



SPEECH EMOTION RECOGNITION

TEAM:

18JG1A0585 - DIVYA SARAGADA

18JG1A0583 - SAHITHI SARIKA PONNADA

18JG1A0579 - LAVANYA RAVINUTALA

# 3.REQUIREMENTS:

3.1 HARDWARE REQUIREMENTS:

* + - **Processors:**Intel® Core™ i5 processor.
    - 4 GB RAM
    - **Operating systems:** Windows10

3.2 SOFTWARE REQUIREMENTS:

* + - **Programing language:**python3.6.8
    - **Framework:** Django3.2.8
    - **Text editor:**Sublime
    - **Tools:** Jupyter Notebook

ABSTRACT:

Emotions are important part of understanding human interactions. Research is going into finding methods that can at the very least mimic human ability to recognise emotions displayed in the form of facial expressions, changes in tone while speaking, etc.

Speech Emotion Recognition (SER) is one of such fields. Using deep learning and machine learning algorithms, we aim to design an automatic emotion recognition system.

Idea is creating a machine learning model that could detect emotions from the speech we have with each other all the time.

VARIOUS EMOTIONS TO BE CLASSIFIED:

Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).

**Methodology / Model :**

* machine learning
* deep learning strategy

**Models:**

* convolutional neural networks (**CNN**)
* support vector machine (**SVM**) classifier
* **MLP** classifier
* **KNN** Classifier

APPLICATIONS:

1.Used by marketing company to suggest products based on emotions.

2.By automotive industry that can detect emotion and adjust speed of autonomous cars as required to avoid accidents.

3. Used in call centers for classifying calls according to emotions.

DATASETS:

Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset :

This portion of the RAVDESS contains

60 trials per actor x 24 actors = 1440 files.

The RAVDESS contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech emotions includes calm, happy, sad, angry, fearful, surprise, and disgust expressions.

File naming convention

Each of the 1440 files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 03-01-06-01-02-01-12.wav). These identifiers define the stimulus characteristics.

Filename identifiers:

* Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
* Vocal channel (01 = speech, 02 = song).
* Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
* Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
* Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
* Repetition (01 = 1st repetition, 02 = 2nd repetition).

Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

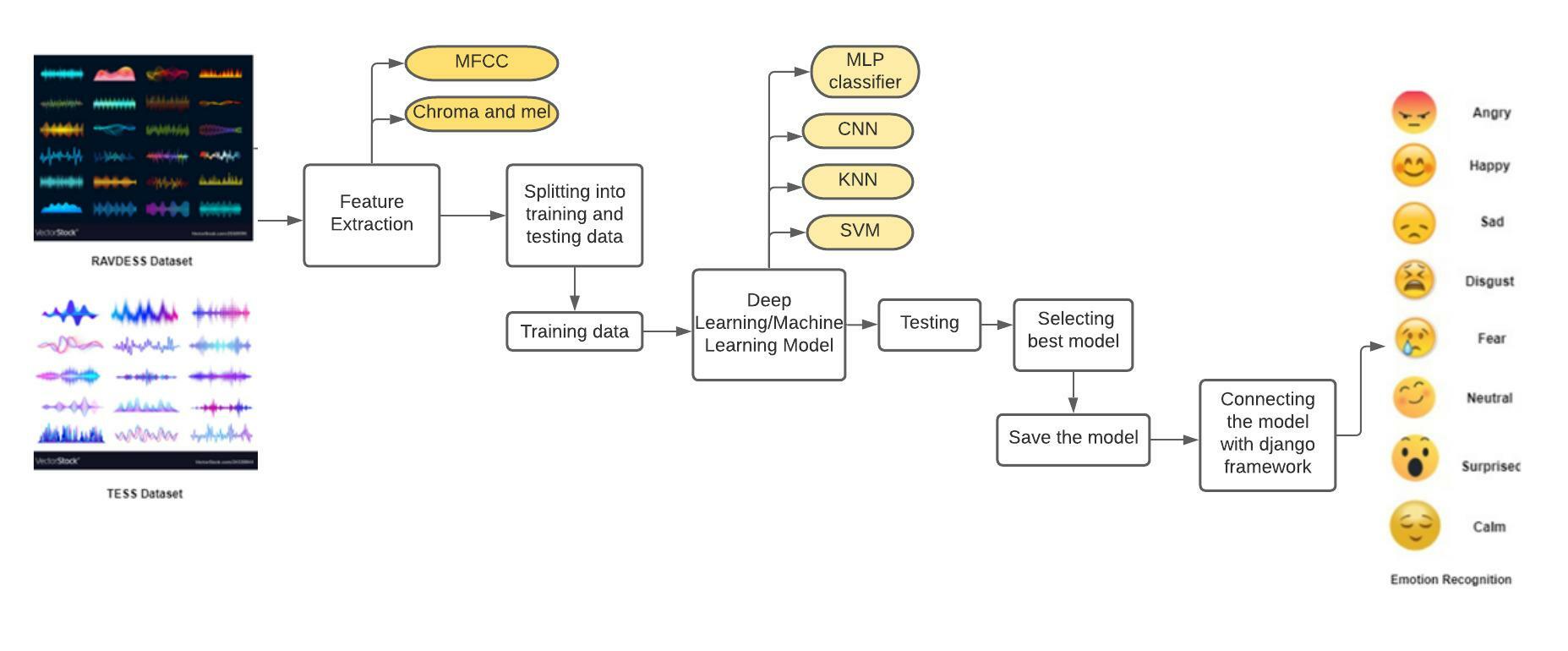
Toronto emotional speech set data **(**TESS):

There are a 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.

Two datasets are combined into 4240 files.

SYSTEM ARCHITECTURE:



SPEECH TO TEXT USING SPEECH RECOGNITION PACKAGE:

Initially we tested the audio by translating it back into the text mode using speech recognition package to know what the audio is all about.

DATA VISUALIZATION:

To test the audio files by plotting out the wave forms and spectrograms to see the intensity and frequency in the plots.

CONVOLUTIONAL NEURAL NETWORKS(CNN):

A **Convolutional neural network** (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data. A convolution is essentially sliding a filter over the input.

