# VISVESVARAYA TECHNOLOGICAL UNIVERSITY



## 

## 

## BELAGAVI – 590018,Karnataka

## INTERNSHIP REPORT ON

“Voice Classification using Machine learning”

**Submitted in partial fulfillment for the award of degree****(18CSI85)**

BACHELOR OF ENGINEERING IN

## Electronics and Communications Engineering

**Submitted by*:***

#### NAME: Manjunatha KV

#### USN: 1JS20EC421

**UNDER THE COMPANY**

**COMPSOFT TECHNOLOGIES COMPANY**

****

**JSS Academy of Technical Education,** **Bangalore**

**Department of Electronics and Communication Engineering**

**Uttarahalli-Kengeri, Bangalore, Karnataka-560060**

****

**JSS Academy of Technical Education,** **Bangalore**

**Department of Electronics and Communication Engineering**

**Uttarahalli-Kengeri, Bangalore, Karnataka-560060**

**CERTIFICATE**

This is to certify that the Internship titled **“**Voice Classification using Machine learning**”** carried out by **Mr.Manjunatha kv,** a Bonafide student of **JSS Academy of Technical** Education Institute Bangalore**,** in partial fulfillment for the award of Bachelor of Engineering, **Department of Electronics and** Communication Engineering in under Visvesvaraya Technological University, Belagavi, during the year 2020-2023. It is certified that all corrections/suggestions indicated have been incorporated in the report.

The project report has been approved as it satisfies the academic requirements in respectof Internship prescribed for the course Internship / Professional Practice (18CSI85)

#### Signature of Guide Signature of HOD Signature of Principal

**External Viva:**

Name of the Examiner Signature with Date

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# DECLARATION

I, **Manjunatha KV**, final year student of **Department of Electronics and Communication Engineering , JSS Academy of Technical Education,** **Bangalore, Uttarahalli-Kengeri, Bangalore, Karnataka-560060** declare that the Internship has been successfully completed, in **COMPSOFT TECHNOLOGIES COMPANY.** This report is submitted in partial fulfillment of the requirements for award of Bachelor Degree in **Department of Electronics and Communication Engineering ,**, during the academic year 2020-2023.

* **NAME : Manjunatha kv**
* **USN : 1JS20EC421**
* **DEPT : Department of Electronics and Communication Engineering**
* **DATE : 1/09/2022 TO 30/09/2022**
* **COMPANY: COMPSOFT TECHNOLOGIES COMPANY**

**OFFER -LETTER**

# ACKNOWLEDGEMENT

This Internship is a result of accumulated guidance, direction and support of several important persons. We take this opportunity to express our gratitude to all who have helped us to complete the Internship.

We express our sincere thanks to our Principal, for providing us adequate facilities to undertake this Internship.

We would like to thank our Head of Dept – branch code, for providing us an opportunity to carry out Internship and for his valuable guidance and support.

We would like to thank our (Lab assistant name) Software Services for guiding us during the period of internship.

We express our deep and profound gratitude to our guide, Guide name, Assistant/Associate Prof, for her keen interest and encouragement at every step in completing the Internship.

We would like to thank all the faculty members of our department for the support extended during the course of Internship.

We would like to thank the non-teaching members of our dept, forhelping us during the Internship.

Last but not the least, we would like to thank our parents and friends without whose constant help, the completion of Internship would have not been possible.

**NAME: Manjunatha kv**

**USN: 1JS20EC421**

# ABSTRACT

Communication is the key to express one’s thoughts and ideas clearly. Amongst all forms of communication, speech is the most preferred and powerful form of communications in human. The era of the Internet of Things (IoT) is rapidly advancing in bringing more intelligent systems available for everyday use.

These applications range from simple wearables and widgets to complex self-driving

vehicles and automated systems employed in various fields. Intelligent applications are interactive and require minimum user effort to function, and mostly function on voice-based input.

This creates the necessity for these computer applications to completely comprehend human speech.

A speech percept can reveal information about the speaker including gender, age, language, and emotion.

Several existing speech recognition systems used in IoT applications are integrated with an emotion detection system in order to analyze the emotional state of the speaker.

The performance of the emotion detection system can greatly influence the overall performance of the IoT 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.

# Table of Contents

|  |  |  |
| --- | --- | --- |
| **Sl no** | **Description** | **Page no** |
| 1 | Company Profile | 8 |
| 2 | About the Company | 9-11 |
| 3 | Introduction | 12 |
| 4 | System Analysis | 13-33 |
| 5 | Requirement Analysis | 34-35 |
| 6 | Design Analysis | 36 |
| 7 | Implementation | 37 |
| 8 | Snapshots | 38 |
| 9 | Conclusion | 39 |
| 10 | References | 40 |

[**CHAPTER**](https://1.bp.blogspot.com/-dODuK8N5h1Q/Wlnyb3V9HFI/AAAAAAAACL4/WxQtCJ1pM5wccDABg4wIrTBUB0vlikXQQCLcBGAs/s1600/poly1.jpg) **:1**

**COMPANY PROFILE**

## A Brief History of Compsoft Technologies

Compsoft Technologies, was incorporated with a goal ”To provide high quality and optimal Technological Solutions to business requirements of our clients”. Every business is a different and has a unique business model and so are the technological requirements. They understand this and hence the solutions provided to these requirements are different as well. They focus on clients requirements and provide them with tailor made technological solutions. They also understand that Reach of their Product to its targeted market or the automation of the existing process into e-client and simple process are the key features that our clients desire from Technological Solution they are looking for and these are the features that we focus on while designing the solutions for their clients.

Sarvamoola Software Services. is a Technology Organization providing solutions for all web design and development, MYSQL, PYTHON Programming, HTML, CSS, ASP.NET and LINQ. Meeting the ever increasing automation requirements, Sarvamoola Software Services. specialize in ERP, Connectivity, SEO Services, Conference Management, effective web promotion and tailor-made software products, designing solutions best suiting clients requirements.

Compsoft Technologies, strive to be the front runner in creativity and innovation in software development through their well-researched expertise and establish it as an out of the box software development company in Bangalore, India. As a software development company, they translate this software development expertise into value for their customers through their professional solutions.

They understand that the best desired output can be achieved only by understanding the clients demand better. Compsoft Technologies work with their clients and help them to defiine their exact solution requirement. Sometimes even they wonder that they have completely redefined their solution or new application requirement during the brainstorming session, and here they position themselves as an IT solutions consulting group comprising of high caliber consultants.

They believe that Technology when used properly can help any business to scale and achieve new heights of success. It helps Improve its efficiency, profitability, reliability; to put it in one sentence ” Technology helps you to Delight your Customers” and that is what we want to achieve.

# [CHAPTER :](https://1.bp.blogspot.com/-dODuK8N5h1Q/Wlnyb3V9HFI/AAAAAAAACL4/WxQtCJ1pM5wccDABg4wIrTBUB0vlikXQQCLcBGAs/s1600/poly1.jpg) 2

# ABOUT THE COMPANY



Compsoft Technologies is a Technology Organization providing solutions for all web design and development, MYSQL, PYTHON Programming, HTML, CSS, ASP.NET and LINQ. Meeting the ever increasing automation requirements, Compsoft Technologies specialize in ERP, Connectivity, SEO Services, Conference Management, effective web promotion and tailor-made software products, designing solutions best suiting clients requirements. The organization where they have a right mix of professionals as a stakeholders to help us serve our clients with best of our capability and with at par industry standards. They have young, enthusiastic, passionate and creative Professionals to develop technological innovations in the field of Mobile technologies, Web applications as well as Business and Enterprise solution. Motto of our organization is to “Collaborate with our clients to provide them with best Technological solution hence creating Good Present and Better Future for our client which will bring a cascading a positive effect in their business shape as well”. Providing a Complete suite of technical solutions is not just our tag line, it is Our Vision for Our Clients and for Us, We strive hard to achieve it.

