VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi-590018

INTERNSHIP REPORT

ON

"Voice Classification using Machine learning"

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

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by:

MOUNESH ALTI
3NA19CS018

Conducted at

COMPSOFT TECHNOLOGIES

NAVODAYA INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2022-2022

NAVODAYA INSTITUTE OF TECHNOLOGY
Department Of Computer Science
RAICHUR

CERTIFICATE

This is to certify that the Internship titled **"Voice Classification Using Machine Learning "** carried out by **Mr. Mounesh Alti**, a bonafide student of Navodaya Institute Of Technology, in partial fulfilment for the award of **Bachelor of Engineering**, in **Computer Science and Engineering** under Visvesvaraya Technological University, Belagavi, during the year 2022-2023. 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 respect of

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

_____ _____

_____ _____

DECLARATION

I, **MOUNESH ALTI,** final year student of Computer Science and Engineering, Navodaya Institute Of Technology - 584103, declare that the Internship has been successfully completed, in **COMPSOFT TECHNOLOGIES.** This report is submitted in partial fulfilment of the requirements for award of Bachelor Degree in Computer Science and Engineering, during the academic year 2022-2023.

Date : 2/10/2022

Place : Raichur

USN : 3NA19CS018

NAME : MOUNESH ALTI

OFFER LETTER

A C K N O W L E D G E M E N T

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, for providing us an opportunity to carry out Internship and for his valuable guidance and support.

We would like to thank our Software Services for guiding us during the period of internship.

We express our deep and profound gratitude to our guide, for his 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, for helping 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: MOUNESH ALTI

USN:3NA19CS018

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 systemsavailable 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-basedinput.

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

CHAPTER 1

COMPANY PROFILE

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, and Conference Management, effective web promotion and tailor-made software products, designing solutions best suiting client's 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 define 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 calibre 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 2

ABOUT THE COMPANY

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; and 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 markup 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

3. INTRODUCTION

Introduction to ML:

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more.

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of Machine Learning.

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem.

Problem Statement:

In today's advanced hi-tech environment, the need for self-sufficiency is recognized in the situation of visually impaired people who are socially restricted. They are in an unfamiliar environment and are unable to help themselves.

Because most tasks require visual information, visually impaired people are at a disadvantage because crucial information about their surroundings is unavailable. It is now possible to extend the support provided to people with visual impairments thanks to recent advancements in inclusive technology.

This project proposes to use Artificial Intelligence, Machine Learning, Image and Text Recognition to assist persons who are blind or visually impaired. The concept is realized using various python modules that includes features such as voice assistant, image recognition, currency recognition, e-book, and chat bot. The software can recognize items in the environment using voice commands and do text analysis to recognize text in a hard copy document.