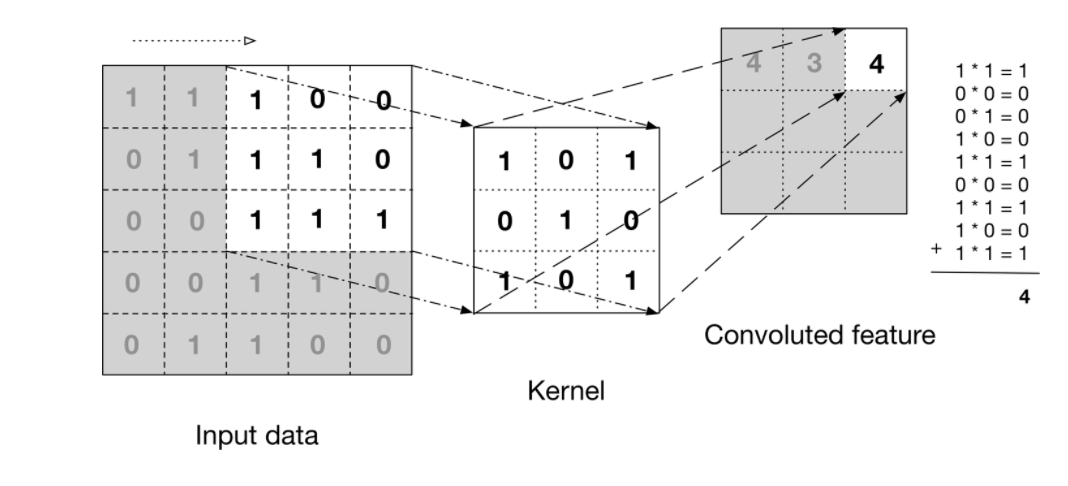
Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

HOW DOES IT WORK:

It uses a special technique called Convolution instead of matrix multiplications.

An RGB image is nothing but a matrix of pixel values having three planes whereas a grayscale image is the same but it has a single plane.

For eg: If we take a grey scale image i.e 2x2 dimension,



The above image shows what a convolution is. We take a filter/kernel(3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.

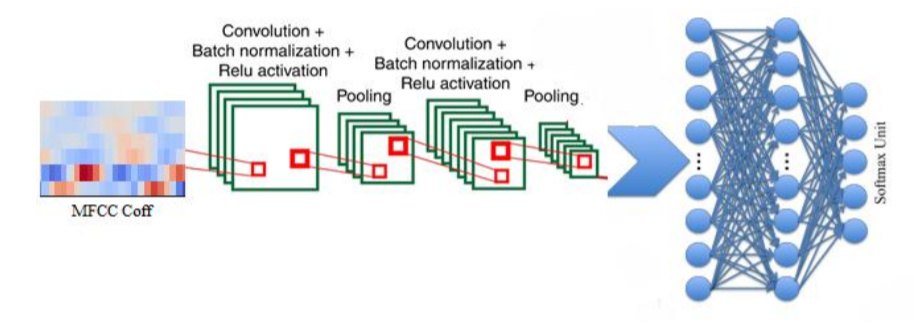
The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

POOLING LAYER:

pooling layer is responsible for reducing the spatial size of the convolved Feature; to decrease the computational power required to process the data by reducing the dimensions

There are 2 types of pooling: average pooling and max pooling.

* **Kernel sizes**: I'm using one 5x1 and 5x1 kernel sizes as the smallest size that worked out on my experiments. Also notice I use odd numbers for kernel sizes, this is a very important
* **Batch normalization**: In short, it normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. This process optimizes training time.
* **Spatial dropout**: A regularization by dropping out entire feature maps by a given rate, preventing activations from becoming strongly correlated.
* **Dense (softmax output)**: the final layer containing the softmax output that will provide the classification probabilities for the input data.



Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a “class.”

In multiclass audio classification spectrogram acts as grey scale image and can be given as a input to the convolution layer.

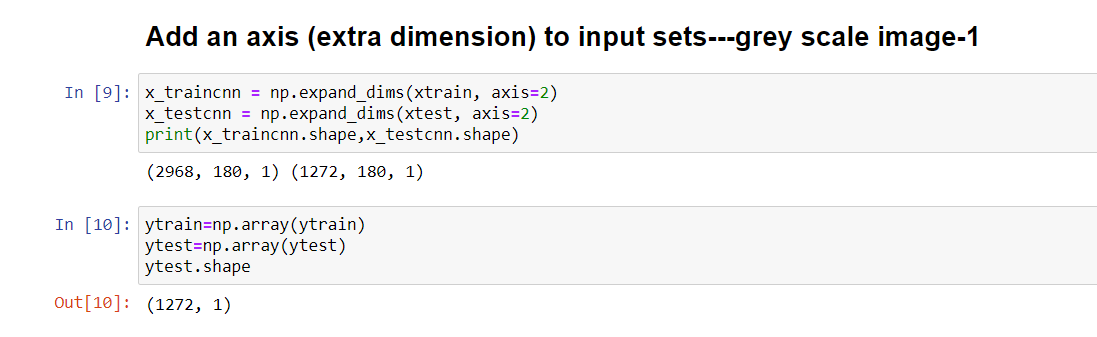
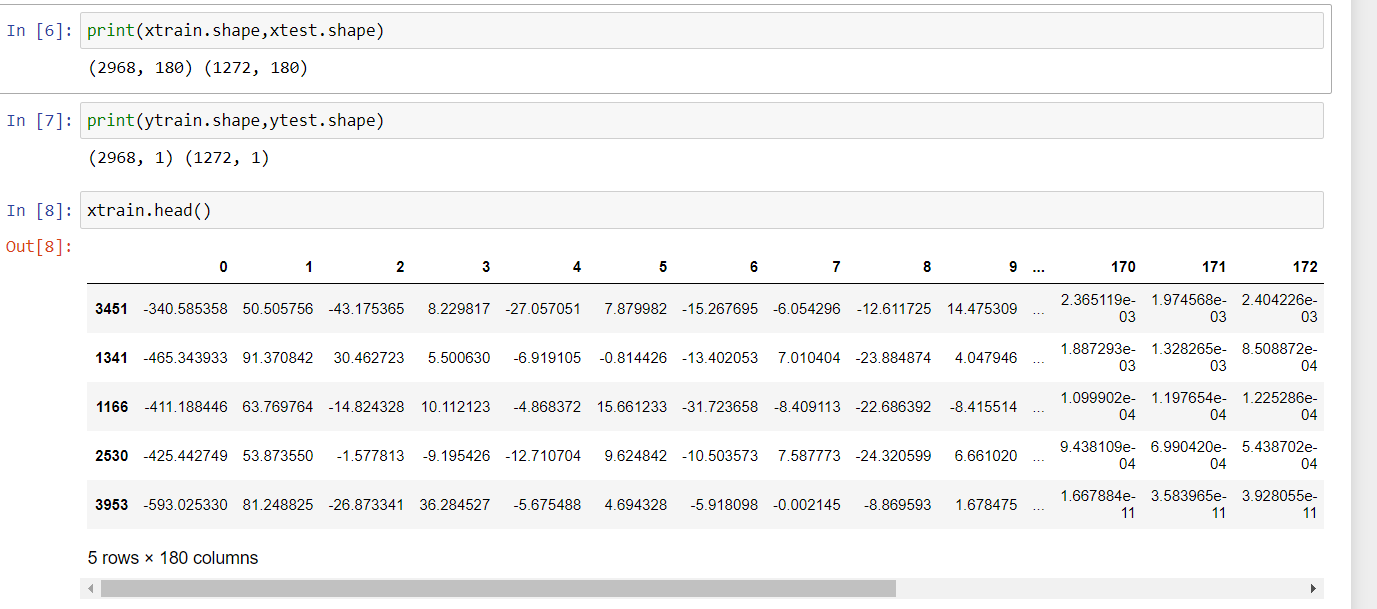
But here we used Extracted Features (1D array) as input ,know as CONV1D layer.

