Department of Computer Science

Pune University

(Degree Project - 1)

Project Name: Speech Emotion Recognition

Name : Jayesh Ahire (21111001**)**

Roshan Avhad (21111006**)**

Project Aim : To Get Emotion from the audio File.

INTRODUCTION

Speech Emotion Recognition (SER) is the task of recognizing the emotional aspects of speech irrespective of the semantic contents. While humans can efficiently perform this task as a natural part of speech communication, the ability to conduct it automatically using programmable devices is still an ongoing subject of research.

Studies of automatic emotion recognition systems aim to create efficient, real-time methods of detecting the emotions of mobile phone users, call center operators and customers, car drivers. pilots, and many other human-machine communication users. Adding emotions to machines has been recognized as a critical factor in making machines appear and act in a human-like manner Robots capable of understanding emotions could provide appropriate emotional responses and exhibit emotional personalities. In some circumstances. humans could be replaced by computer-generated characters having the ability to conduct very natural and convincing conversations by appealing to human emotions. Machines need to understand emotions conveyed by speech. Only with this capability, an entirely meaningful dialogue based on mutual human-machine trust and understanding can be achieved.

DATABASES USED FOR SER :

Speech emotional databases are used by many researchers in a variety of research activities.

The quality of the databases utilized and the performance achieved are the most important factors in the evaluation of emotion recognition. The methods available and objectives in the collection of speech databases vary depending on the motivation for speech systems development.

The categorization of databases can also be described as:

Simulated database: In these databases, the speech data has been recorded by well-trained and experienced performers .Among all databases, this one is considered the simplest way

to obtain the speech-based dataset on various emotions It is considered that almost 60% of speech databases are gathered by this technique

Induced database: This is another type of database in which the emotional set is collected by creating an artificial emotional situation This is done without the knowledge of the performer or speaker. As compared to an actor-based database, this is a more naturalistic database. However, an issue of ethics may apply, because the speaker should know that they have

been recorded for research-based activities.

Natural database: While most realistic, these databases are hard to obtain due to the difficulty in recognition Natural emotional speech databases.

Here , **tess** dataset is used .

TRADITIONAL TECHNUIQUES OF SER

An emotion recognition system based on digitized speech is comprised of three fundamental components signal preprocessing feature extraction and classification 251. Acoustic preprocessing such as denoising as well as segmentation is carried out to determine meaningful units of this signal [26]. feature extraction is utilized to identify the rare event feature available in the signal. Lastly, the mapping of extracted feature vectors to relevant emotion is carried out by classifiers. In this section, a detailed discussion of speech signal processing, feature extraction, and classification is provided . Also, the differences between spontaneous and acted speech are discussed due to their relevance to the topic In the first stage of speech-based signal processing, speech enhancement is carried out where the noisy components are removed. The second stage involves two parts, feature extraction, an feature selection. The required features are extracted from the pre-processed speech signal and the selection is made from.

NEED FOR DEEP LEARNING TECHNIQUES FOR SER

Speech processing usually functions in a straightforward manner on an audio signal . It Is considered significant and necessary for various speech-based applications such as SER, speech denoising and music classification.

With recent advancements, SER has gained significance. However, it still requires accurate methodologies to mimic human-like behavior for interaction with human beings. As discussed earlier, an SER system is made up of various components that include feature selection and extraction, feature classification, acoustic modeling, recognition per unit, and most importantly language-based modeling. The traditional SER systems typically incorporate various classification models such as GMMs and HMMs. The GMMs are utilized for illustration of acoustic features of sound units, while, the HMMs are utilized for dealing with temporal variations occurrence in speech signals.

Deep learning methods are comprised of various nonlinear components that perform computation on a parallel basis . However, these methods need to be structured with deeper layers of architecture to overcome the limitations of other techniques. Deep learning techniques such as Deep Boltzmann Machine (DBM), Recurrent Neural Network (RNN), Recursive Neural Network (RNN), Deep Belief Network (DBN), Convolutional Neural Networks (CNN) and Auto Encoder (A) are considered a few of the fundamental deep learning techniques used for SER, that significantly improves the overall performance of the designed system

PROBLEM DEFINITION

These methods required enormous engineering features and any variation in the features would need re-modeling the overall architecture of the technique. Nevertheless, recent development in deep learning applications and methods for Search Emotion Recognition can be varied also.

There are numerous literature and studies on the application of these algorithms to understand emotions and state of mind from human speech. Additionally, to deep learning, neural networks, and application of improvements of long short-term memory (LSTM) networks, generative adversarial models, and lots more, a wave in research on speech emotion recognition and its application now emerges. It is essential to understand its application and its role in emotion. For this reason, the objective of the current paper is to understand deep learning techniques for speech emotion recognition, from databases to models. After applying the deep learning and feature extraction methods further, it is very difficult to obtain high accuracy in the model because of the similarities between the different emotions like happy and surprising emotions have the same kind of frequency and tone. The length of the voice is also a problem because we all know that the human emotions do not remain the same throughout the sentence it keeps on changing so the system has to identify the parts of the data to understand the full emotion of the voice.