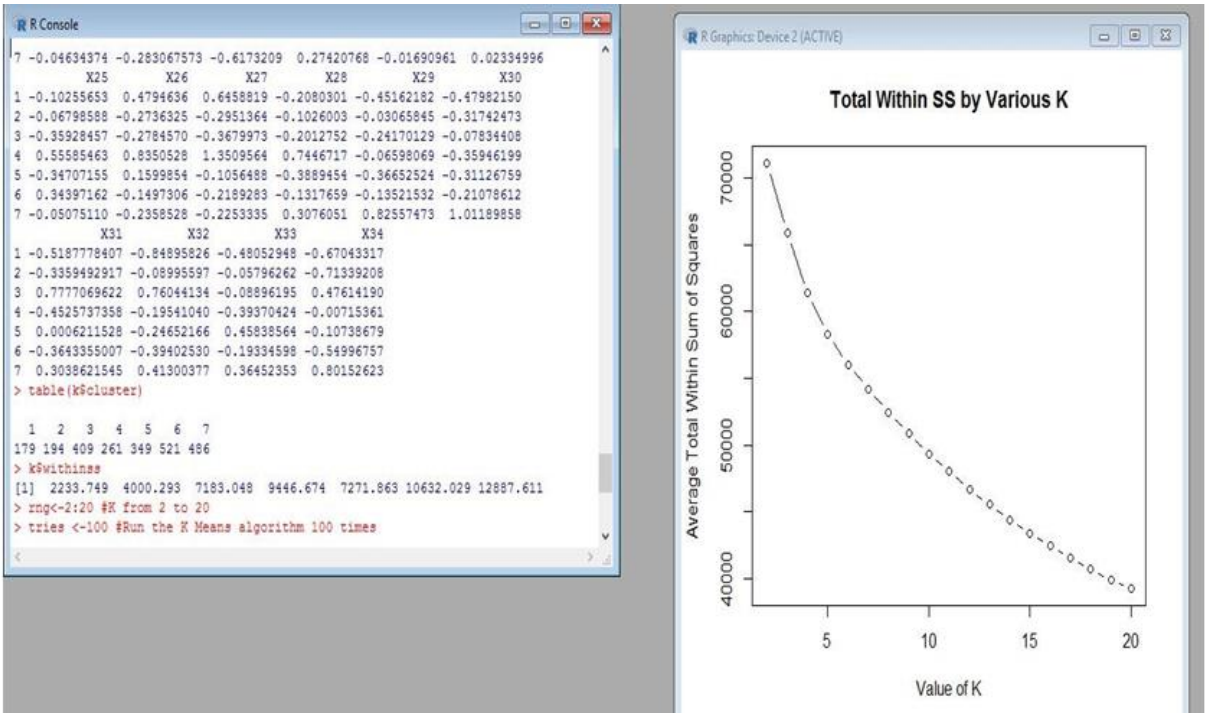
It will be an effective approach for blind individuals to engage with the world and make use of technology's features. The goal of this project is to design a virtual assistant to guide a visually impaired person as well as improve the accuracy of the model using various datasets and machine learning algorithms.

CHAPTER 4

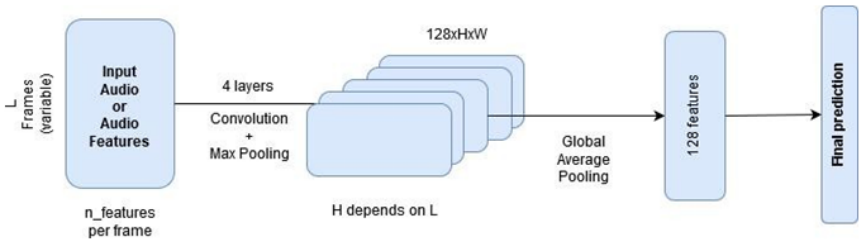
SYSTEM ANALYSIS

SYSTEM ANALYSIS

1. Existing System



2. Proposed System



3. Objective of the System

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

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):
    o """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."""

    o os.system(f"ffmpeg -i {audio_path} -ac 1 -ar 16000 {target_path}")
    o # os.system(f"ffmpeg -i {audio_path} -ac 1 {target_path}")
    o if remove:
        ▪ os.remove(audio_path)


- def convert_audios(path, target_path, remove=False):
    o """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."""

    o 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)

    o 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:
        i. raise TypeError("The audio_path file you specified isn't appropriate for this operation")

Voice-Classification-and-Emotion-Recognization-Using-ML/model

In [1]:

```
import soundfile # to read audio file
import numpy as np
import librosa # to extract speech features
import glob
import os
import pickle # to save model after training
from sklearn.model_selection import train_test_split # for splitting training and testing
from sklearn.neural_network import MLPClassifier # multi-layer perceptron model
from sklearn.metrics import accuracy_score # to measure how good we are
```