CODE:



In multiclass audio classification spectrogram acts as grey scale image and can be given as a input to the convolution layer.

But here we used Extracted Features (2D array) as input ,know as CONV1D layer.



## Building the Model using CNN:

import matplotlib.pyplot as plt

import tensorflow.keras as keras

from keras.preprocessing import sequence

from keras.models import Sequential

from keras.layers import Dense, Embedding

from keras.utils import to\_categorical

from keras.layers import Input, Flatten, Dropout, Activation, BatchNormalization

from keras.layers import Conv1D, MaxPooling1D

from keras.models import Model

from keras.callbacks import ModelCheckpoint

model = Sequential()

#first layer

model.add(Conv1D(64, 5,padding='same',

input\_shape=(180,1)))

model.add(Activation('relu'))

model.add(Dropout(0.1))

model.add(MaxPooling1D(pool\_size=(4)))

#second layer

model.add(Conv1D(128, 3,padding='same',))

model.add(Activation('relu'))

model.add(Dropout(0.1))

model.add(MaxPooling1D(pool\_size=(4)))

#third layer

model.add(Conv1D(256, 3,padding='same',))

model.add(Activation('relu'))

model.add(Dropout(0.1))

#dense layer

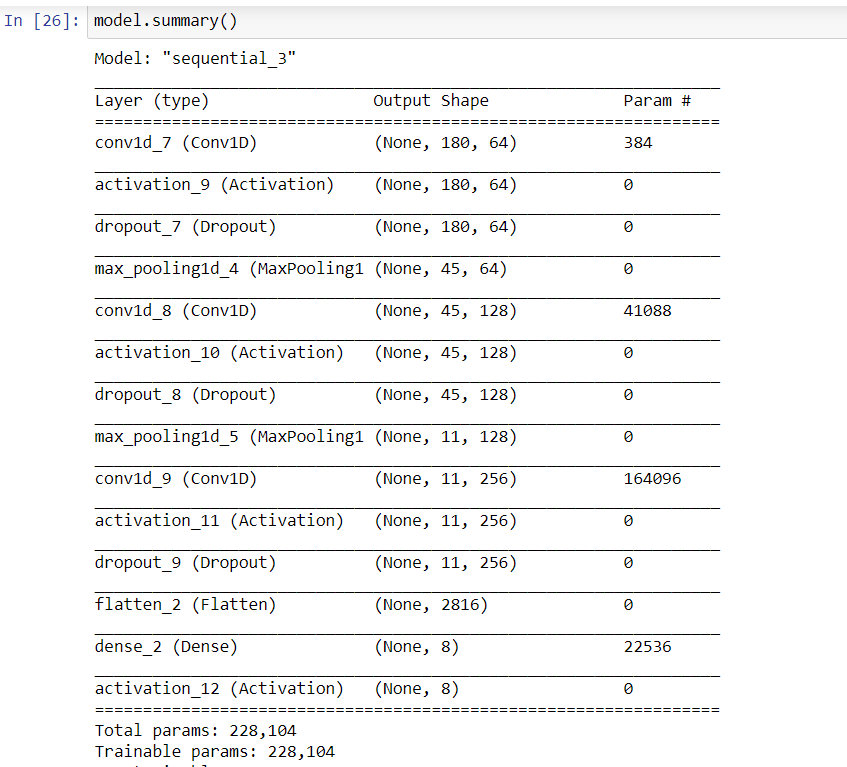
model.add(Flatten())

model.add(Dense(8))

model.add(Activation('softmax'))