# Products of Compsoft Technologies.

# Android Apps

It is the process by which new applications are created for devices running the Android operating system. Applications are usually developed in Java (and/or Kotlin; or other such option) programming language using the Android software development kit (SDK), but other development environments are also available, some such as Kotlin support the exact same Android APIs (and bytecode), while others such as Go have restricted API access.

The Android software development kit includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and zutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but softwaredevelopment is possible by using specialized Android applications.

# Web Application

It is a client–server computer program in which the client (including the user interface and client- side logic) runs in a web browser. Common web applications include web mail, online

retail sales, online auctions, wikis, instant messaging services and many other functions. web applications use web documents written in a standard format such as HTML and JavaScript,which are supported by a variety of web browsers. Web applications can be considered as a specifific variant of client–server software where the client software is downloaded to the client machine when visiting the relevant web page, using standard procedures such as HTTP. The Client web software updates may happen each time the web page is visited. During the session, the web browser interprets and displays the pages, and acts as the universal client for any web application. The use of web application frameworks can often reduce the number of errors in a program, both by making the code simpler, and by allowing one team to concentrate on the framework while another focuses on a specifified use case. In applications which are exposed to constant hacking attempts on the Internet, security- related problems can be caused by errors in the program.

Frameworks can also promote the use of best practices such as GET after POST. There are some who view a web application as a two-tier architecture. This can be a “smart” client that performs all the work and queries a “dumb” server, or a “dumb” client that relies on a “smart” server. The client would handle the presentation tier, the server would have the database (storage tier), and the business logic (application tier) would be on one of them or on both. While this increases the scalability of the applications and separates the display and the database, it still doesn‟t allow for true specialization of layers, so most applications will outgrow this model. An emerging strategy for application software companies is to provide web access to software previously distributed as local applications. Depending on the type of application, it may require the development of an entirely different browser-based interface, or merely adapting an existing application to use different presentation technology. These programs allow the user to pay a monthly or yearly fee for use of a software application without having to install it on a local hard drive. A company which follows this strategy is known as an application service provider (ASP), and ASPs are currently receiving much attention in the software industry.

Security breaches on these kinds of applications are a major concern because it can involve both enterprise information and private customer data. Protecting these assets is an important part of any web application and there are some key operational areas that must be included in the development process. This includes processes for authentication, authorization, asset handling, input, and logging and auditing. Building security into the applications from the beginning can be more effective and less disruptive in the long run.

# Web design

It is encompasses many different skills and disciplines in the production and maintenance of websites. The different areas of web design include web graphic design; interface design; authoring, including standardized code and proprietary software; user experience design; an

search engine optimization. The term web design is normally used to describe the design process relating to the front-end (client side) design of a website including writing mark up. Web design partially overlaps web engineering in the broader scope of web development. Web designers are expected to have an awareness of usability and if their role involves creating mark up then they are also expected to be up to date with web accessibility guidelines. Web design partially overlaps web engineering in the broader scope of web development.

## Departments and services offered

Compsoft Technologies plays an essential role as an institute, the level of education, development of student’s skills are based on their trainers. If you do not have a good mentor then you may lag in many things from others and that is why we at Compsoft Technologies gives you the facility of skilled employees so that you do not feel unsecured about the academics. Personality development and academic status are some of those things which lie on mentor’s hands. If you are trained well then you can do well in your future and knowing its importance of Compsoft Technologies always tries to give you the best.

They have a great team of skilled mentors who are always ready to direct their trainees in the best possible way they can and to ensure the skills of mentors we held many skill development programs as well so that each and every mentor can develop their own skills with the demands of the companies so that they can prepare a complete packaged trainee.

## Services provided by Compsoft Technologies.

* Core Java and Advanced Java
* Web services and development
* Dot Net Framework
* Python
* Selenium Testing
* Conference / Event Management Service
* Academic Project Guidance
* On The Job Training
* Software Training

**CHAPTER :3**

**INTRODUCTION**

# Introduction to ML

According to a report from the United Nations , an increasing number of people will interact with a voice assistance machine than with their partners in the next five years. With proliferation of Virtual Personal Assistants (VPA) such as Siri, Alexa and Google Assistant in our day-to-day interactions, they fill a role of answering our questions and fulfilling our requests quickly and accurately. Though these assistants understand our commands, they are not proficient enough in recognizing our mood and reacting accordingly. Therefore, it is pertinent to develop an efficient emotion recognition system which can enhance the capabilities of these assistants and revolutionize the whole industry. Speech is a rich, dense form of communication that can convey information effectively. It contains two types of information, namely linguistic and paralinguistic. The former refers to the verbal content, the underlying language code, while the latter refers to the implicit information such as body language, gestures, facial expressions, tone, pitch, emotion etc. Paralinguistic characteristics can help understand the mental state of the person (emotion), gender, attitude, dialect, and more . Recorded speech has key features that can be leveraged to extract information, such as emotion, in a structured way. To get such information would be invaluable in facilitating more natural conversations between the virtual assistant and the user since emotion color everyday human interactions. There are two widely used representations of emotion: continuous and discrete. In the continuous representation, the emotion of an utterance can be expressed as continuous values along multiple psychological dimensions. According to Ayadi, Kamel, & Karray , “emotion can be characterized in two dimensions: activation and valence.” Activation is the “amount of energy required to express a certain emotion” and research has shown that joy, anger, and fear can be linked to high energy and pitch in speech, whereas sadness can be linked to low energy and slow speech. Valence gives more nuance and helps distinguish between emotions like being angry and happy since increased activation can indicate both . In the discrete representation, emotions can be discretely expressed as specific categories, such as angry, sad, happy, etc.

# Problem Statement:

Voice Classification using ML that analyses the sentiment behind the tone of the voice and predicts the sentiment involved

# [CHAPTER :](https://1.bp.blogspot.com/-dODuK8N5h1Q/Wlnyb3V9HFI/AAAAAAAACL4/WxQtCJ1pM5wccDABg4wIrTBUB0vlikXQQCLcBGAs/s1600/poly1.jpg) 4

# SYSTEM ANALYSIS

## Existing System

## C:\Users\Chaitra\Pictures\Screenshots\Screenshot (1078).png

## C:\Users\Chaitra\Pictures\Screenshots\Screenshot (1079).png

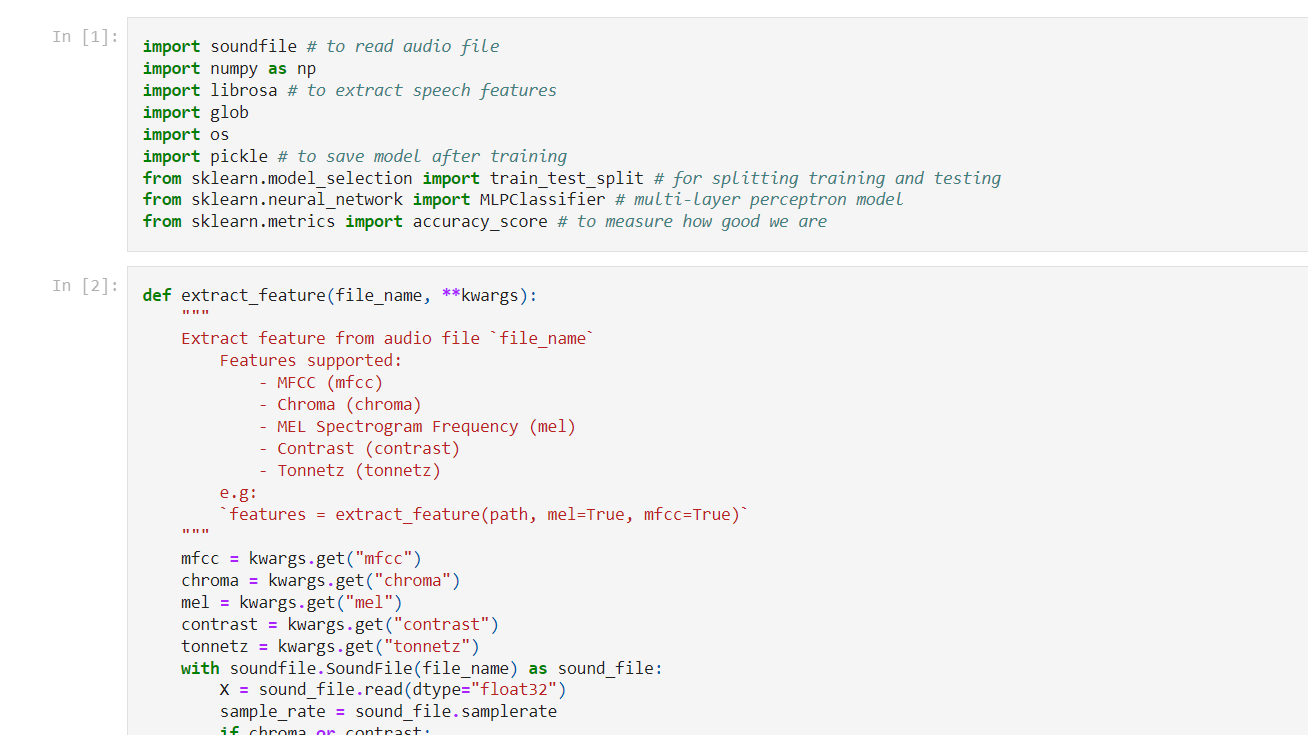
**CODE For** Voice Classification using Machine learning”

