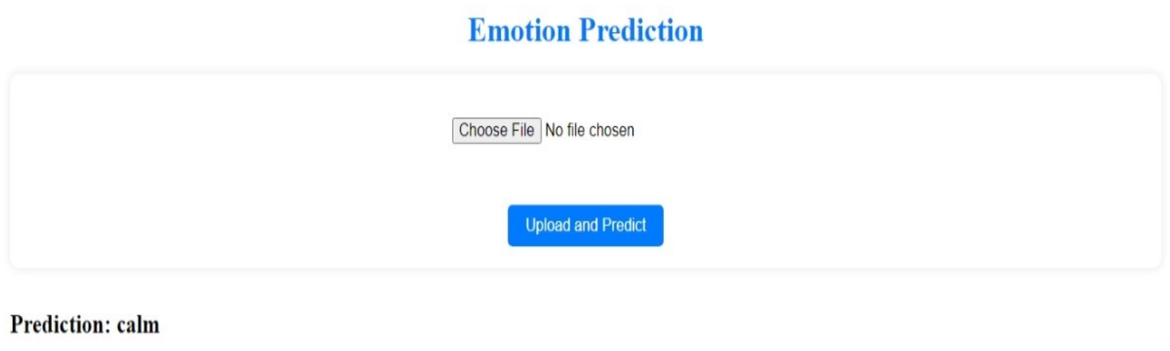


```
2.03850650e-05,  1.54896370e-05,  1.05029867e-05,
7.88099078e-06,  5.73163652e-06,  4.09687254e-06,
3.76592902e-06,  4.18312402e-06,  3.56546707e-06,
3.86794409e-06,  1.36981498e-06,  6.01521819e-08,
1.55641212e-11,  7.65793620e-17,  5.58889147e-17,
3.88112879e-17,  2.76741116e-17,  2.23722544e-17,
2.95930085e-17,  3.35745365e-17,  4.70915610e-17,
1.31603681e-16,  1.59495020e-16,  3.93175919e-17]])
```

OUTPUT



```
In [ ]:  #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))

Accuracy: 73.44%
```



Emotion Prediction

Prediction: calm

8. RESULTS

The speech emotion recognition of human voice using machine learning is successfully obtained. By using just small amount of dataset, **73.44%** accuracy has been obtained. If more dataset are used then accuracy will also increase. It is seen that speech emotion recognition using MLP is very efficient and easy to implement.

9. DISCUSSION AND CHALLENGES

Discussed the various challenges faced in accurately recognizing emotions from speech, such as variability in speech patterns, cultural differences, and the subjective nature of emotions.

Explored different feature extraction methods used in SER, such as Mel-Frequency Cepstral Coefficients (MFCCs), Prosodic features, and Deep Learning-based representations, discussing their effectiveness and limitations.

Discussed the importance of standardized datasets for benchmarking SER algorithms and techniques, and explore popular datasets like the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) and the Berlin Database of Emotional Speech.

Examined how cultural differences impact the recognition of emotions from speech and explore strategies for developing culturally robust SER systems.

Explored the various real-world applications of SER, such as in human-computer interaction, customer service, mental health monitoring, and virtual assistants, discussing both current applications and potential future developments.

Discussed the ethical implications of SER technology, including concerns related to privacy, consent, bias, and potential misuse of emotion recognition systems.

Explored the benefits of combining speech with other modalities, such as facial expressions and physiological signals, for more robust emotion recognition systems. Discuss the vulnerability of SER systems to adversarial attacks and explore potential defense mechanisms to mitigate these attacks.

CHALLENGES

Many people can display their emotions in various ways, and they may react differently emotionally to the same stimulus. It is challenging because of the variety in emotional expression. Noise in speech signals is a common problem that can make emotion identification models less accurate. Moreover, the quality of the speech stream may be impacted by signal distortion brought on by elements like reverberation and compression. For recognising voice emotions, a variety of machine learning techniques and deep learning architectures can be applied. Finding the ideal model for a given dataset can be difficult.