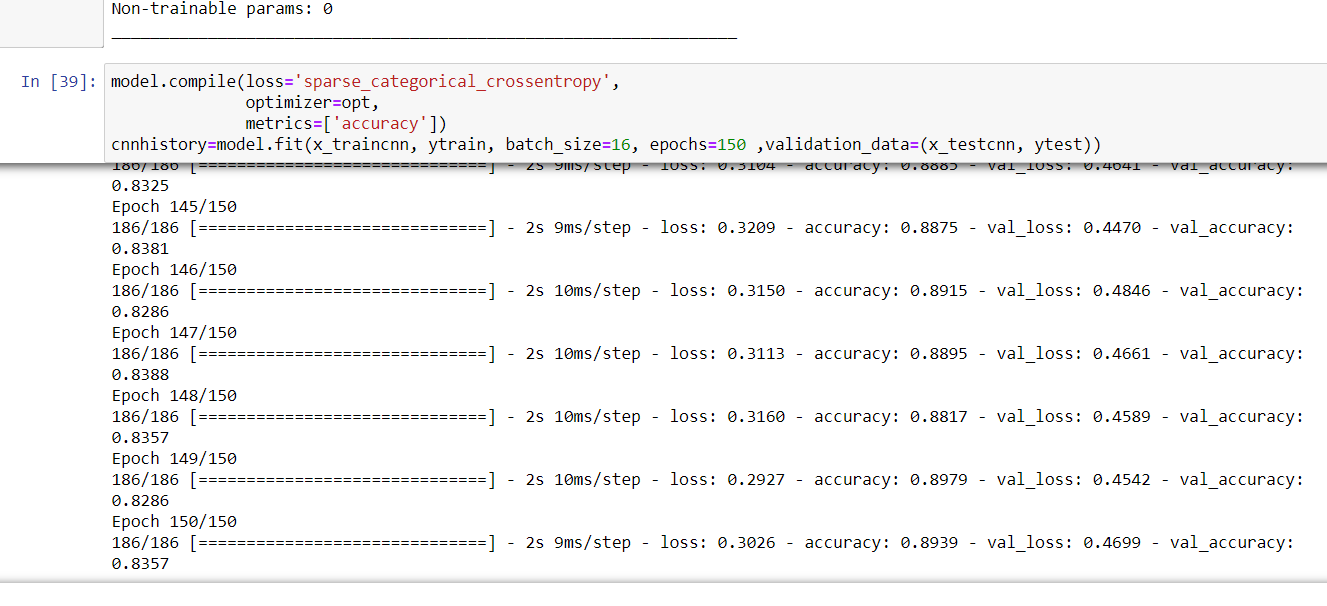
opt = keras.optimizers.Adam(lr=0.00005)

model.summary()



COMPILATION AND FITTING TRAINING DATA TO THIS MODEL:

Using ADAM OPTIMIZER with default parameters to start from somewhere, then reduced learning rate to 1e-4 (as loss history was noisy). Loss function is categorical cross-entropy as we are predicting between 8 classes.



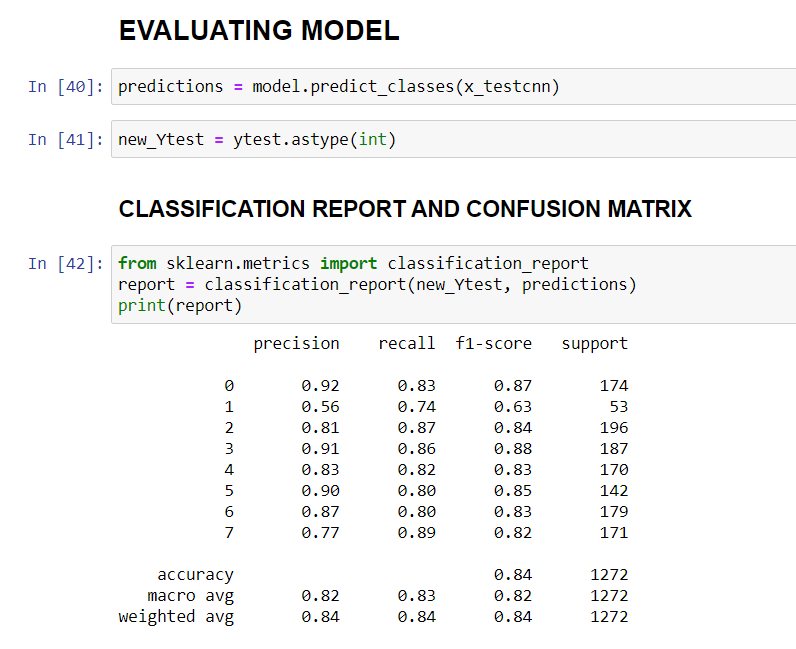
EVALUATING MODEL:

Predicting our cnn model on testing data and checking the accuracy.

Model is evaluated using confusion matrix and classification report from sklearn library.

**Classification matrix:**

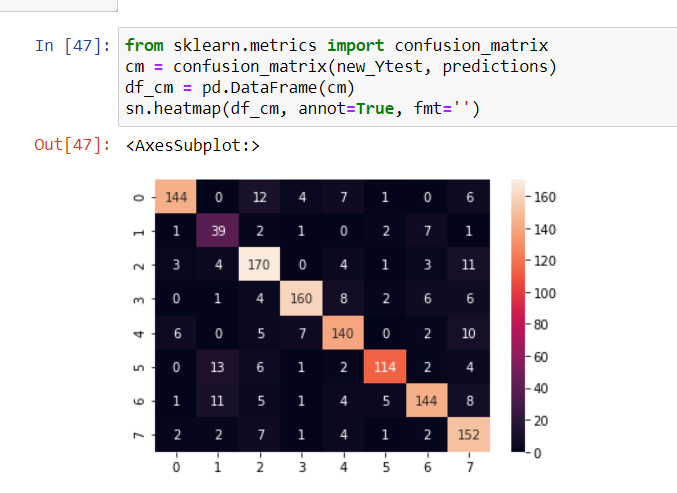
Classification report is straightforward - a report of Precision/recall/f1 score -Measure for each element in your test data.