* **"""**
* **A utility script used for converting audio samples to be**
* **suitable for feature extraction**
* **"""**
* **import os**
* **def convert\_audio(audio\_path, target\_path, remove=False):**
  + **"""This function sets the audio `audio\_path` to:**
* **16000Hz Sampling rate**
* **one audio channel ( mono )**
  + - **Params:**
      * **audio\_path (str): the path of audio wav file you want to convert**
      * **target\_path (str): target path to save your new converted wav file**
      * **remove (bool): whether to remove the old file after converting**
    - **Note that this function requires ffmpeg installed in your system."""**
  + **os.system(f"ffmpeg -i {audio\_path} -ac 1 -ar 16000 {target\_path}")**
  + **# os.system(f"ffmpeg -i {audio\_path} -ac 1 {target\_path}")**
  + **if remove:**
    - **os.remove(audio\_path)**
* **def convert\_audios(path, target\_path, remove=False):**
  + **"""Converts a path of wav files to:**
* **16000Hz Sampling rate**
* **one audio channel ( mono )**
  + - **and then put them into a new folder called `target\_path`**
    - **Params:**
      * **audio\_path (str): the path of audio wav file you want to convert**
      * **target\_path (str): target path to save your new converted wav file**
      * **remove (bool): whether to remove the old file after converting**
    - **Note that this function requires ffmpeg installed in your system."""**
  + **for dirpath, dirnames, filenames in os.walk(path):**
    - **for dirname in dirnames:**
    - **dirname = os.path.join(dirpath, dirname)**
    - **target\_dir = dirname.replace(path, target\_path)**
    - **if not os.path.isdir(target\_dir):**
      * **os.mkdir(target\_dir)**
  + **for dirpath, \_, filenames in os.walk(path):**
    - **for filename in filenames:**
    - **file = os.path.join(dirpath, filename)**
    - **if file.endswith(".wav"):**
      * **# it is a wav file**
      * **target\_file = file.replace(path, target\_path)**
      * **convert\_audio(file, target\_file, remove=remove)**
* **if \_\_name\_\_ == "\_\_main\_\_":**
  + **import argparse**
  + **parser = argparse.ArgumentParser(description="""Convert ( compress ) wav files to 16MHz and mono audio channel ( 1 channel )**

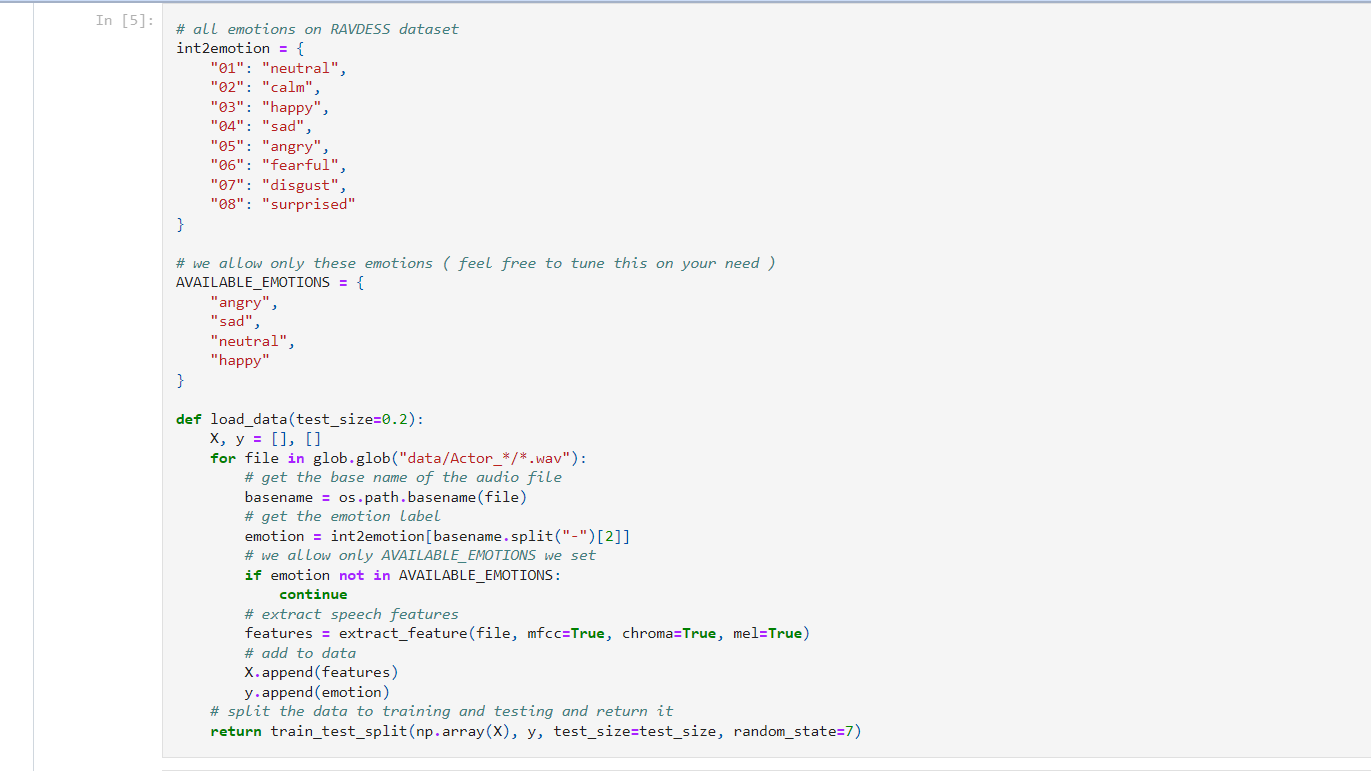
**This utility helps for compressing wav files for training and testing""")**

* + **parser.add\_argument("audio\_path", help="Folder that contains wav files you want to convert")**
  + **parser.add\_argument("target\_path", help="Folder to save new wav files")**
  + **parser.add\_argument("-r", "--remove", type=bool, help="Whether to remove the old wav file after converting", default=False)**
  + **args = parser.parse\_args()**
  + **audio\_path = args.audio\_path**
  + **target\_path = args.target\_path**
  + **if os.path.isdir(audio\_path):**
    - **if not os.path.isdir(target\_path):**
    - **os.makedirs(target\_path)**
    - **convert\_audios(audio\_path, target\_path, remove=args.remove)**
  + **elif os.path.isfile(audio\_path) and audio\_path.endswith(".wav"):**
    - **if not target\_path.endswith(".wav"):**
    - **target\_path += ".wav"**
    - **convert\_audio(audio\_path, target\_path, remove=args.remove)**
  + **else:**
    1. **raise TypeError("The audio\_path file you specified isn't appropriate for this operation")**

[**Voice-Classification-and-Emotion-Recognization-Using-ML**](https://github.com/WebdeveloperIsaac/Voice-Classification-and-Emotion-Recognization-Using-ML)**/model**

****

****

****

****

**We Are going to build a speech emotion detection classifier.**

But first we need to learn about what is speech recognition (SER) and why are we building this project? Well, few of the reasons are-

First, lets define SER i.e. Speech Emotion Recognition.