OBJECTIVE

This article looked at how you can use speech data in real-world applications, including automatic speech recognition (ASR) and speech emotion recognition (SER). We explored open-source Python packages to help start ASR and suggested project ideas. We also took a deeper dive into building a robust SER model using the TESS (TORONTO EMOTION SPEECH SET) dataset to train an LSTM model. This hands-on experience will equip you to start building projects and master the concepts of SER.

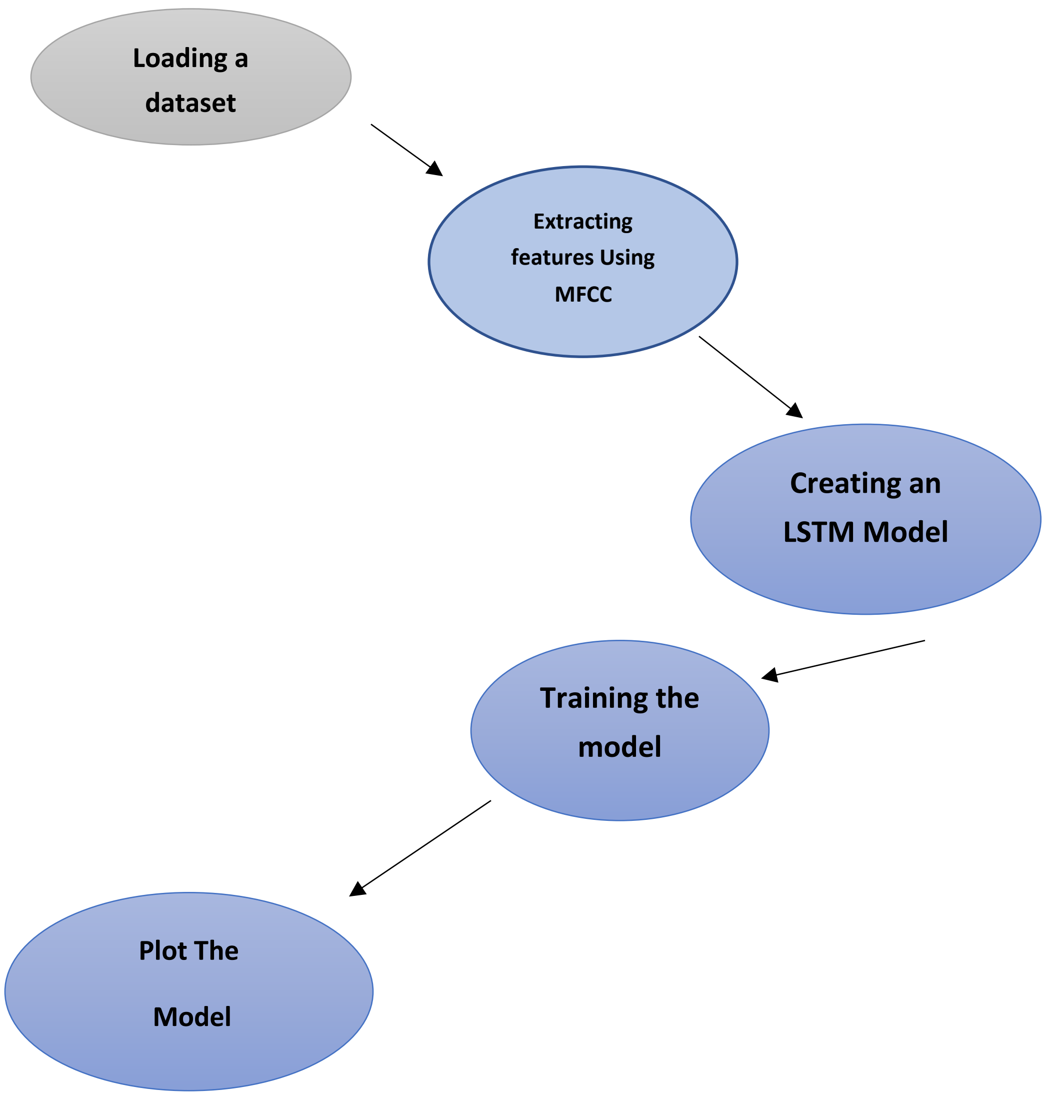
Scientists apply various audio processing techniques to capture this hidden layer of information that can amplify and extract tonal and acoustic features from speech Converting audio signals into numeric or vector format is not as straightforward as images. The transformation method will determine how much pivotal information is retained when we abandon the "audio" format.

If a particular data transformation cannot capture the softness and calmness, it would be challenging for the models to learn the emotion and classify the sample

Some methods to transform audio data into numeric include Mel Spectrograms that visualize audio signals based on their frequency components which can be plotted as an audio wave and fed to train a CNN as an image classifier. We can capture this using Mel-frequency cepstral coefficients (MCCs). Each of these data formats has its benefits and disadvantages based on the application.