In [2]:

```
def extract_feature(file_name, **kwargs):
    """
    Extract feature from audio file `file_name`
        Features supported:
            - MFCC (mfcc)
            - Chroma (chroma)
            - MEL Spectrogram Frequency (mel)
            - Contrast (contrast)
            - Tonnetz (tonnetz)
        e.g:
        `features = extract_feature(path, mel=True, mfcc=True)`
    """
    mfcc = kwargs.get("mfcc")
    chroma = kwargs.get("chroma")
    mel = kwargs.get("mel")
    contrast = kwargs.get("contrast")
    tonnetz = kwargs.get("tonnetz")
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate = sound_file.samplerate
        if chroma or contrast:
            stft = np.abs(librosa.stft(X))
        result = np.array([])
        if mfcc:
            mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
            result = np.hstack((result, mfccs))
        if chroma:
            chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
            result = np.hstack((result, chroma))
        if mel:
            mel = np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
            result = np.hstack((result, mel))
        if contrast:
            contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=sample_rate).T,axis=0)
            result = np.hstack((result, contrast))
        if tonnetz:
            tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X), sr=sample_rate).T,axis=0)
            result = np.hstack((result, tonnetz))
    return result
```

In [5]:
```
# all emotions on RAVDESS dataset
int2emotion = {
    "01": "neutral",
    "02": "calm",
    "03": "happy",
    "04": "sad",
    "05": "angry",
    "06": "fearful",
    "07": "disgust",
    "08": "surprised"
}

# we allow only these emotions ( feel free to tune this on your need )
AVAILABLE_EMOTIONS = {
    "angry",
    "sad",
    "neutral",
    "happy"
}

def load_data(test_size=0.2):
    X, y = [], []
    for file in glob.glob("data/Actor_*/*.wav"):
        # get the base name of the audio file
        basename = os.path.basename(file)
        # get the emotion label
        emotion = int2emotion[basename.split("-")[2]]
        # we allow only AVAILABLE_EMOTIONS we set
        if emotion not in AVAILABLE_EMOTIONS:
            continue
        # extract speech features
        features = extract_feature(file, mfcc=True, chroma=True, mel=True)
        # add to data
        X.append(features)
        y.append(emotion)
    # split the data to training and testing and return it
    return train_test_split(np.array(X), y, test_size=test_size, random_state=7)
```

In [14]:
```
# load RAVDESS dataset, 75% training 25% testing
X_train, X_test, y_train, y_test = load_data(test_size=0.25)
```

In [15]:
```
# print some details
# number of samples in training data
print("[+] Number of training samples:", X_train.shape[0])
# number of samples in testing data
print("[+] Number of testing samples:", X_test.shape[0])
# number of features used
# this is a vector of features extracted
# using extract_features() function
print("[+] Number of features:", X_train.shape[1])
```

In [ ]:
```
# best model, determined by a grid search
model_params = {
    'alpha': 0.01,
    'batch_size': 256,
    'epsilon': 1e-08,
    'hidden_layer_sizes': (300,),
    'learning_rate': 'adaptive',
    'max_iter': 500,
}
```

In [ ]:
```
# initialize Multi Layer Perceptron classifier
# with best parameters ( so far )
model = MLPClassifier(**model_params)
```

In [ ]:
```
# train the model
print("[*] Training the model...")
model.fit(X_train, y_train)
```

In [ ]:
```
# predict 25% of data to measure how good we are
y_pred = model.predict(X_test)

# calculate the accuracy
accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
```

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. STESS 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 [ ]:

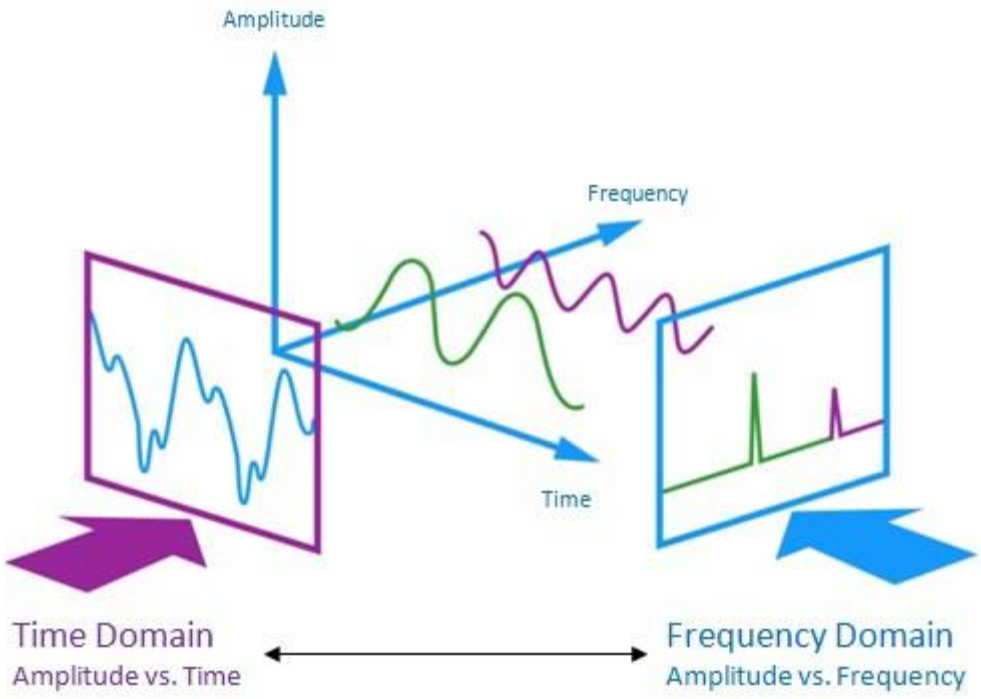
```
x = pitch(data, sample_rate)
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

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 models directly 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** 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 signal 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.

MFCCs Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.

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

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:
```

```
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():
```

""" Record a world or from the microphone and return the data as array of signal shorts normalizes the audio, trims silence from the start and end pads with 0.5 seconds of blank sound to make sure VLC et al can play without grtting chopped off.

"""

```
p = pyaudio.PyAudio()
stream = p.open(format=FORMAT, channels=1, rate=RATE,
input=True, output=True,
frames_per_buffer=CHUNK_SIZE)

num_silent = 0
snd_started = False

r = array('h')

while 1:
# little endian, signed short
snd_data = array('h', stream.read(CHUNK_SIZE))
if byteorder == 'big':
snd_data.byteswap()
r.extend(snd_data)

silent = is_silent(snd_data)

if silent and snd_started:
num_silent += 1
elif not silent and not snd_started:
snd_started = True

if snd_started and num_silent > SILENCE:
break

sample_width = p.get_sample_size(FORMAT)
stream.stop_stream()
stream.close()
```

```
p.terminate()

r = normalize(r)
r = trim(r)
r = add_silence(r, 0.5)
return sample_width, r      def record_to_file(path):


"Records from the microphone and outputs the resulting data to 'path'"
sample_width, data = record()
data = pack('<' + ('h'*len(data)), *data)

wf = wave.open(path, 'wb')
wf.setnchannels(1)
wf.setsampwidth(sample_width)

wf.setframerate(RATE)
wf.writeframes(data)
wf.close()

if __name__ == "__main__":
# load the saved model (after training)
model = pickle.load(open("result/mlp_classifier.model", "rb"))
print("Please talk")
filename = "test.wav"
# record the file (start talking)
record_to_file(filename)
# extract features and reshape it
Features=extract featuare(filename,mfcc=True,chroma=True,mel=true).reshape(1-1)

# predict
result = model.predict(features)[0]
# show the result !
print("result:", result)
```