Speech Emotion Recognition, abbreviated as SER, is the act of attempting to recognize human emotion and affective states from speech. This is capitalizing on the fact that voice often reflects underlying emotion through tone and pitch. This is also the phenomenon that animals like dogs and horses employ to be able to understand human emotion.

**Why we need it?**

Emotion recognition is the part of speech recognition which is gaining more popularity and need for it increases enormously. Although there are methods to recognize emotion using machine learning techniques, this project attempts to use deep learning to recognize the emotions from data.

SER(Speech Emotion Recognition) is used in call center for classifying calls according to emotions and can be used as the performance parameter for conversational analysis thus identifying the unsatisfied customer, customer satisfaction and so on.. for helping companies improving their services

It can also be used in-car board system based on information of the mental state of the driver can be provided to the system to initiate his/her safety preventing accidents to happen

**Datasets used in this project**

Crowd-sourced Emotional Mutimodal Actors Dataset (Crema-D)

Ryerson Audio-Visual Database of Emotional Speech and Song (Ravdess)

Surrey Audio-Visual Expressed Emotion (Savee)

Toronto emotional speech set (Tess)

Importing Libraries

In [1]:

import pandas as pd

import numpy as np

import os

import sys

# 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.

import librosa

import librosa.display

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.metrics import confusion\_matrix, classification\_report

from sklearn.model\_selection import train\_test\_split

# to play the audio files

from IPython.display import Audio

import keras

from keras.callbacks import ReduceLROnPlateau

from keras.models import Sequential

from keras.layers import Dense, Conv1D, MaxPooling1D, Flatten, Dropout, BatchNormalization

from keras.utils import np\_utils, to\_categorical

from keras.callbacks import ModelCheckpoint

import warnings

if not sys.warnoptions:

warnings.simplefilter("ignore")

warnings.filterwarnings("ignore", category=DeprecationWarning)

**Data Preparation**

As we are working with four different datasets, so i will be creating a dataframe storing all emotions of the data in dataframe with their paths.

We will use this dataframe to extract features for our model training.

In [ ]:

# Paths for data.

Ravdess = "/kaggle/input/ravdess-emotional-speech-audio/audio\_speech\_actors\_01-24/"

Crema = "/kaggle/input/cremad/AudioWAV/"

Tess = "/kaggle/input/toronto-emotional-speech-set-tess/tess toronto emotional speech set data/TESS Toronto emotional speech set data/"

Savee = "/kaggle/input/surrey-audiovisual-expressed-emotion-savee/ALL/"

1. **Ravdess Dataframe**

Here is the filename identifiers as per the official RAVDESS website:

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).

So, here's an example of an audio filename. 02-01-06-01-02-01-12.mp4 This means the meta data for the audio file is:

Video-only (02)

Speech (01)

Fearful (06)

Normal intensity (01)

Statement "dogs" (02)

1st Repetition (01)

12th Actor (12) - Female (as the actor ID number is even)

In [ ]:

ravdess\_directory\_list = os.listdir(Ravdess)

file\_emotion = []

file\_path = []

for dir in ravdess\_directory\_list:

# as their are 20 different actors in our previous directory we need to extract files for each actor.

actor = os.listdir(Ravdess + dir)

for file in actor:

part = file.split('.')[0]

part = part.split('-')

# third part in each file represents the emotion associated to that file.

file\_emotion.append(int(part[2]))

file\_path.append(Ravdess + dir + '/' + file)

# dataframe for emotion of files

emotion\_df = pd.DataFrame(file\_emotion, columns=['Emotions'])

# dataframe for path of files.

path\_df = pd.DataFrame(file\_path, columns=['Path'])

Ravdess\_df = pd.concat([emotion\_df, path\_df], axis=1)

# changing integers to actual emotions.

Ravdess\_df.Emotions.replace({1:'neutral', 2:'calm', 3:'happy', 4:'sad', 5:'angry', 6:'fear', 7:'disgust', 8:'surprise'}, inplace=True)

Ravdess\_df.head()

**2. Crema DataFrame**

In [ ]:

crema\_directory\_list = os.listdir(Crema)

file\_emotion = []

file\_path = []

for file in crema\_directory\_list:

# storing file paths

file\_path.append(Crema + file)

# storing file emotions

part=file.split('\_')

if part[2] == 'SAD':

file\_emotion.append('sad')

elif part[2] == 'ANG':

file\_emotion.append('angry')

elif part[2] == 'DIS':

file\_emotion.append('disgust')

elif part[2] == 'FEA':

file\_emotion.append('fear')

elif part[2] == 'HAP':

file\_emotion.append('happy')

elif part[2] == 'NEU':

file\_emotion.append('neutral')

else:

file\_emotion.append('Unknown')

# dataframe for emotion of files

emotion\_df = pd.DataFrame(file\_emotion, columns=['Emotions'])

# dataframe for path of files.

path\_df = pd.DataFrame(file\_path, columns=['Path'])

Crema\_df = pd.concat([emotion\_df, path\_df], axis=1)

Crema\_df.head()

**3. STRESS dataset**

In [ ]:

tess\_directory\_list = os.listdir(Tess)

file\_emotion = []

file\_path = []

for dir in tess\_directory\_list:

directories = os.listdir(Tess + dir)

for file in directories:

part = file.split('.')[0]

part = part.split('\_')[2]

if part=='ps':

file\_emotion.append('surprise')

else:

file\_emotion.append(part)

file\_path.append(Tess + dir + '/' + file)

# dataframe for emotion of files

emotion\_df = pd.DataFrame(file\_emotion, columns=['Emotions'])

# dataframe for path of files.

path\_df = pd.DataFrame(file\_path, columns=['Path'])

Tess\_df = pd.concat([emotion\_df, path\_df], axis=1)

Tess\_df.head()

**4. CREMA-D dataset**

The audio files in this dataset are named in such a way that the prefix letters describes the emotion classes as follows:

'a' = 'anger'

'd' = 'disgust'

'f' = 'fear'

'h' = 'happiness'

'n' = 'neutral'

'sa' = 'sadness'

'su' = 'surprise'

In [ ]:

savee\_directory\_list = os.listdir(Savee)

file\_emotion = []

file\_path = []

for file in savee\_directory\_list:

file\_path.append(Savee + file)

part = file.split('\_')[1]

ele = part[:-6]

if ele=='a':

file\_emotion.append('angry')

elif ele=='d':

file\_emotion.append('disgust')

elif ele=='f':

file\_emotion.append('fear')

elif ele=='h':

file\_emotion.append('happy')

elif ele=='n':

file\_emotion.append('neutral')

elif ele=='sa':

file\_emotion.append('sad')

else:

file\_emotion.append('surprise')

# dataframe for emotion of files

emotion\_df = pd.DataFrame(file\_emotion, columns=['Emotions'])

# dataframe for path of files.

path\_df = pd.DataFrame(file\_path, columns=['Path'])

Savee\_df = pd.concat([emotion\_df, path\_df], axis=1)

Savee\_df.head()

In [ ]:

# creating Dataframe using all the 4 dataframes we created so far.

data\_path = pd.concat([Ravdess\_df, Crema\_df, Tess\_df, Savee\_df], axis = 0)

data\_path.to\_csv("data\_path.csv",index=False)

data\_path.head()

Data Visualisation and Exploration

First let's plot the count of each emotions in our dataset.

In [ ]:

plt.title('Count of Emotions', size=16)

sns.countplot(data\_path.Emotions)

plt.ylabel('Count', size=12)

plt.xlabel('Emotions', size=12)

sns.despine(top=True, right=True, left=False, bottom=False)

plt.show()

We can also plot waveplots and spectograms for audio signals

Waveplots - Waveplots let us know the loudness of the audio at a given time.