We will try to obtain the data from the MFCC and plot the data in a suitable array form that is used by the model for example we are using here the LSTM model of feature recognition we will use the numeric values given by the MFCC as input to the LSTM model and will try to recognize the emotion.

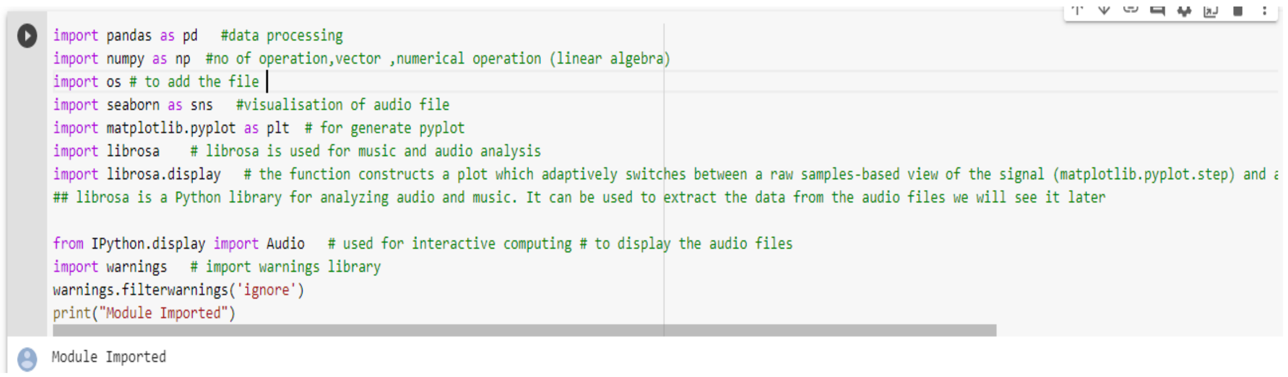
Design Flow/Process :



LOADING THE DATASET:

There are set of 200 target words were spoken in the carrier phrase "Say the word 'by two actresses (aged 26 and 64 years) and recordings were made of the set portraying each of seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral). There are 2800 data points (audio files) in total.

The dataset is organized such that each of the two female actor and their emotions are contain within its own folder. And within that, all 200 target words audio file can be found. The format of the audio file is in WAV format.

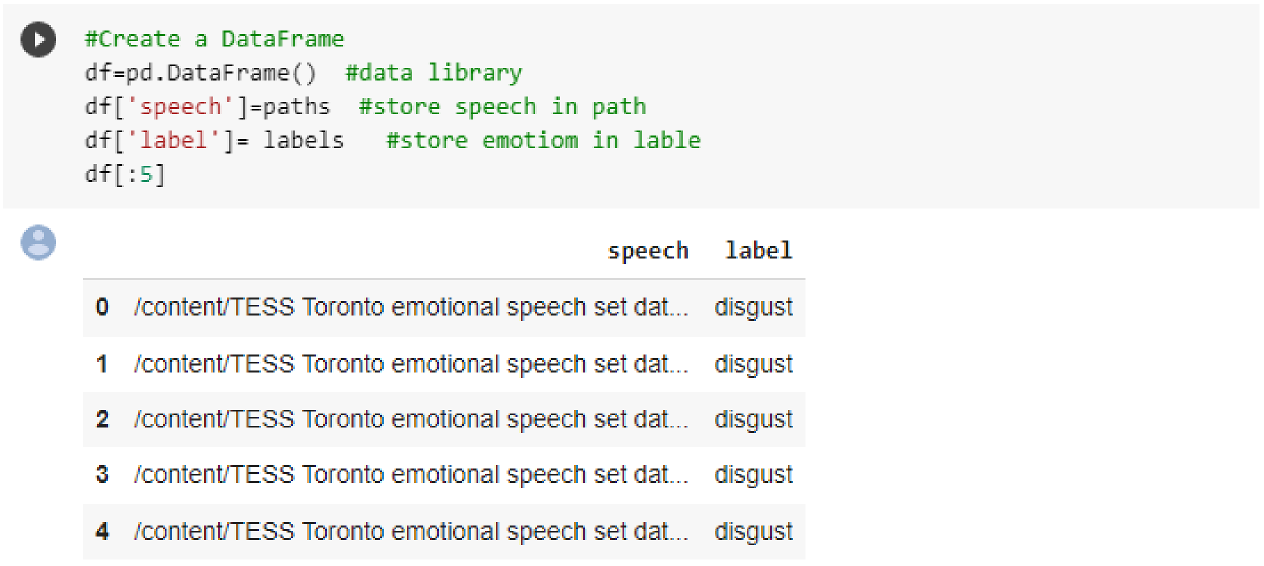


```
import pandas as pd   #data processing
import numpy as np  #no of operation,vector ,numerical operation (linear algebra)
import os # to add the file
import seaborn as sns   #visualisation of audio file
import matplotlib.pyplot as plt  # for generate pyplot
import librosa    # librosa is used for music and audio analysis
import librosa.display   # the function constructs a plot which adaptively switches between a raw samples-based view of the signal (matplotlib.pyplot.step) and
## librosa is a Python library for analyzing audio and music. It can be used to extract the data from the audio files we will see it later

from IPython.display import Audio   # used for interactive computing # to display the audio files
import warnings   # import warnings library
warnings.filterwarnings('ignore')
print("Module Imported")
```

```
Module Imported
```

First of All we have imported some libraries:

The first five i.e pandas, numpy, os, seaborn, and mathplotlib are for **visualization purpose**.