CHAPTER 5

REQUIREMENT ANALYSIS

5.REQUIREMENT ANALYSIS

ALGORITHM	APPROACH 1	APPROACH 2	APPROACH 3
SVM	**83%**	**77 %**	**90%**
Decision Tree	**71%**	**65 %**	**68%**
KNN	**80%**	**80 %**	**87%**
Logistic Regression	**70%**	**77 %**	**86%**
Random Forest	**76%**	**69 %**	**72%**
GaussianNaïve Bayes	**74%**	**73 %**	**75%**
Gradient Boosting Trees	**77%**	**69 %**	**75%**

CHAPTER 6

DESIGN ANALYSIS

6.DESIGN ANALYSIS

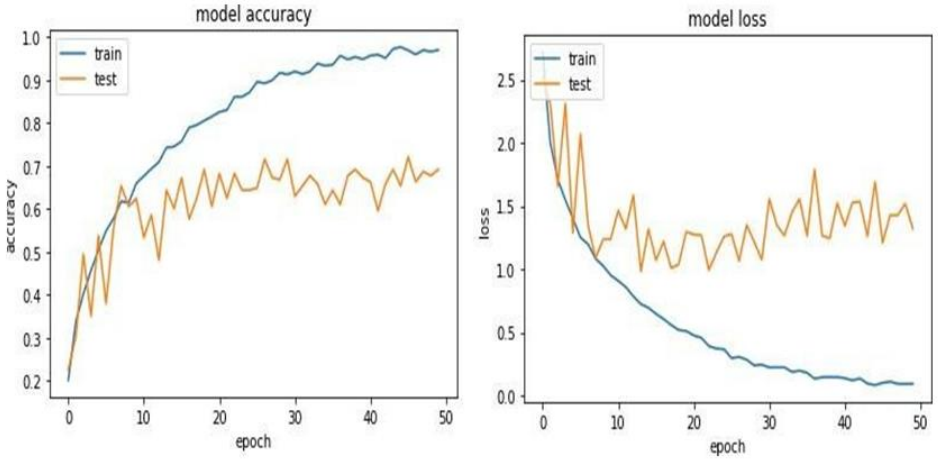
1D CNNs and 2D CNNs were implemented on 6 emotion class classification and 12 gender+emotionclass classification parallelly and it was observed that including gender gave better performance. Since CNNs are natural feature extractors, 1D CNN and 1D CNN-LSTM architectures were trained on the raw audio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

2D CNNs were implemented on the engineered features such as MFCCs and Log-mel spectrogram. The training of 2D CNNs started with 2 convolutional 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 7

IMPLEMENTATION

7.IMPLEMENTATION

Implementation is the stage where the theoretical design is turned into a working system. Themost 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 workaccording 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 andtesting 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 testedsystem 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 requirementsare 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.