Spectograms - A spectrogram is a visual representation of the spectrum of frequencies of sound or other signals as they vary with time. It’s a representation of frequencies changing with respect to time for given audio/music signals.

In [ ]:

def create\_waveplot(data, sr, e):

plt.figure(figsize=(10, 3))

plt.title('Waveplot for audio with {} emotion'.format(e), size=15)

librosa.display.waveplot(data, sr=sr)

plt.show()

def create\_spectrogram(data, sr, e):

# stft function converts the data into short term fourier transform

X = librosa.stft(data)

Xdb = librosa.amplitude\_to\_db(abs(X))

plt.figure(figsize=(12, 3))

plt.title('Spectrogram for audio with {} emotion'.format(e), size=15)

librosa.display.specshow(Xdb, sr=sr, x\_axis='time', y\_axis='hz')

#librosa.display.specshow(Xdb, sr=sr, x\_axis='time', y\_axis='log')

plt.colorbar()

In [ ]:

emotion='fear'

path = np.array(data\_path.Path[data\_path.Emotions==emotion])[1]

data, sampling\_rate = librosa.load(path)

create\_waveplot(data, sampling\_rate, emotion)

create\_spectrogram(data, sampling\_rate, emotion)

Audio(path)

In [ ]:

emotion='angry'

path = np.array(data\_path.Path[data\_path.Emotions==emotion])[1]

data, sampling\_rate = librosa.load(path)

create\_waveplot(data, sampling\_rate, emotion)

create\_spectrogram(data, sampling\_rate, emotion)

Audio(path)

In [ ]:

emotion='sad'

path = np.array(data\_path.Path[data\_path.Emotions==emotion])[1]

data, sampling\_rate = librosa.load(path)

create\_waveplot(data, sampling\_rate, emotion)

create\_spectrogram(data, sampling\_rate, emotion)

Audio(path)

In [ ]:

emotion='happy'

path = np.array(data\_path.Path[data\_path.Emotions==emotion])[1]

data, sampling\_rate = librosa.load(path)

create\_waveplot(data, sampling\_rate, emotion)

create\_spectrogram(data, sampling\_rate, emotion)

Audio(path)

Data Augmentation

Data augmentation is the process by which we create new synthetic data samples by adding small perturbations on our initial training set.

To generate syntactic data for audio, we can apply noise injection, shifting time, changing pitch and speed.

The objective is to make our model invariant to those perturbations and enhace its ability to generalize.

In order to this to work adding the perturbations must conserve the same label as the original training sample.

In images data augmention can be performed by shifting the image, zooming, rotating ...

First, let's check which augmentation techniques works better for our dataset.

In [ ]:

def noise(data):

noise\_amp = 0.035\*np.random.uniform()\*np.amax(data)

data = data + noise\_amp\*np.random.normal(size=data.shape[0])

return data

def stretch(data, rate=0.8):

return librosa.effects.time\_stretch(data, rate)

def shift(data):

shift\_range = int(np.random.uniform(low=-5, high = 5)\*1000)

return np.roll(data, shift\_range)

def pitch(data, sampling\_rate, pitch\_factor=0.7):

return librosa.effects.pitch\_shift(data, sampling\_rate, pitch\_factor)

# taking any example and checking for techniques.

path = np.array(data\_path.Path)[1]

data, sample\_rate = librosa.load(path)

1**. Simple Audio**

In [ ]:

plt.figure(figsize=(14,4))

librosa.display.waveplot(y=data, sr=sample\_rate)

Audio(path)

2**. Noise Injection**

In [ ]:

x = noise(data)

plt.figure(figsize=(14,4))

librosa.display.waveplot(y=x, sr=sample\_rate)

Audio(x, rate=sample\_rate)

We can see noise injection is a very good augmentation technique because of which we can assure our training model is not overfitted

3. **Stretching**

In [ ]:

x = stretch(data)

plt.figure(figsize=(14,4))

librosa.display.waveplot(y=x, sr=sample\_rate)

Audio(x, rate=sample\_rate)

**4. Shifting**

In [ ]:

x = shift(data)

plt.figure(figsize=(14,4))

librosa.display.waveplot(y=x, sr=sample\_rate)

Audio(x, rate=sample\_rate)

**5. Pitch**

In [ ]:

x = pitch(data, sample\_rate)

plt.figure(figsize=(14,4))

librosa.display.waveplot(y=x, sr=sample\_rate)

Audio(x, rate=sample\_rate)

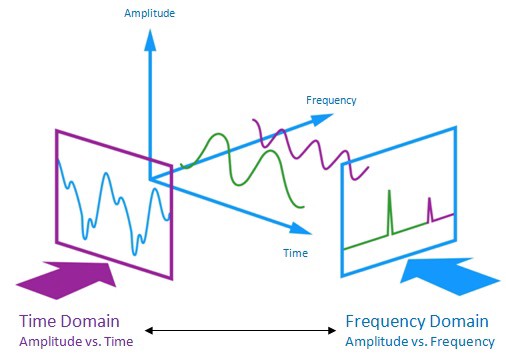
From the above types of augmentation techniques i am using noise, stretching(ie. changing speed) and

some pitching.

Feature Extraction

Extraction of features is a very important part in analyzing and finding relations between different things. As we already know that the data provided of audio cannot be understood by the modelsdirectly so we need to convert them into an understandable format for which feature extraction is used.

The audio signal is a three-dimensional signal in which three axes represent time, amplitude and frequency.



I am no expert on audio signals and feature extraction on audio files so i need to search and found a very good blog written by **[Askash Mallik](https://medium.com/heuristics/audio-signal-feature-extraction-and-clustering-935319d2225)**on feature extraction.As stated there with the help of the sample rate and the sample data, one can perform several transformations on it to extract valuable features out of it.

* Zero Crossing Rate : The rate of sign-changes of the signa l during the duration of a particular frame.
* Energy : The sum of squares of the signal values, normalized by the respective frame length.
* Entropy of Energy : The entropy of sub-frames’ normalized energies. It can be interpreted as a measure of abrupt changes.
* Spectral Centroid : The center of gravity of the spectrum.
* Spectral Spread : The second central moment of the spectrum.
* Spectral Entropy : Entropy of the normalized spectral energies for a set of sub-frames.
* Spectral Flux : The squared difference between the normalized magnitudes of the spectra of the two successive frames.
* Spectral Rolloff : The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
* Chroma Vector : A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-type music (semitone spacing).

Chroma Deviation : The standard deviation of the 12 chroma coefficients.

In this project i am not going deep in feature selection process to check which features are good for our dataset rather i am only extracting 5 features:

**Zero Crossing Rate**

Chroma\_stft

MFCC

RMS(root mean square) value

MelSpectogram to train our model.

In [ ]:

def extract\_features(data):

# ZCR

result = np.array([])

zcr = np.mean(librosa.feature.zero\_crossing\_rate(y=data).T, axis=0)

result=np.hstack((result, zcr)) # stacking horizontally

# Chroma\_stft

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

chroma\_stft = np.mean(librosa.feature.chroma\_stft(S=stft, sr=sample\_rate).T, axis=0)

result = np.hstack((result, chroma\_stft)) # stacking horizontally

# MFCC

mfcc = np.mean(librosa.feature.mfcc(y=data, sr=sample\_rate).T, axis=0)

result = np.hstack((result, mfcc)) # stacking horizontally

# Root Mean Square Value

rms = np.mean(librosa.feature.rms(y=data).T, axis=0)

result = np.hstack((result, rms)) # stacking horizontally

# MelSpectogram

mel = np.mean(librosa.feature.melspectrogram(y=data, sr=sample\_rate).T, axis=0)

result = np.hstack((result, mel)) # stacking horizontally

return result

def get\_features(path):

# duration and offset are used to take care of the no audio in start and the ending of each audio files as seen above.

data, sample\_rate = librosa.load(path, duration=2.5, offset=0.6)

# without augmentation

res1 = extract\_features(data)

result = np.array(res1)

# data with noise

noise\_data = noise(data)

res2 = extract\_features(noise\_data)

result = np.vstack((result, res2)) # stacking vertically

# data with stretching and pitching

new\_data = stretch(data)

data\_stretch\_pitch = pitch(new\_data, sample\_rate)

res3 = extract\_features(data\_stretch\_pitch)

result = np.vstack((result, res3)) # stacking vertically

return result

In [ ]:

X, Y = [], []

for path, emotion in zip(data\_path.Path, data\_path.Emotions):

feature = get\_features(path)

for ele in feature:

X.append(ele)

# appending emotion 3 times as we have made 3 augmentation techniques on each audio file.

Y.append(emotion)

In [ ]:

len(X), len(Y), data\_path.Path.shape

In [ ]:

Features = pd.DataFrame(X)

Features['labels'] = Y

Features.to\_csv('features.csv', index=False)

Features.head()

We have applied data augmentation and extracted the features for each audio files and saved them.