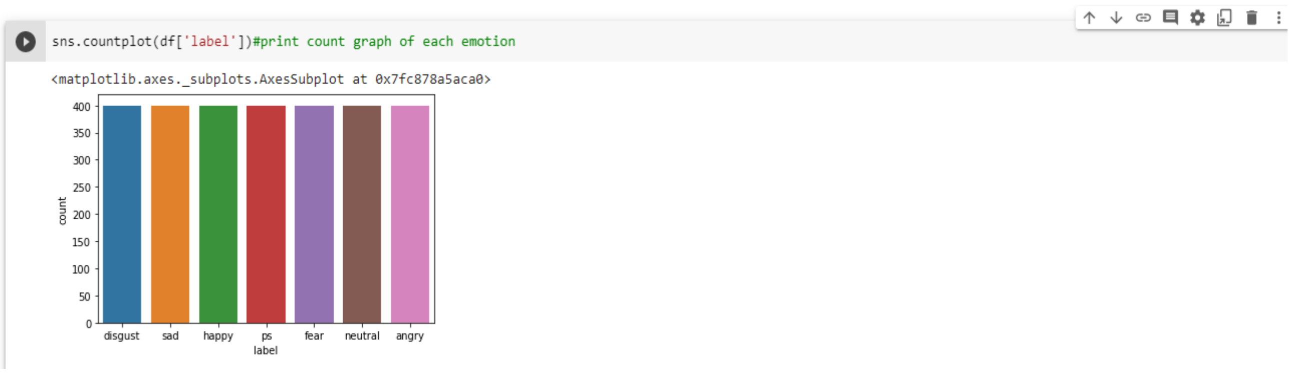


```
#Create a DataFrame
df=pd.DataFrame()  #data library
df['speech']=paths  #store speech in path
df['label']= labels   #store emotiom in lable
df[:5]
```

	speech	label
0	/content/TESS Toronto emotional speech set dat...	disgust
1	/content/TESS Toronto emotional speech set dat...	disgust
2	/content/TESS Toronto emotional speech set dat...	disgust
3	/content/TESS Toronto emotional speech set dat...	disgust
4	/content/TESS Toronto emotional speech set dat...	disgust

Here we have created a data frame for paths and labels

The total no samples use for each emotion i.e fear, anger ,disgust, neutral, sad, ps, and happy are 400. Total 2800 samples



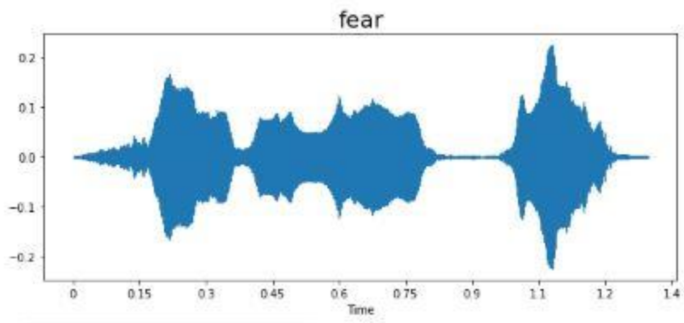
```
sns.countplot(df['label'])#print count graph of each emotion
<matplotlib.axes._subplots.AxesSubplot at 0x7fc878a5aca0>
```

No of sample present of each dataset

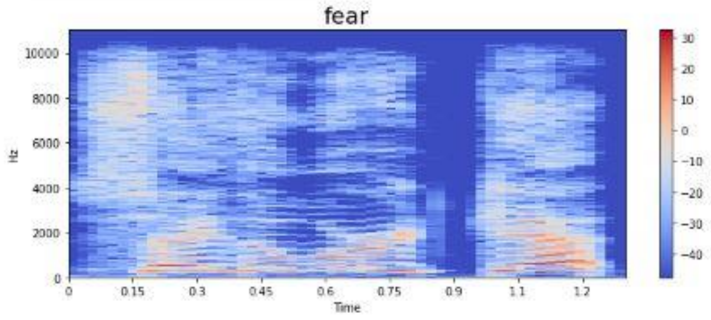
Here are some images that are showing different waveforms and spectrograms of each emotion:

<u>These are done only for visualization of spectrograms and waveforms</u>

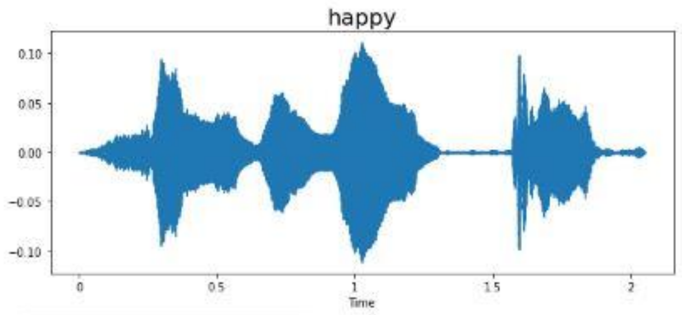
Fear:


fear
0.2
0.1
0.0
−0.1
−0.2
0
0.15
0.3
0.45
0.6
0.75
0.9
1.1
1.2
1.4
Time

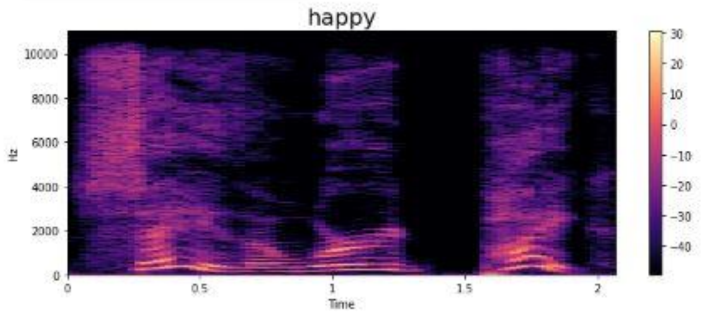
0:00 / 0:01


fear
Hz
10000
8000
6000
4000
2000
0
0
0.15
0.3
0.45
0.6
0.75
0.9
1.1
1.2
Time
30
20
10
0
−10
−20
−30
−40

Happy :


happy
0.10
0.05
0.00
−0.05
−0.10
0
0.5
1
1.5
2
Time

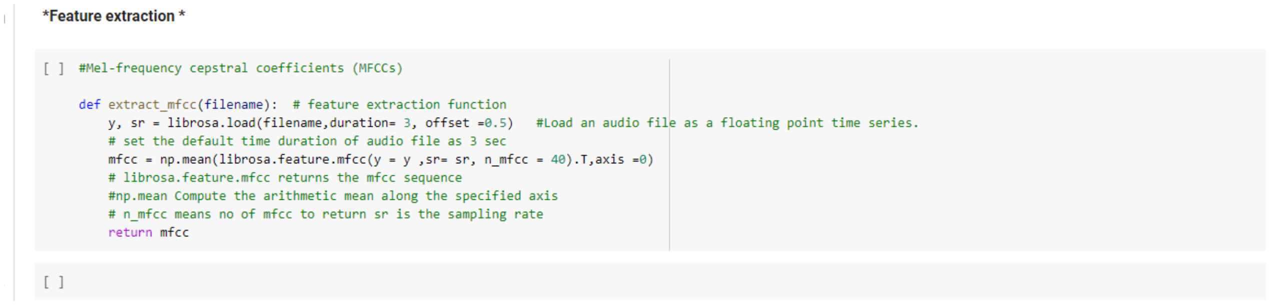
0:00 / 0:02


happy
Hz
10000
8000
6000
4000
2000
0
0
0.5
1
1.5
2
Time
30
20
10
0
−10
−20
−30
−40

Extracting features Using MFCC :

Feature extraction is the process of extracting various acoustic features of the speech which can directly affect the accuracy of the classification results, acoustic features include physical aspects of spoken language that can be recorded and analyzed these include waveform analysis. FFT or LPC analysis, voice onset time . format frequency. measurements and so on For the feature extraction purpose various methods are used such as MFCC, LPC, LPCC, LSF, PLP, DWT. We are using the MFCC method in our model for the process of feature extraction.

So Mel Frequency Cepstral Coefficient (MFCC) is basically one of the methods used to extract the various acoustic features of a speech from a raw data to perform various things on the extracted data which can be further used for the classification of the emotions of a speech

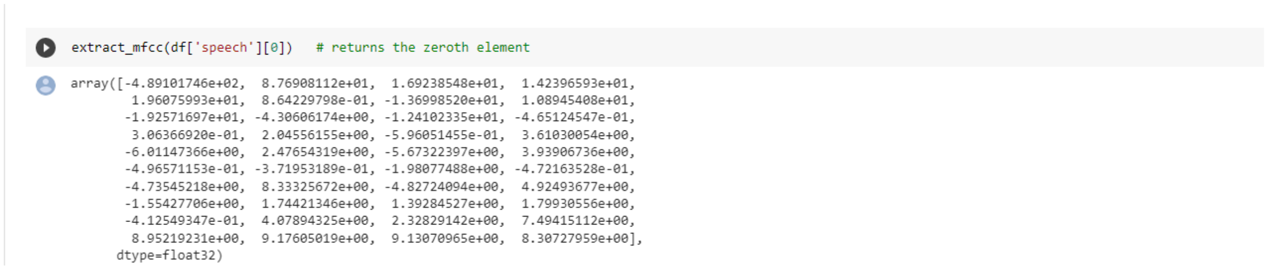


*Feature extraction *

```
[ ] #Mel-frequency cepstral coefficients (MFCCs)

    def extract_mfcc(filename):  # feature extraction function
        y, sr = librosa.load(filename,duration= 3, offset =0.5)   #Load an audio file as a floating point time series.
        # set the default time duration of audio file as 3 sec
        mfcc = np.mean(librosa.feature.mfcc(y = y ,sr= sr, n_mfcc = 40).T,axis =0)
        # librosa.feature.mfcc returns the mfcc sequence
        #np.mean Compute the arithmetic mean along the specified axis
        # n_mfcc means no of mfcc to return sr is the sampling rate
        return mfcc

[ ]
```