10. CONCLUSION

Various experiments are conducted by changing the several parameters like dimension of the model, number of epochs, changing the partition ratio between training and test data set. Different accuracy was found for different experiments. A lighter CNN architecture with 75% of training data and 25% of test data gave good result compared to deeper CNN architecture while classifying among ten classes. The accuracy of this model was found to be 69%. The performance of deeper CNN model was found to be excellent when the classification was among two classes the rationale is there have been a greater number of coaching samples available to classify among two classes. When an equivalent model was wont to classify among ten classes the training dataset was divided into ten labels this led to the less number of coaching samples available for each class. This led us to the poor accuracy of the model. With the greater number of coaching samples available for every class and with the assistance of GPU's to hurry up the training process more accuracy are often achieved in the future enhancements.

11. FUTURE SCOPE

The project can be extended to integrate with the robot to help it to have a better understanding of the mood the corresponding human is in, which will help it to have a better conversation as well as it can be integrated with various music applications to recommend songs to its users according to his/her emotions, it can also be used in various online shopping applications such as Amazon to improve the product recommendation for its users. Moreover, in the upcoming years we can construct a sequence to sequence model to create voice having different emotions. E.g. sad voice, an excited one etc.

Human-Computer Interaction: SER can enhance human-computer interaction by enabling systems to adapt their responses based on the emotional state of the user. This could lead to more intuitive and personalized user experiences in applications like virtual assistants, customer service bots, and gaming.

Mental Health Monitoring: SER technology can be integrated into mental health monitoring systems to assess individuals' emotional states based on their speech patterns. This could aid in early detection of mood disorders such as depression and anxiety, allowing for timely intervention and support.

Market Research and Customer Feedback Analysis: Companies can utilize SER to analyze customer feedback from call center recordings, surveys, and social media interactions. Understanding the emotional tone of customer responses can provide valuable insights into consumer preferences, satisfaction levels, and brand sentiment.

Education and Learning: SER can be used in educational settings to assess students' engagement levels and emotional responses during lectures or online courses. Adaptive learning systems could adjust their content and teaching style based on real-time feedback, enhancing the learning experience.

Healthcare: In healthcare, SER can aid in diagnosing neurological conditions such as Parkinson's disease or Alzheimer's by detecting changes in speech patterns characteristic of these disorders. It can also assist in monitoring patients' emotional well-being during telehealth consultations or therapy sessions.

Smart Assistive Technologies: Integration of SER into smart home devices and wearable technology can enable more responsive and empathetic assistance for individuals with disabilities or special needs. These devices can detect distress or frustration in the user's voice and provide appropriate support or assistance.

Emotionally Intelligent AI: Advancements in SER could contribute to the development of emotionally intelligent artificial intelligence systems capable of understanding and appropriately responding to human emotions. Such systems could enhance communication and collaboration between humans and machines in various contexts.

Security and Surveillance: SER can be employed in security and surveillance applications to analyze audio data for suspicious or threatening behavior based on the emotional content of speech. This could help enhance public safety and security measures.

12. REFERENCES

1. Awni Hannun, Ann Lee, Qjantong Xu and Ronan Collobert, Sequence to sequence speech recognition with time-depth deperable convolutions, interspeech 2019, Sep 2019.

2. Lawrence R Rabiner Ronald W Schafer, "Introduction to Digital Speech Processing", Vol. 1, Nos. 1–2 (2007) 1–194, 2007 L. R. Rabiner and R. W... Schafer.

3. Li, J., Deng, L., Gong, Y. (2014). An Overview of Noise-Robust Automatic Speech Recognition, IEEE/ACM Transactions on Audio Speech & Language Processing, Vol.22, No.4, pp.745-777.

4. Jinyu Li, Li Deng, Yifan Gong, and Reinhold Hach- Umbach "An overview of noiserobust sutomatic speech recognition" IEEE/ACM Transactions on Audio, Speech and Language Processing, Vol 22, no 4, pp 745-777, 2014.

5. Chang, A X., Martini, B and Culurciello E (2015) 'Recurrent Neural Networks hardware implementation on FPGA', arXiv preprint arXiv:1511.05552.