Data Preparation

As of now we have extracted the data, now we need to normalize and split our data for training and testing.

In [ ]:

X = Features.iloc[: ,:-1].values

Y = Features['labels'].values

In [ ]:

# As this is a multiclass classification problem onehotencoding our Y.

encoder = OneHotEncoder()

Y = encoder.fit\_transform(np.array(Y).reshape(-1,1)).toarray()

In [ ]:

# splitting data

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, random\_state=0, shuffle=True)

x\_train.shape, y\_train.shape, x\_test.shape, y\_test.shape

In [ ]:

# scaling our data with sklearn's Standard scaler

scaler = StandardScaler()

x\_train = scaler.fit\_transform(x\_train)

x\_test = scaler.transform(x\_test)

x\_train.shape, y\_train.shape, x\_test.shape, y\_test.shape

In [ ]:

# making our data compatible to model.

x\_train = np.expand\_dims(x\_train, axis=2)

x\_test = np.expand\_dims(x\_test, axis=2)

x\_train.shape, y\_train.shape, x\_test.shape, y\_test.shape

Modelling

In [ ]:

model=Sequential()

model.add(Conv1D(256, kernel\_size=5, strides=1, padding='same', activation='relu', input\_shape=(x\_train.shape[1], 1)))

model.add(MaxPooling1D(pool\_size=5, strides = 2, padding = 'same'))

model.add(Conv1D(256, kernel\_size=5, strides=1, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=5, strides = 2, padding = 'same'))

model.add(Conv1D(128, kernel\_size=5, strides=1, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=5, strides = 2, padding = 'same'))

model.add(Dropout(0.2))

model.add(Conv1D(64, kernel\_size=5, strides=1, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=5, strides = 2, padding = 'same'))

model.add(Flatten())

model.add(Dense(units=32, activation='relu'))

model.add(Dropout(0.3))

model.add(Dense(units=8, activation='softmax'))

model.compile(optimizer = 'adam' , loss = 'categorical\_crossentropy' , metrics = ['accuracy'])

model.summary()

In [ ]:

rlrp = ReduceLROnPlateau(monitor='loss', factor=0.4, verbose=0, patience=2, min\_lr=0.0000001)

history=model.fit(x\_train, y\_train, batch\_size=64, epochs=50, validation\_data=(x\_test, y\_test), callbacks=[rlrp])

In [ ]:

print("Accuracy of our model on test data : " , model.evaluate(x\_test,y\_test)[1]\*100 , "%")

epochs = [i for i in range(50)]

fig , ax = plt.subplots(1,2)

train\_acc = history.history['accuracy']

train\_loss = history.history['loss']

test\_acc = history.history['val\_accuracy']

test\_loss = history.history['val\_loss']

fig.set\_size\_inches(20,6)

ax[0].plot(epochs , train\_loss , label = 'Training Loss')

ax[0].plot(epochs , test\_loss , label = 'Testing Loss')

ax[0].set\_title('Training & Testing Loss')

ax[0].legend()

ax[0].set\_xlabel("Epochs")

ax[1].plot(epochs , train\_acc , label = 'Training Accuracy')

ax[1].plot(epochs , test\_acc , label = 'Testing Accuracy')

ax[1].set\_title('Training & Testing Accuracy')

ax[1].legend()

ax[1].set\_xlabel("Epochs")

plt.show()

In [ ]:

# predicting on test data.

pred\_test = model.predict(x\_test)

y\_pred = encoder.inverse\_transform(pred\_test)

y\_test = encoder.inverse\_transform(y\_test)

In [ ]:

df = pd.DataFrame(columns=['Predicted Labels', 'Actual Labels'])

df['Predicted Labels'] = y\_pred.flatten()

df['Actual Labels'] = y\_test.flatten()

df.head(10)

In [ ]:

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize = (12, 10))

cm = pd.DataFrame(cm , index = [i for i in encoder.categories\_] , columns = [i for i in encoder.categories\_])

sns.heatmap(cm, linecolor='white', cmap='Blues', linewidth=1, annot=True, fmt='')

plt.title('Confusion Matrix', size=20)

plt.xlabel('Predicted Labels', size=14)

plt.ylabel('Actual Labels', size=14)

plt.show()

In [ ]:

print(classification\_report(y\_test, y\_pred))

We can see our model is more accurate in predicting surprise, angry emotions and it makes sense also because audio files of these emotions differ to other audio files in a lot of ways like pitch, speed etc..

We overall achieved 61% accuracy on our test data and its decent but we can improve it more by applying more augmentation techniques and using other feature extraction methods.

This Concludes Our InternShip Project for Voice Classification using ML that analyses the sentiment behind the tone of the voice and predicts the sentiment invovled from the Dataset

## TEST [Voice-Classification-and-Emotion-Recognization-Using-ML](https://github.com/WebdeveloperIsaac/Voice-Classification-and-Emotion-Recognization-Using-ML)

|  |
| --- |
| import pyaudio |
|  | import os |
|  | import wave |
|  | import pickle |
|  | from sys import byteorder |
|  | from array import array |
|  | from struct import pack |
|  | from sklearn.neural\_network import MLPClassifier |
|  |  |
|  | from utils import extract\_feature |
|  |  |
|  | THRESHOLD = 500 |
|  | CHUNK\_SIZE = 1024 |
|  | FORMAT = pyaudio.paInt16 |
|  | RATE = 16000 |
|  |  |
|  | SILENCE = 30 |
|  |  |
|  | def is\_silent(snd\_data): |
|  | "Returns 'True' if below the 'silent' threshold" |
|  | return max(snd\_data) < THRESHOLD |
|  |  |
|  | def normalize(snd\_data): |
|  | "Average the volume out" |
|  | MAXIMUM = 16384 |
|  | times = float(MAXIMUM)/max(abs(i) for i in snd\_data) |
|  |  |
|  | r = array('h') |
|  | for i in snd\_data: |
|  | r.append(int(i\*times)) |
|  | return r |
|  |  |
|  | def trim(snd\_data): |
|  | "Trim the blank spots at the start and end" |
|  | def \_trim(snd\_data): |
|  | snd\_started = False |
|  | r = array('h') |
|  |  |
|  | for i in snd\_data: |
|  | if not snd\_started and abs(i)>THRESHOLD: |
|  | snd\_started = True |
|  | r.append(i) |
|  |  |
|  | elif snd\_started: |
|  | r.append(i) |
|  | return r |
|  |  |
|  | # Trim to the left |
|  | snd\_data = \_trim(snd\_data) |
|  |  |
|  | # Trim to the right |
|  | snd\_data.reverse() |
|  | snd\_data = \_trim(snd\_data) |
|  | snd\_data.reverse() |
|  | return snd\_data |
|  |  |
|  | def add\_silence(snd\_data, seconds): |
|  | "Add silence to the start and end of 'snd\_data' of length 'seconds' (float)" |
|  | r = array('h', [0 for i in range(int(seconds\*RATE))]) |
|  | r.extend(snd\_data) |
|  | r.extend([0 for i in range(int(seconds\*RATE))]) |
|  | return r |
|  |  |
|  | def record(): |
|  | """ |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# [CHAPTER :](https://1.bp.blogspot.com/-dODuK8N5h1Q/Wlnyb3V9HFI/AAAAAAAACL4/WxQtCJ1pM5wccDABg4wIrTBUB0vlikXQQCLcBGAs/s1600/poly1.jpg) 5 REQUIREMENT ANALYSIS

|  |  |  |  |
| --- | --- | --- | --- |
| **ALGORITHM** | **APPROAC H 1** | **APPROAC H 2** | **APPROAC H 3** |
| SVM | **83%** | **77%** | **90%** |
| Decision  Tree | **71%** | **65%** | **68%** |
| KNN | **80%** | **80%** | **87%** |
| Logistic Regressin | **70%** | **77%** | **86%** |
| Random Forest | **76%** | **69%** | **72%** |
| Gaussian Naïve  Bayes | **74%** | **73%** | **75%** |
| Gradient Boostig  Trees | **77%** | **69%** | **75%** |