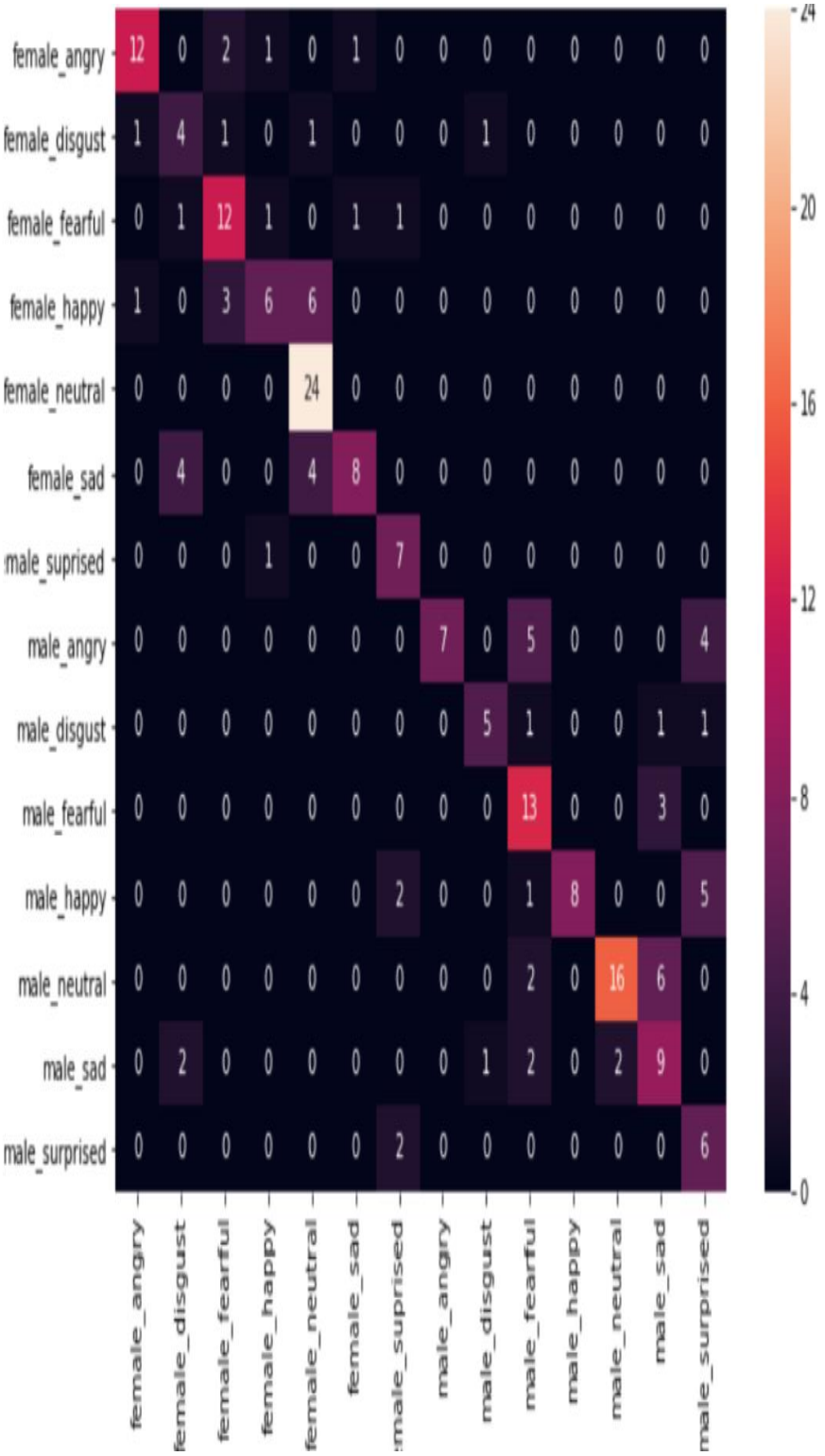
Unit testing is the important and major part of the project. So errors are rectified easily inparticular module and program clarity is increased. In this project entire system is dividedinto several modules and is developed individually. So unit testing is conducted to individual modules.