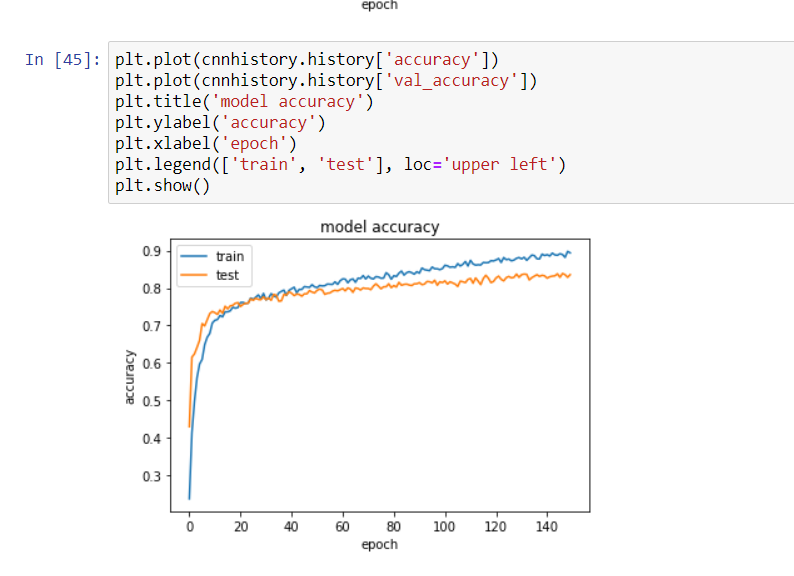
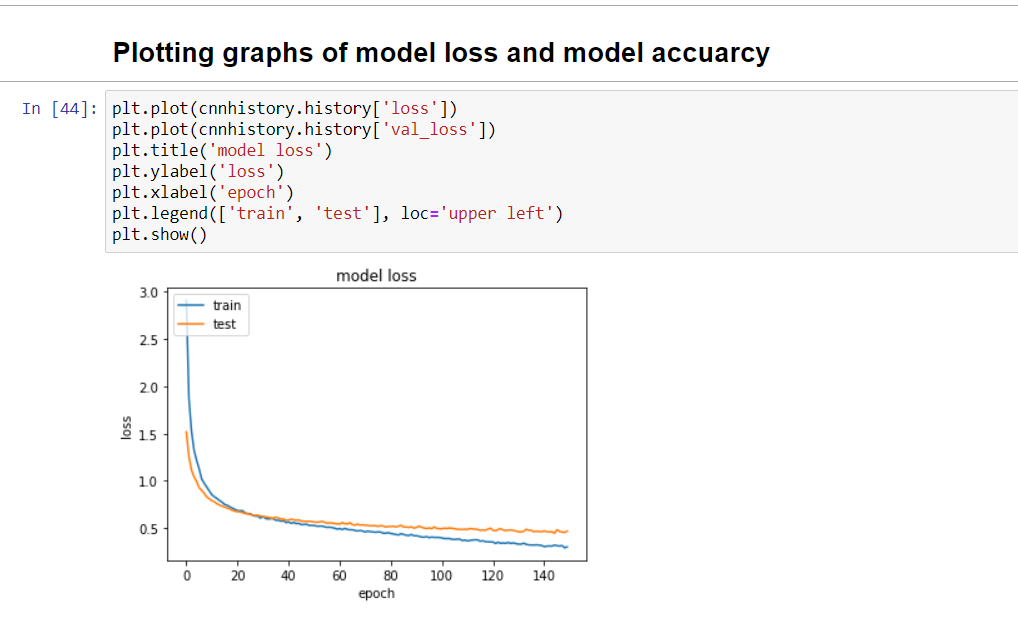


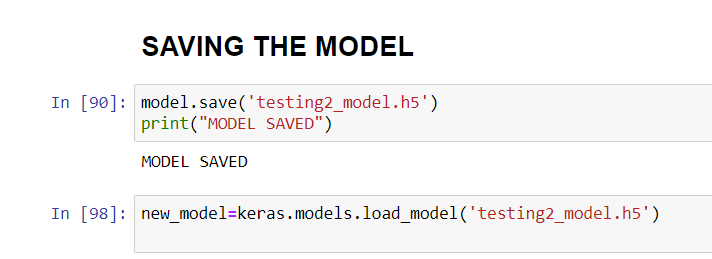
**Confusion matrix:**

A confusion matrix is a summary of prediction results on a classification problem.

The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.







The model is saved as testing2\_model.h5 in the same directory ,which later can be used to connect with django framework.

# DEPLOYMENT OF MODEL USING DJANGO FRAMEWORK:

**DJANGO:**

* It is a high level python framework for web development.
* Elegant system for mapping URLs to python code

**Creating a project:**

🡪django –admin startproject SpeechEmotion

**Creating an app within a project directory:**

🡪 python manage.py startapp Prediction

#### Update URL’s.py

To access all the different parts of our Django app, we need to specify the URLs of the app.

Urls.py:

from django.contrib import admin

from django.urls import path

from Prediction import views

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path('admin/', admin.site.urls),

path('home/',views.home,name='home'),

path('output/',views.output,name='output'),

path('record/',views.record,name='record'),

]

#### Update View in Django:

In Django, view is a python function that takes all the web requests of the site and returns web responses. The responses can be anything, in the project we need to redirect the user to the form, collect the data from, process it, and show the result to the users. All these things will be done in the view.

## Views.py:

from django.shortcuts import render

from django.contrib import messages

from django.http import HttpResponse

def home(req):

if req.method=="POST":

#try:

file=req.FILES['filewav']

def speech\_to\_text(filewav):

import speech\_recognition as sr

# get audio from the microphone

r = sr.Recognizer()

with sr.AudioFile(filewav) as source:

audio = r.listen(source)

try:

output = " " + r.recognize\_google(audio)

except sr.UnknownValueError:

output = "Could not understand audio"

except sr.RequestError as e:

output = "Could not request results; {0}".format(e)

return output

import keras

import numpy as np

import librosa

class predictions:

def \_\_init\_\_(self, path, file): #Init method is used to initialize the main parameters.

self.path = path

self.file = file

def load\_model(self): #method to load h5 model and returns summary of that model

self.model = keras.models.load\_model(self.path)

return self.model.summary()

def makepredictions(self):

chroma=True

mel=True

mfcc=True

X, sample\_rate = librosa.load(self.file, res\_type='kaiser\_fast')

result=np.array([])

if chroma:

stft=np.abs(librosa.stft(X))

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))

x=result

x = np.expand\_dims(x, axis=1)

x = np.expand\_dims(x, axis=0)

predictions=np.argmax(self.model.predict(x), axis=-1)

# predictions = self.model.predict\_classes(x)

p=predictions

p1=self.convertclasstoemotion(predictions)

return p,p1

@staticmethod

def convertclasstoemotion(pred):

label\_conversion ={

0:'angry',

1:'calm',

2:'disguist',

3:'fear',

4:'happy',

5:'neutral',

6:'sad',

7:'surprised'

}

for key, value in label\_conversion.items():

if int(key) == pred:

label = value

return label

pred = predictions(path='C:\\Users\\DIVYA\\Desktop\\SPEECH EMOTION RECOGNITION\\testing10\_model.h5',file=file)

pred.load\_model()

p,p1=pred.makepredictions()

output=speech\_to\_text(file)

return render(req,'Prediction/output.html',{'text':output,'emotion':p1,'audiofile':file})