So we have created here a function named extract mfcc and in which we have defined the duration of the feature extraction process as the various data have different lengths so we have set the duration to 3 sec and ofiset to 0.5 in the third step we have extracted the features of our data given in the filename.

For example for the first data the mfcc of the first file in the array file is looking something like this:

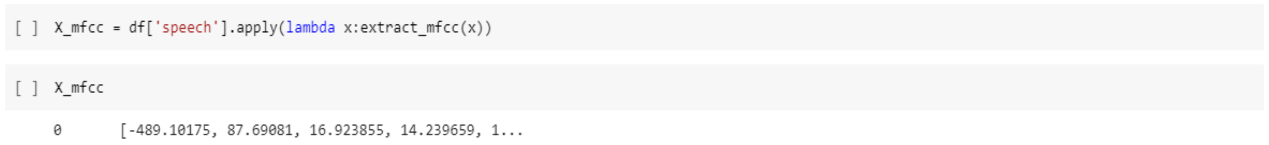


```
extract_mfcc(df['speech'][0])   # returns the zeroth element
```

```
array([-4.89101746e+02,  8.76908112e+01,  1.69238548e+01,  1.42396593e+01,
        1.96075993e+01,  8.64229798e-01, -1.36998520e+01,  1.08945408e+01,
       -1.92571697e+01, -4.30606174e+00, -1.24102335e+01, -4.65124547e-01,
        3.06366920e-01,  2.04556155e+00, -5.96051455e-01,  3.61030054e+00,
       -6.01147366e+00,  2.47654319e+00, -5.67322397e+00,  3.93906736e+00,
       -4.96571153e-01, -3.71953189e-01, -1.98077488e+00, -4.72163528e-01,
       -4.73545218e+00,  8.33325672e+00, -4.82724094e+00,  4.92493677e+00,
       -1.55427706e+00,  1.74421346e+00,  1.39284527e+00,  1.79930556e+00,
       -4.12549347e-01,  4.07894325e+00,  2.32829142e+00,  7.49415112e+00,
        8.95219231e+00,  9.17605019e+00,  9.13070965e+00,  8.30727959e+00],
      dtype=float32)
```

So we have got 39 values here and we will use these values for the input

Extracting features:



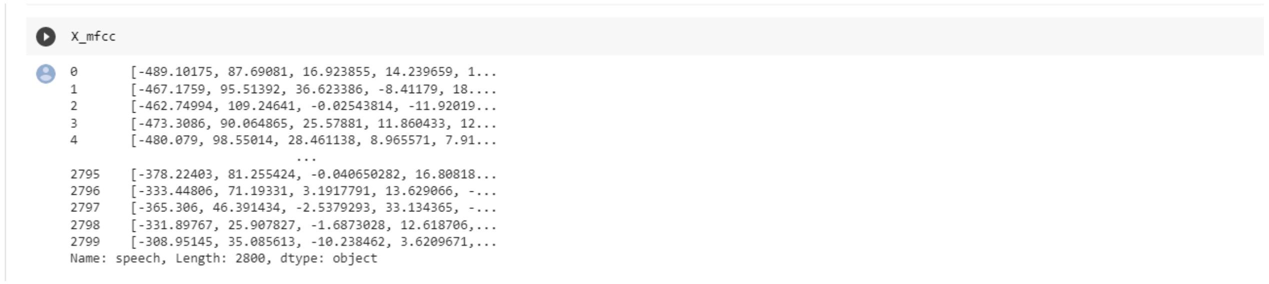
```
[ ] X_mfcc = df['speech'].apply(lambda x:extract_mfcc(x))
```

```
[ ] X_mfcc
```

```
0      [-489.10175, 87.69081, 16.923855, 14.239659, 1...
```

Like in the previous step we have used the mfcc on only one sample here in this module we are going to apply the mfcc function on the whole sample that is our whole 2800 samples, and this is one of the major steps in this whole process.