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

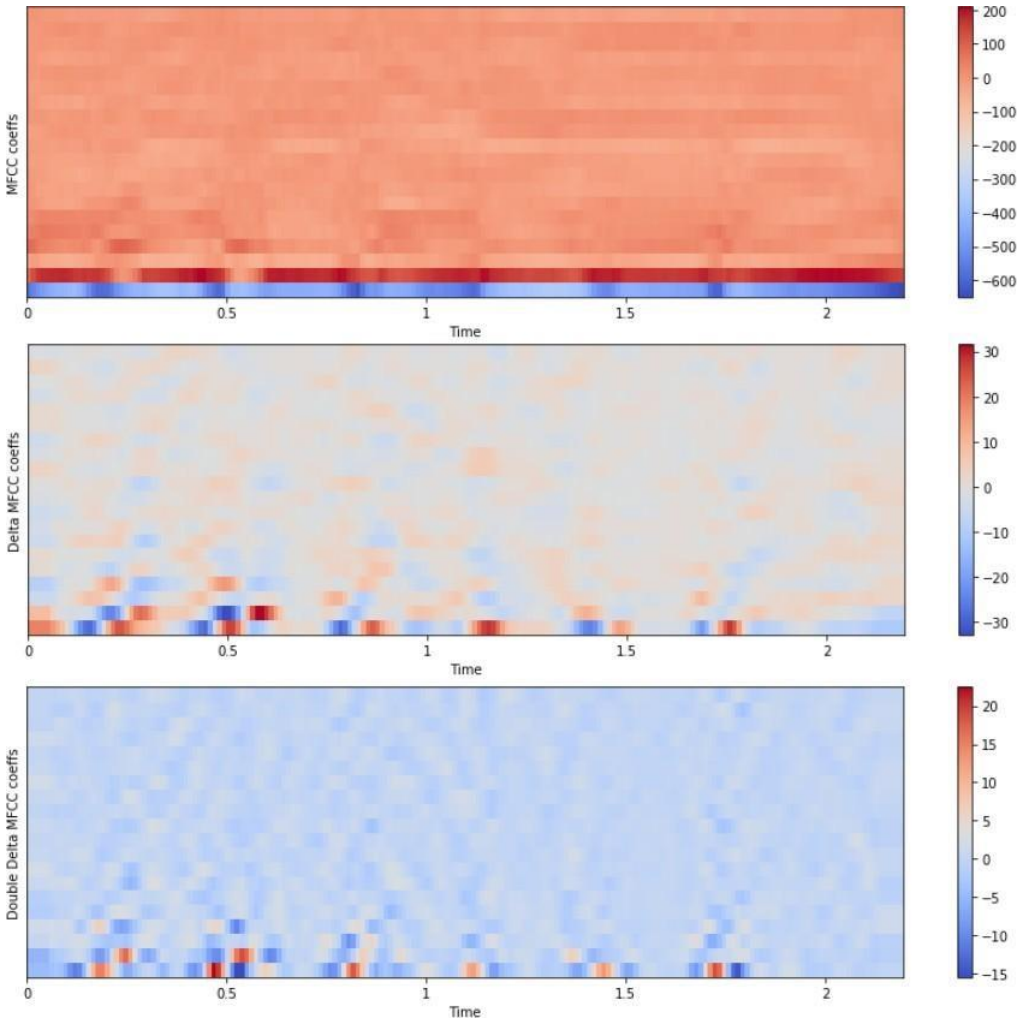
CHAPTER 8

SNAPSHOTS

8.SNAPSHOTS

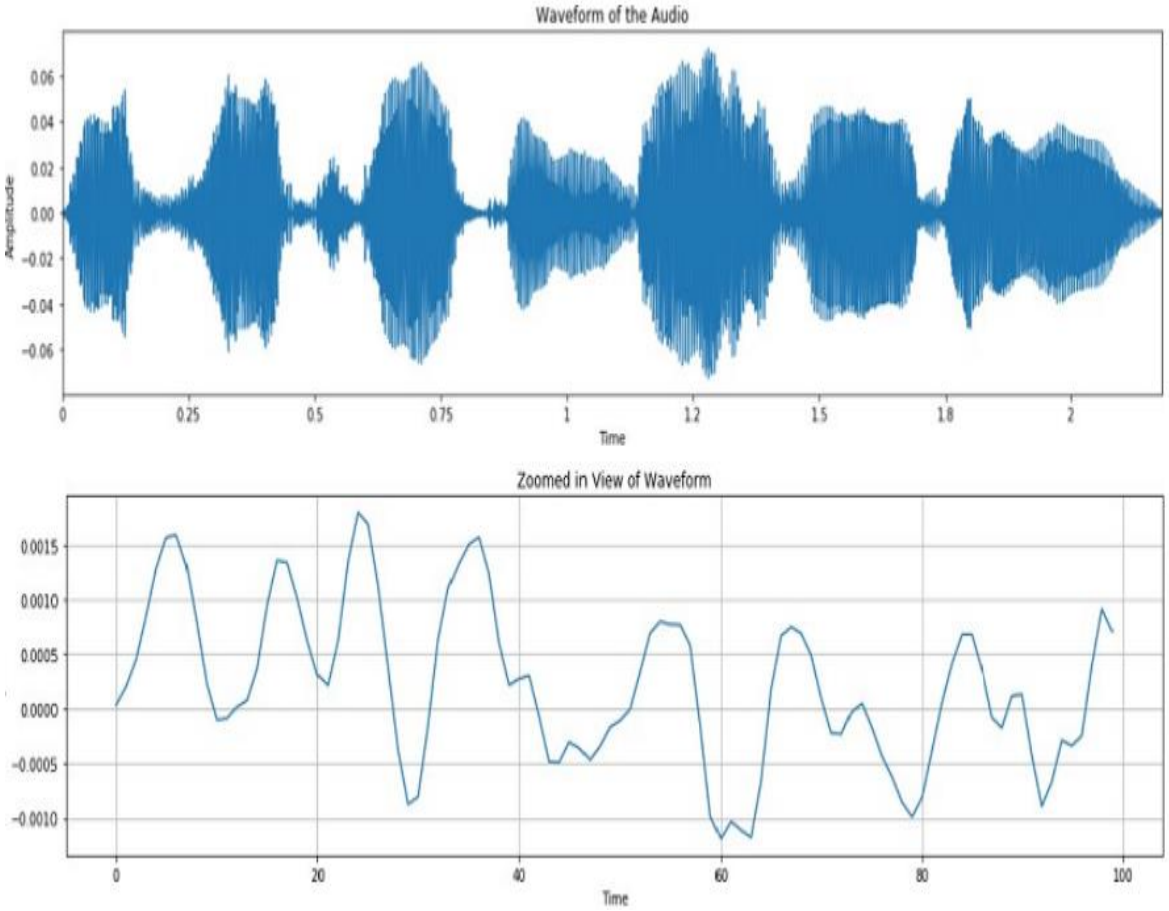

female_angry 12 0 2 1 0 1 0 0 0 0 0 0 0 0
female_disgust 1 4 1 0 1 0 0 0 1 0 0 0 0 0
female_fearful 0 1 12 1 0 1 1 0 0 0 0 0 0 0
female_happy 1 0 3 6 6 0 0 0 0 0 0 0 0 0
female_neutral 0 0 0 0 24 0 0 0 0 0 0 0 0 0
female_sad 0 4 0 0 4 8 0 0 0 0 0 0 0 0
male_suprised 0 0 0 1 0 0 7 0 0 0 0 0 0 0
male_angry 0 0 0 0 0 0 0 7 0 5 0 0 0 4
male_disgust 0 0 0 0 0 0 0 0 5 1 0 0 1 1
male_fearful 0 0 0 0 0 0 0 0 0 13 0 0 3 0
male_happy 0 0 0 0 0 0 2 0 0 1 8 0 0 5
male_neutral 0 0 0 0 0 0 0 0 0 2 0 16 6 0
male_sad 0 2 0 0 0 0 0 0 1 2 0 2 9 0
male_surprised 0 0 0 0 0 0 2 0 0 0 0 0 0 6
female_angry
female_disgust
female_fearful
female_happy
female_neutral
female_sad
male_suprised
male_angry
male_disgust
male_fearful
male_happy
male_neutral
male_sad
male_surprised
24
20
16
12
8
4
0

```
[+] Number of training samples: 504
[+] Number of testing samples: 168
[+] Number of features: 180
[*] Training the model...
Accuracy: 75.00%
```



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.



Training the Model: After we prepare and load the dataset, we simply train it on a suited sklearn model.

CHAPTER 9

CONCLUSION

9.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 thespeaker.

CHAPTER 10

REFERENCE

10.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*