#except Exception:

# messages.warning(req,"PLEASE SELECT AUDIO FILE")

# return render(req,'Prediction/home.html')

return render(req,'Prediction/home.html')

# Create your views here.

def output(req):

return render(req,'Prediction/output.html')

def record(req):

if req.method=='POST':

import pyaudio

import wave

CHUNK = 1024

FORMAT = pyaudio.paInt16

CHANNELS = 2

RATE = 44100

RECORD\_SECONDS = 5

WAVE\_OUTPUT\_FILENAME = "C:\\Users\\DIVYA\\Desktop\\SpeechEmotion\\Prediction\\static\\audio\\output.wav"

p = pyaudio.PyAudio()

stream = p.open(format=FORMAT,

channels=CHANNELS,

rate=RATE,

input=True,

frames\_per\_buffer=CHUNK)

print("\* recording")

frames = []

for i in range(0, int(RATE / CHUNK \* RECORD\_SECONDS)):

data = stream.read(CHUNK)

frames.append(data)

print("\* done recording")

messages.success(req,"AUDIO ADDED SUCCESSFULLY..")

stream.stop\_stream()

stream.close()

p.terminate()

wf = wave.open(WAVE\_OUTPUT\_FILENAME, 'wb')

wf.setnchannels(CHANNELS)

wf.setsampwidth(p.get\_sample\_size(FORMAT))

wf.setframerate(RATE)

wf.writeframes(b''.join(frames))

wf.close()

return render(req,'Prediction/home.html')

return render(req,'Prediction/home.html')