**REQUIREMENT ANALYSIS**

# Hardware Requirement Specification

* + Processor : Intel core i5 processer
  + Memory : 15.6 GB
  + Hard Disk : 40 GB
  + 8 GB of RAM
  + OS : Windows 8 or later

# Software Requirement Specification

* + - Programming Language : Python
    - IDE : Visual Studio Code
    - Open source dataset

# [CHAPTER](https://1.bp.blogspot.com/-dODuK8N5h1Q/Wlnyb3V9HFI/AAAAAAAACL4/WxQtCJ1pM5wccDABg4wIrTBUB0vlikXQQCLcBGAs/s1600/poly1.jpg) 6

* 1. **DESIGN & ANALYSIS**

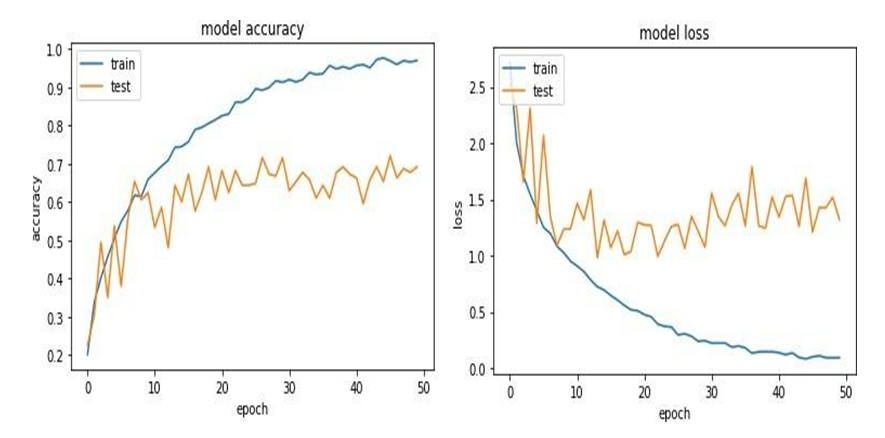
1D CNNs and 2D CNNs were implemented on 6 emotion class classification and 12 gender+emotionclass classification parallelly and it was observed that including gender gave better performance. Since CNNs are natural feature extractors, 1D CNN and 1D CNN-LSTM architectures were trained on the raw audio input.

2D CNNs were implemented on the engineered features such as MFCCs and Log-mel spectrogram. The training of 2D CNNs started with 2 convolu×tional layers with 3 3 filters and max pooling with

2 filters with stride 2. They were tuned by adding more convolution layers and increasing thefilter sizes in the initial layers. It was found that increasing the depth beyond 4 layers did not improve performance. Also the champion model on 14 class prediction was obtained with 12 × 12 filters and

7 × 7 filters in the first and second layers respectively. Also, the final 2 layers had 3 × 3

filters.



# [CHAPTER](https://1.bp.blogspot.com/-dODuK8N5h1Q/Wlnyb3V9HFI/AAAAAAAACL4/WxQtCJ1pM5wccDABg4wIrTBUB0vlikXQQCLcBGAs/s1600/poly1.jpg) : 7

# IMPLEMENTATION

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively.

The system can be implemented only after thorough testing is done and if it is found to work according to the specification. It involves careful planning, investigation of the current system and it constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods a part from planning.

Two major tasks of preparing the implementation are education and training of the users and testing of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required just for implementation.

The implementation phase comprises of several activities. The required hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

TESTING

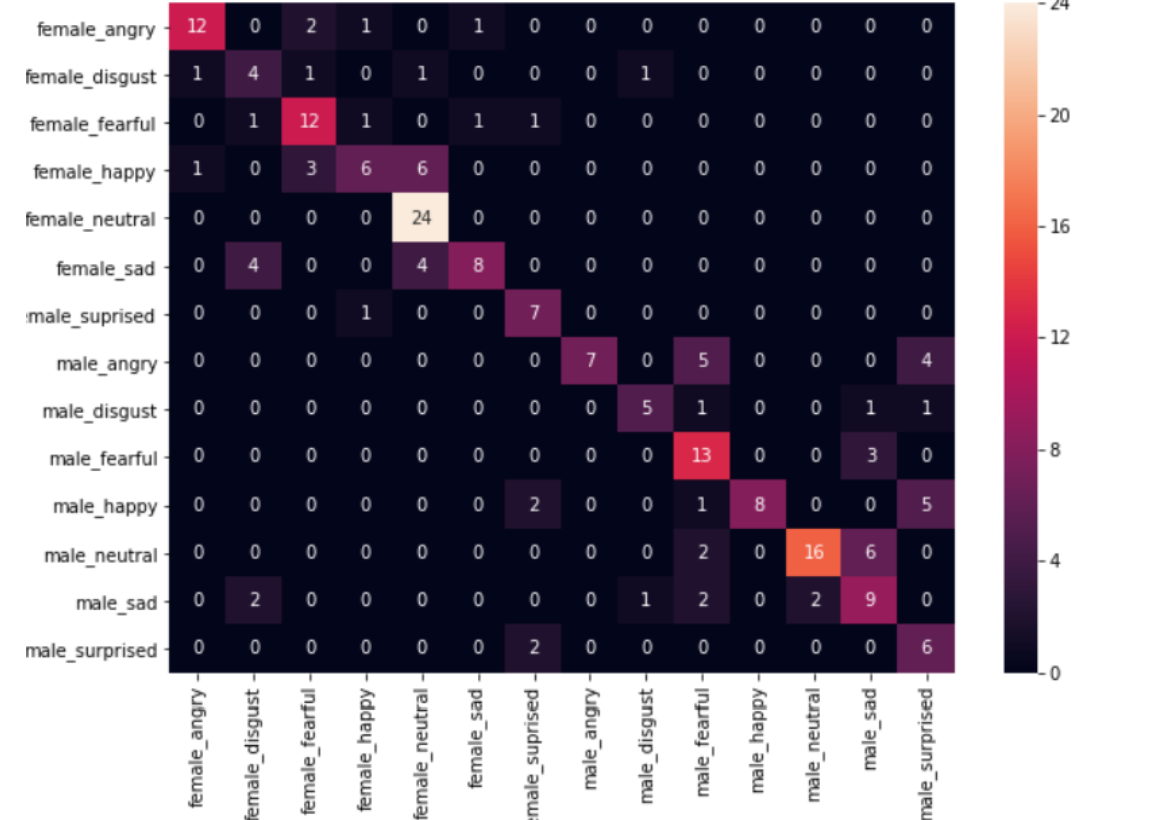
The testing phase is an important part of software development. It is the Information zed system will help in automate process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. Software testing is carried out in three steps:

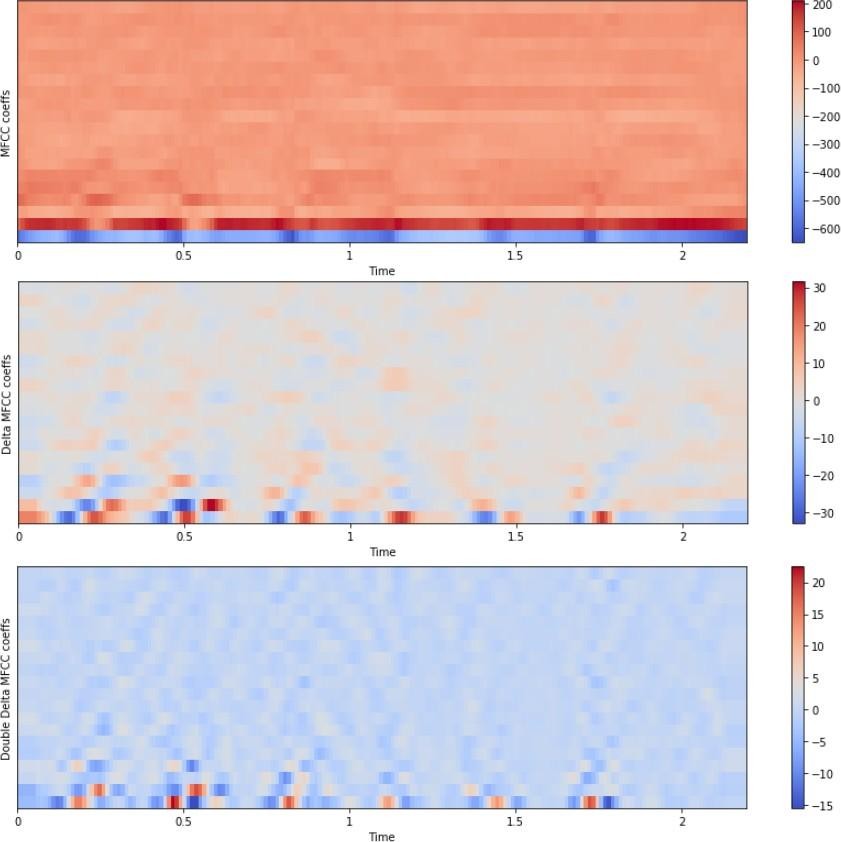
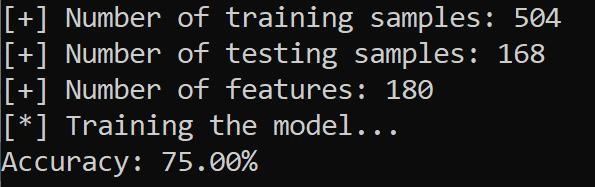
The first includes unit testing, where in each module is tested to provide its correctness, validity and also determine any missing operations and to verify whether theobjectives have been met. Errors are noted down and corrected immediately.

Unit testing is the important and major part of the project. So errors are rectified easily in particular module and program clarity is increased. In this project entire system is dividedinto several modules and is developed individually. So unit testing is conducted to individual modules.

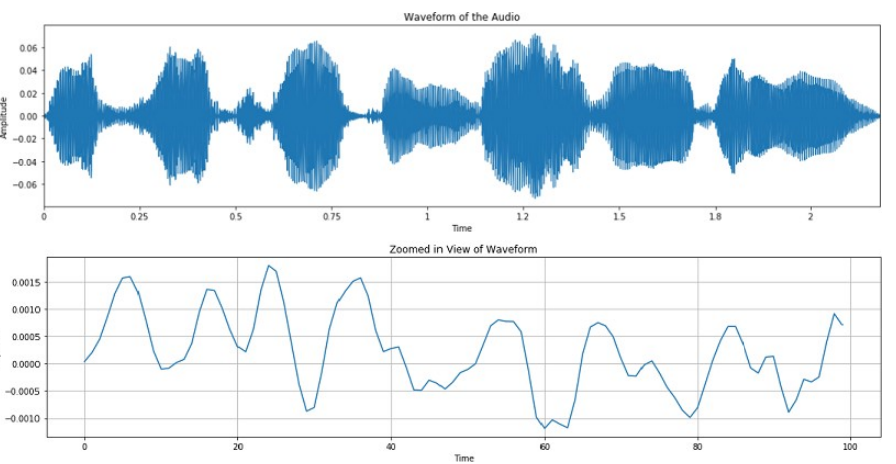
The second step includes Integration testing. It need not be the case, the software whose modules when run individually and showing perfect results, will also show perfect results when run as a whole.

* 1. **SNAPSHOTS**





Dependencies Used: Librosa Numpy Soundfile Scikit-learn PyAudio

Preparing the Dataset: Here, we download and convert the dataset to be suited for extraction. Loading the Dataset: This process is about loading the dataset in Python which involves extracting audio features, such as obtaining different features such as power, pitch, and vocal tract configuration from the speech signal, we will use librosa library to do that. [](https://github.com/WebdeveloperIsaac/Voice-Classification-and-Emotion-Recognization-Using-ML/blob/main/Snapshots/wave.png)**Training the Model: After we prepare and load the dataset, we simply train it on a suited sklearn model.**

**CONCLUSION**

The emerging growth and development in the field of AI and machine learning have led to the new era of automation. Most of these automated devices work based on voice commands from the user. Many advantages can be built over the existing systems if besides recognizing the words, the machines could comprehend the emotion of the speaker (user). Some applications of a speech emotion detection systemare computer-based tutorial applications, automated call center conversations, a diagnostic tool used for therapy and automatic translation system.

In this thesis, the steps of building a speech emotion detection system were discussed in detail and some experiments were carried out to understand the impact of each step. Initially, the limited number of publically available speech database made it challenging to implement a well-trained model. Next, several novel approaches to feature extraction had been proposed in the earlier works, and selecting the best approach included performing many experiments. Finally, the classifier selection involved learning aboutthe strength and weakness of each classifying algorithm with respect to emotion recognition. At the end of the experimentation, it can be concluded that an integrated feature space will produce a better recognition rate when compared to a single feature.

For future advancements, the proposed project can be further modeled in terms of efficiency, accuracy, and usability. Additional to the emotions, the model can be extended to recognize feelings such as depression and mood changes. Such systems can be used by therapists to monitor the mood swings of the patients. A challenging product of creating machines with emotion is to incorporate a sarcasm detection system. Sarcasm detection is a more complex problem of emotion detection since sarcasm cannot be easily identified using only the words or tone of the speaker.

# REFERENCE

* *Ericsdotter C and Ericsson A M 2001 Gender differences in vowel duration in read Swedish: Preliminary results Working Papers - Lund University Department of Linguistics 34 – 37*
* *Whiteside S P 1996 Temporal-Based Acoustic-Phonetic Patterns in Read Speech: Some Evidence for Speaker Sex Differences, J International Phonetic Association* ***26*** *23–40*
* *Byrd D 1992 Preliminary results on speaker-dependent variation in the TIMIT database*

*J AcoustSoc Am* ***92*** *593–596*

* *Henton C G 1989 Fact and fiction in the description of male and female pitch Language and Communication* ***9*** *299-311*
* *Bishop J and Keating P 2012 Perception of pitch location within a speaker’s range: Fundamental Frequency, voice quality and speaker sex The Journal of the Acoustical Society of America* ***32-2*** *1100-1112*
* *Smith D R and Patterson R D 2005 The interaction of glottal-pulse rate and vocal-tract length in judgments of speaker size, sex, and age in The Journal of the Acoustical Society of America,* ***118-5*** *3177-3186*
* *Muhammad G, AlSulaiman M, Mahmood A and Ali Z 2011 Automatic voice disorder classification using vowel formants, 2011 Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '11) 1–6*
* *Gaikwad S, Gawali B, and Mehrotra S C 2012 Gender identification using SVM with combination of MFCC, Advances in Computational Research* ***4*** *69–73*
* *Zeng Y M, Wu Z Y, Falk T and Chan W Y 2006 Robust GMM based gender classification using pitch and RASTA-PLP parameters of speech Proceedings of the International Conference on Machine Learning and Cybernetics 3376–3379*