After this step we will get the data in some sequential form like this:

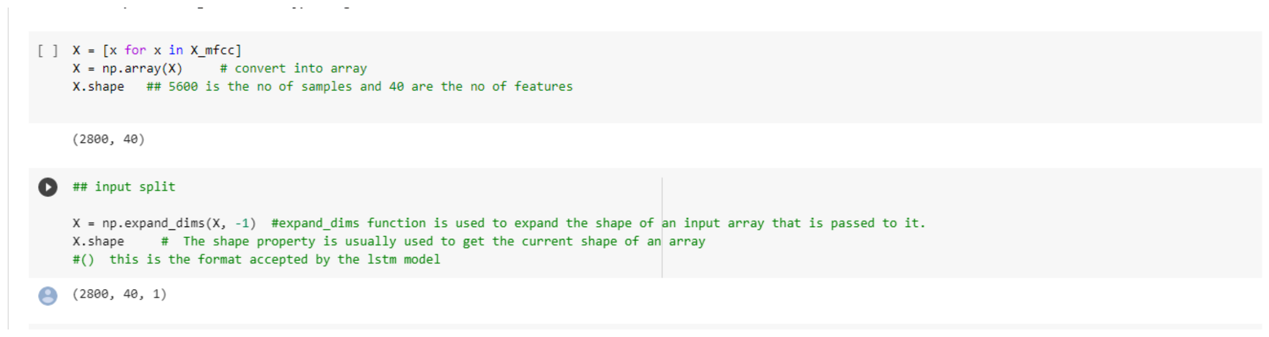


```
X_mfcc
```

```
0       [-489.10175, 87.69081, 16.923855, 14.239659, 1...
1       [-467.1759, 95.51392, 36.623386, -8.41179, 18....
2       [-462.74994, 109.24641, -0.02543814, -11.92019...
3       [-473.3086, 90.064865, 25.57881, 11.860433, 12...
4       [-480.079, 98.55014, 28.461138, 8.965571, 7.91...
                                ...
2795    [-378.22403, 81.255424, -0.040650282, 16.80818...
2796    [-333.44806, 71.19331, 3.1917791, 13.629066, -...
2797    [-365.306, 46.391434, -2.5379293, 33.134365, -...
2798    [-331.89767, 25.907827, -1.6873028, 12.618706,...
2799    [-308.95145, 35.085613, -10.238462, 3.6209671,...
Name: speech, Length: 2800, dtype: object
```

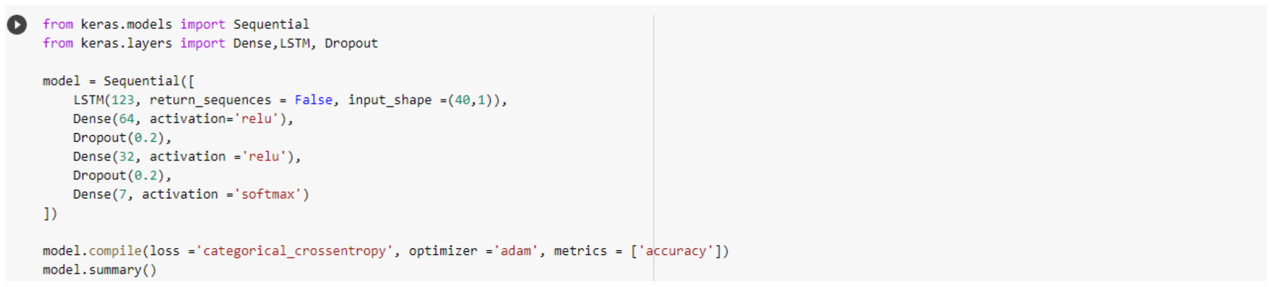
Creating the LSTM Model:

To convert the data into the 3d array form as used by the LSTM model we will use the following steps:



```
[ ] X = [x for x in X_mfcc]
    X = np.array(X)     # convert into array
    X.shape   ## 5600 is the no of samples and 40 are the no of features

    (2800, 40)
```

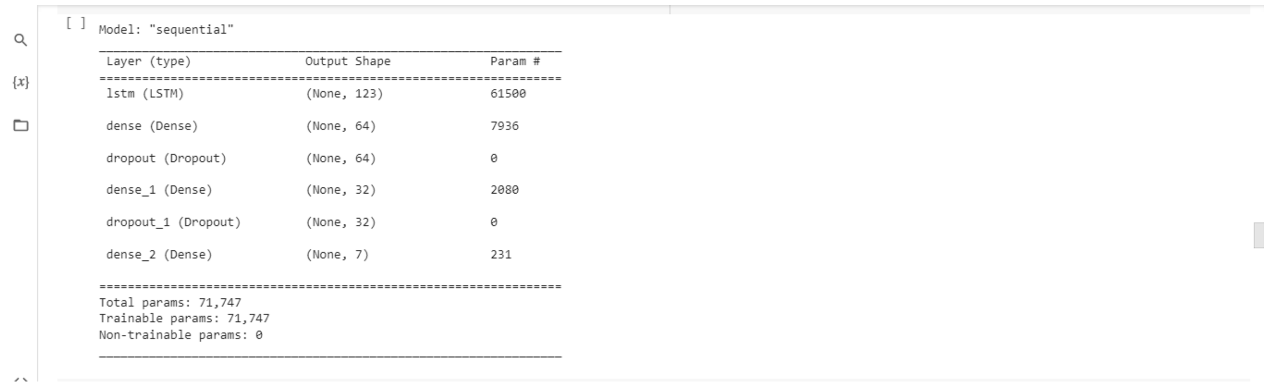
```
## input split

X = np.expand_dims(X, -1)  #expand_dims function is used to expand the shape of an input array that is passed to it.
X.shape     #  The shape property is usually used to get the current shape of an array
#()  this is the format accepted by the lstm model

(2800, 40, 1)
```



```
from keras.models import Sequential
from keras.layers import Dense,LSTM, Dropout

model = Sequential([
    LSTM(123, return_sequences = False, input_shape =(40,1)),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation ='relu'),
    Dropout(0.2),
    Dense(7, activation ='softmax')
])

model.compile(loss ='categorical_crossentropy', optimizer ='adam', metrics = ['accuracy'])
model.summary()
```