## TEMPLATES:

## home.html:

{% load static %}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">

<title>HOME PAGE</title>

<link rel="stylesheet" type="text/css" href="{% static 'css/mystyle.css' %}">

</head>

<body>

<form action="{% url 'home' %}" method="POST" enctype="multipart/form-data">

{% csrf\_token %}

<center>

{% include 'Prediction/messages.html' %}

<h1>PREDICT THE EMOTION</h1>

<br>

<input name="filewav" type="file" accept="audio/x-wav"/>

<br><br><br>

<button class="btn btn-info">PREDICT&nbsp;EMOTION</button>

<br><br>

</form>

<form action="{% url 'record' %}" method="POST" enctype="multipart/form-data">

{% csrf\_token %}

<button class="btn btn-danger">

RECORD&nbsp;AUDIO</button>

</center>

</form>

</body>

</html>

## Output.html:

{% load static %}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">

<title>OUTPUT PAGE</title>

<link rel="stylesheet" type="text/css" href="{% static 'css/mystyle.css' %}">

</head>

<body>

<center>

{% include 'Prediction/messages.html' %}

<br><br><br><br>

<h2>AUDIO :</h2>

<br>

<h3>{{ text }}</h3>

<!--<audio controls >

<source src="{{ audiofile }}" type="audio/wav">

</audio>-->

<br><br><br><br>

<h1>PREDICTED EMOTION :</h1>

<h1>{{ emotion }}</h1>

</center>

</body>

</html>

## Messages.html:

{% if messages %}

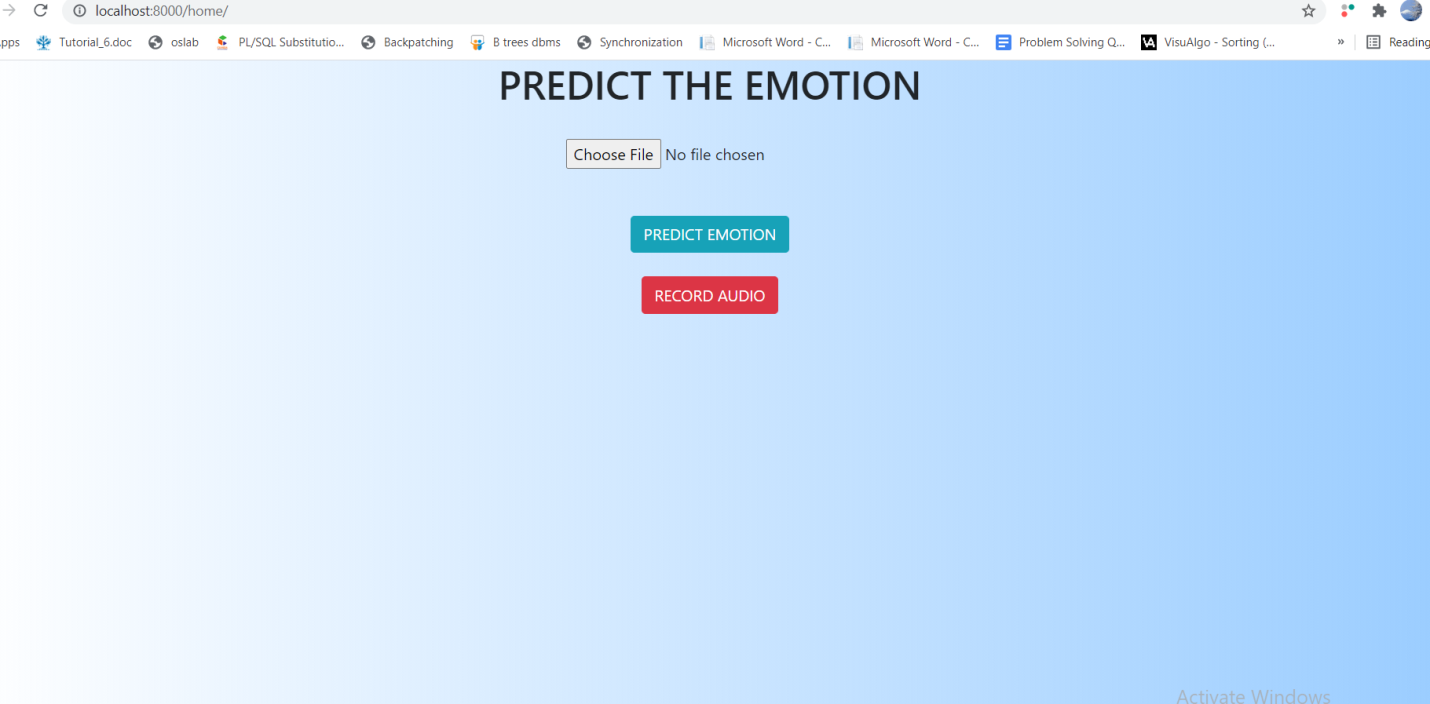
{% for message in messages %}

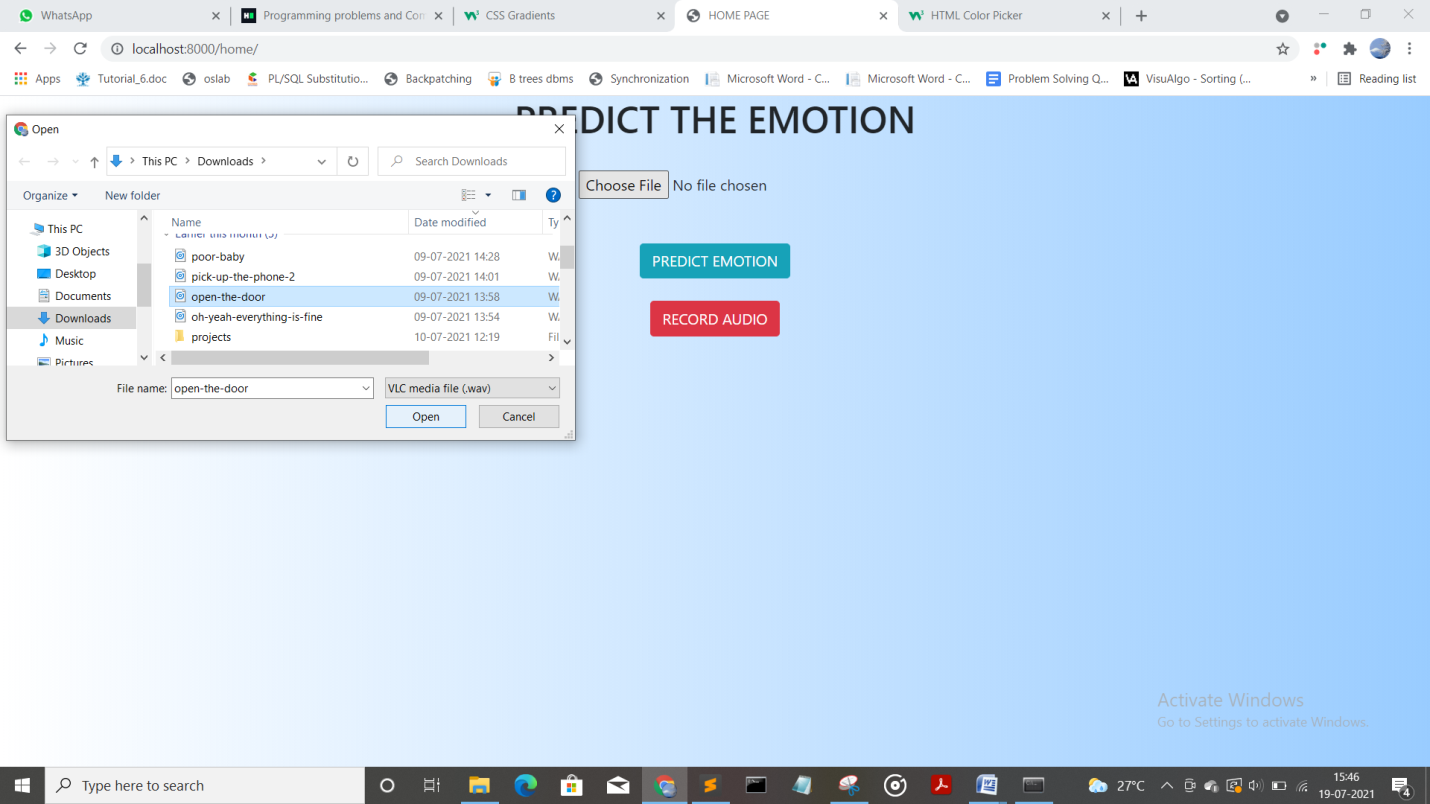
<div class="alert alert-{{ message.tags }}" role="alert">{{ message }}</div>