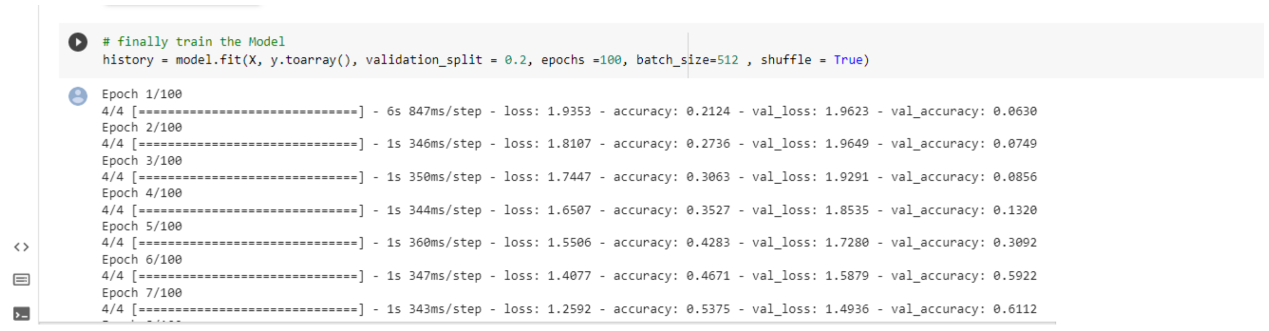
For creating the LSTM model we have to import the sequential, dense, LSTM and the dropout form keras models and keras.lavers respectively. Then specifying the various values required

Then specifying the various values required. In the next step compiling the model for loss and accuracy and then model. summary to view the model it will look something like this.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 123)               61500

 dense (Dense)               (None, 64)                7936

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 dropout_1 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 7)                 231

=================================================================
Total params: 71,747
Trainable params: 71,747
Non-trainable params: 0
_________________________________________________________________
```

Training The Model.



```
# finally train the Model
history = model.fit(X, y.toarray(), validation_split = 0.2, epochs =100, batch_size=512 , shuffle = True)
```

```
Epoch 1/100
4/4 [==============================] - 6s 847ms/step - loss: 1.9353 - accuracy: 0.2124 - val_loss: 1.9623 - val_accuracy: 0.0630
Epoch 2/100
4/4 [==============================] - 1s 346ms/step - loss: 1.8107 - accuracy: 0.2736 - val_loss: 1.9649 - val_accuracy: 0.0749
Epoch 3/100
4/4 [==============================] - 1s 350ms/step - loss: 1.7447 - accuracy: 0.3063 - val_loss: 1.9291 - val_accuracy: 0.0856
Epoch 4/100
4/4 [==============================] - 1s 344ms/step - loss: 1.6507 - accuracy: 0.3527 - val_loss: 1.8535 - val_accuracy: 0.1320
Epoch 5/100
4/4 [==============================] - 1s 360ms/step - loss: 1.5506 - accuracy: 0.4283 - val_loss: 1.7280 - val_accuracy: 0.3092
Epoch 6/100
4/4 [==============================] - 1s 347ms/step - loss: 1.4077 - accuracy: 0.4671 - val_loss: 1.5879 - val_accuracy: 0.5922
Epoch 7/100
4/4 [==============================] - 1s 343ms/step - loss: 1.2592 - accuracy: 0.5375 - val_loss: 1.4936 - val_accuracy: 0.6112
```

Validation split = it will do the splitting for us.

Epochs = the number of complete passes through the complete dataset.

Batch size= the number of training examples utilized in one iteration.

Emotion Recognized.

So at last by training and testing the model the emotion of the different samples of the voices are recognized by an overall accuracy o**f 70%.** The detailed result of the experiment are given below in the result analysis.

Conclusion :

In this project we have tried to analyze some samples of speech using the deep learning technique. Firstly we loaded the datasets then we **visualized** the different human emotions using our functions waveshow and spectrogram using the Librosa library. Then we extracted the acoustic features of all our samples using the Mfcc method and arranged the sequential data obtained in the 3D array form as accepted by the LSTM model. Then we build the LSTM model and after training the model. we visualized the data into the graphical form using matplotlib library and after some repeated testing using different values the average accuracy of the model is found to be 70%.