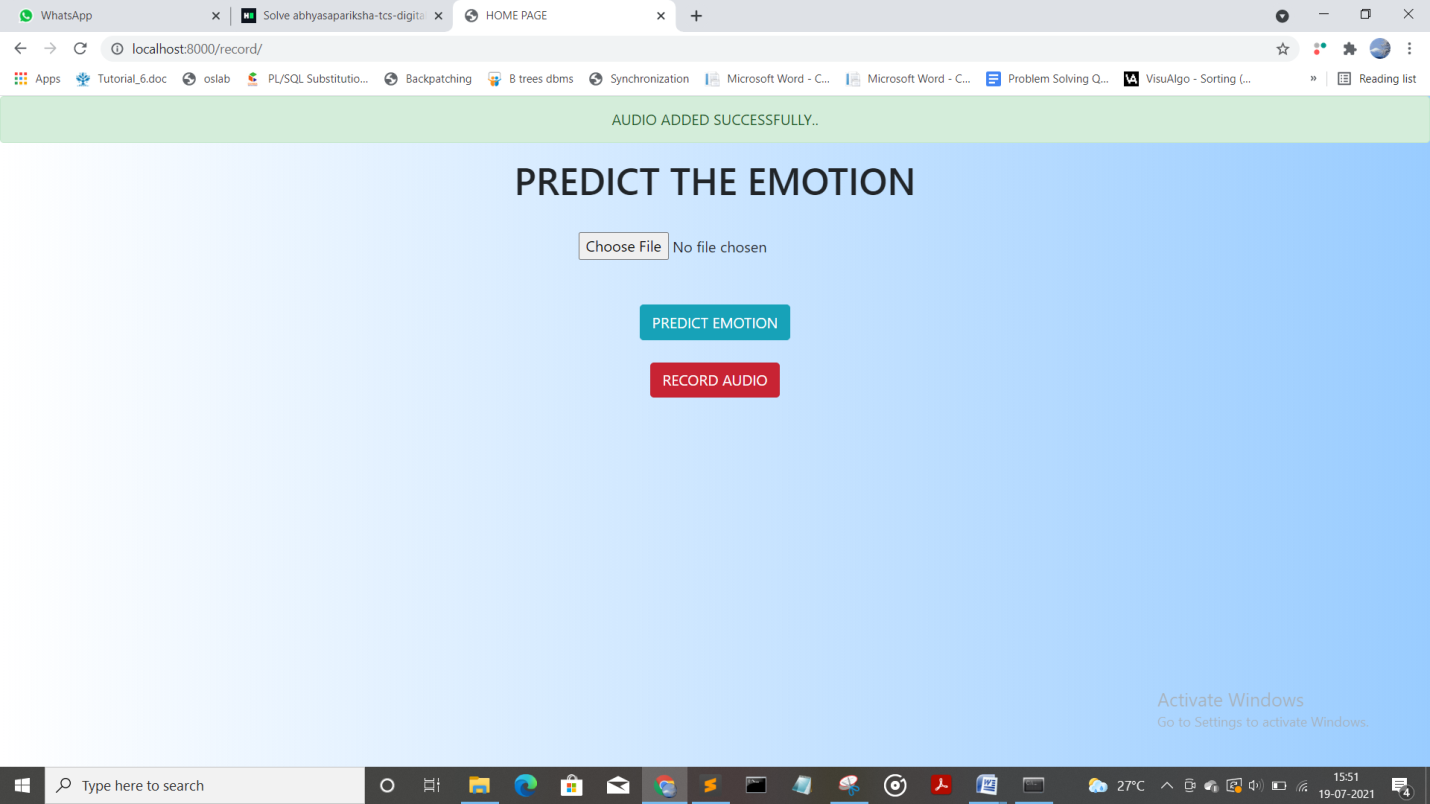
{% endfor %}

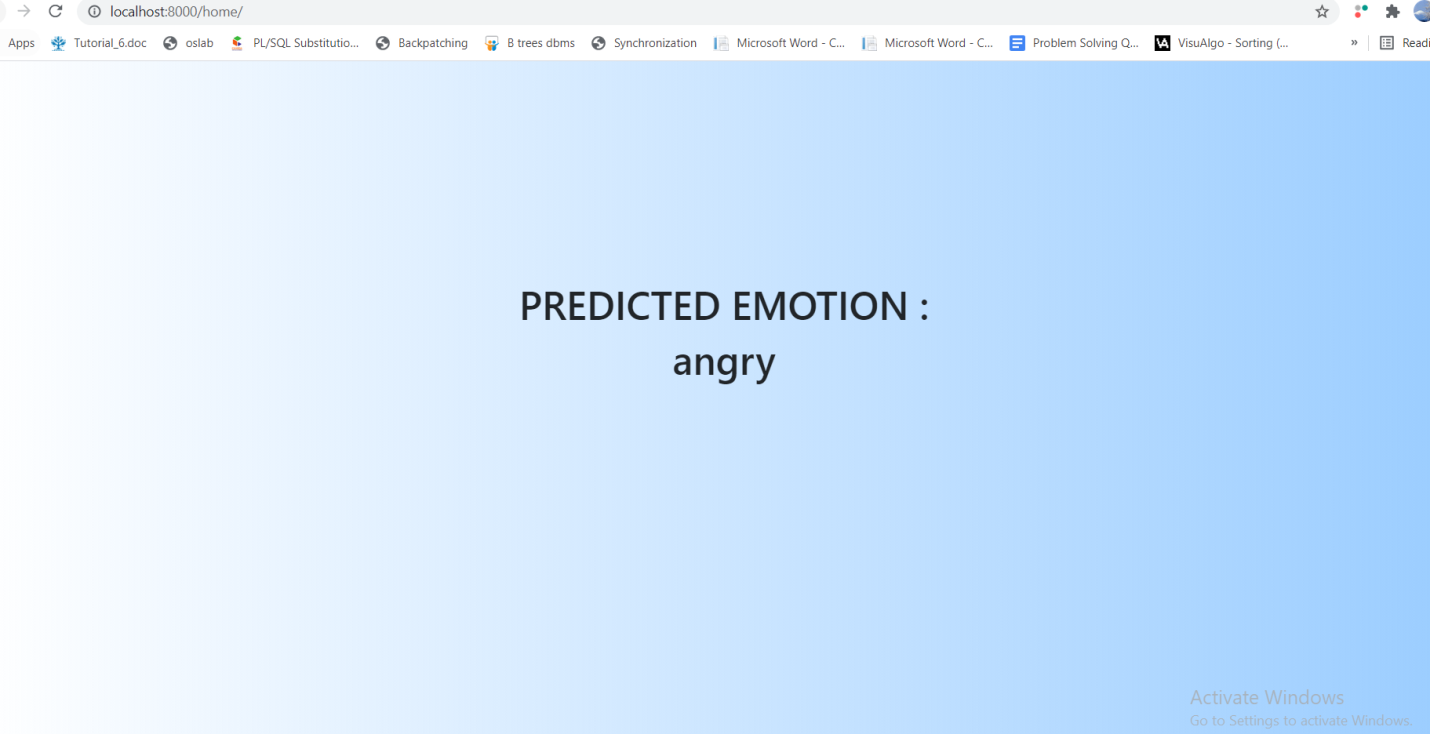
{% endif %}

## OUTPUT SCREENSHOTS:









## CONCLUSION:

Emotion Recognition serves as the performance parameter for conversational analysis thus identifying the unsatisfied customer, customer satisfaction so on.

In this project MFCC,Chroma and mel features are extracted using python Librosa Library.Models like MLP classifier and CNN model both got an accuracy of 85% and 83% each.So CNN model is deployed using Django framework.

Future of this project ,this can be trained with large number of data and can be used in various applications.

Github link: <https://github.com/divyasaragada/SpeechEmotion>

## REFERENCES:

1. <https://www.irjet.net/archives/V7/i9/IRJET-V7I9154.pdf>
2. <https://www.aionlinecourse.com/blog/deploy-machine-learning-model-using-django-and-rest-api>
3. <https://github.com/LeadingIndiaAI/Speech-Emotion-Recognition-using-Deep-learning>
4. <https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/>