

# **Women Safety Application**

A Report submitted in partial fulfilment of the requirement for the award  
of degree of

**Bachelor of Technology**

In

Electronics and Communication Engineering

Under the Supervision of

Name of the Supervisor – Dr. Meena Rao

**By**

Ankit Singh Patel (01415002820)

Abhishek Singh (00415002820)



**MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY**

C-4, Janakpuri, New Delhi-58

Affiliated to Guru Gobind Singh Indraprastha University, Delhi

Dec, 2023

# DECLARATION

We, student of B.Tech (Electronics & Communication Engineering) hereby declare that the project work done on “Women Safety Application ” submitted to Maharaja Surajmal Institute of Technology, Janakpuri Delhi in partial fulfilment of the requirement for the award of degree of Bachelor of Technology comprises of our original work and has not been submitted anywhere else for any other degree to the best of our knowledge.

*Ankit Singh Patel*  
(01415002820)

*Abhishek Singh*  
(00415002820)

## **CERTIFICATE**

This is to certify that the project work done on “*Women Safety Application*” submitted to Maharaja Surajmal Institute of Technology, Janakpuri Delhi by “ (1) Ankit Singh Patel (2) Abhishek Singh” in partial fulfillment of the requirement for the award of degree of Bachelor of Technology, is a bonafide work carried out by him/her under my supervision and guidance. This project work comprises of original work and has not been submitted anywhere else for any other degree to the best of my knowledge.

Signature of Supervisor

Signature of  
HOD

## ACKNOWLEDGEMENT

We extend our heartfelt appreciation to everyone who has contributed to the successful completion of our minor project on a Speaker Identification System during the seventh semester of our B.Tech (ECE) program at Maharaja Surajmal Institute of Technology.

As a collaborative team, Ankit Singh Patel, Abhishek Singh worked together with dedication, contributing their skills and efforts to enrich the overall quality of the project.

Special thanks are extended to our esteemed supervisor, Dr. Meena Rao, whose guidance and expertise played a crucial role in shaping our approach and refining the technical aspects of our work. Her mentorship has been invaluable, providing direction and encouragement throughout the project.

We express our gratitude to Dr. Neeru Rathi, Head of the Department (ECE), for continuous support and encouragement, fostering an environment that promotes learning and innovation within our department.

Sincere appreciation is also extended to Dr. Archana Balyan, Director of Maharaja Surajmal Institute of Technology, for providing an atmosphere that encourages academic excellence and research.

Lastly, we acknowledge our families and friends for their unwavering support during this project. Their understanding and encouragement have been vital to our academic journey.

This project has been a significant learning experience for all of us, and we are fortunate to have had the guidance and support of such esteemed individuals and institutions.

Ankit Singh Patel (01415002820)

Abhishek Singh (00415002820)

## Contents

Abstract	6
<b>Chapter 1: Introduction</b>	
1.1 Concept	7
1.2 Challenges	8
1.3 Technologies	8
<b>Chapter 2: Fundamentals of the Technology</b>	
2.1 Object detection and recognition	10
2.2 Classical machine learning models	12
2.3 Data Handling in audio domain	14
2.4 Convolutional neural network	
2.5 Feature extraction technique	15
2.6 Libraries	16
<b>Chapter 3: Proposed Methodology</b>	20
<b>Chapter 4: System analysis</b>	
4.1 Data dictionary	27
4.3 Screenshots	27
<b>Chapter 5: Result</b>	33
<b>Chapter-6: Conclusion and Future Scope of work</b>	35
<b>Bibliography</b>	38

## ABSTRACT

Audio information plays a rather important role in the increasing digital content that is available today; resulting in a need for methodologies that automatically analyze such content. Speaker Identification is one of the vital field of research based upon Voice Signals. Its other notable fields are: Speech Recognition, Speech-to-Text Conversion, and vice versa, etc. Mel Frequency Cepstral Coefficient (MFCC) is considered a key factor in performing Speaker Identification. But, there are other features lists available as an alternate to MFCC; like- Linear Predictor Coefficient (LPC), Spectrum Sub-band Centroid (SSC), Rhythm, Turbulence, Line Spectral Frequency (LPF), ChromaFactor, etc. Gaussian Mixture Model (GMM) is the most popular model for training on our data. The training task can also be executed on other significant models; viz. Hidden Markov Model (HMM). Recently, most of the model training phase for a speaker identification project is executed using Deep learning; especially, Artificial Neural Networks (ANN). In this project, we are mainly focused on implementing MFCC and GMM in pair to achieve our target.

We have considered MFCC with “tuned parameters” as the primary feature and delta-MFCC as secondary feature. And, we have implemented GMM with some tuned parameters to train our model. We have performed this project on two different kinds of Dataset; viz. “VoxForge” Dataset and a custom dataset which we have prepared by ourselves. We have obtained an outstanding result on both of these Datasets; viz. 100% accuracy on VoxForge Dataset and 95.29 % accuracy on self prepared Dataset. We demonstrate that speaker identification task can be performed using MFCC and GMM together with outstanding accuracy in Identification/ Diarization results.

**Keywords:** *Speaker Identification, LPC, LSF, Speaker Diarization, MFCC, HMM, GMM, VoxForge, ANN, SSC, Turbulence, ChromaFactor, etc.*

### 1.1 Concept

In a world where women's safety is constantly under threat, technology has emerged as a powerful ally. Audio processing, in particular, holds immense potential to protect women and empower them to live fearlessly.

Smartphones and wearable devices equipped with audio processing offer a range of features that can be lifesavers. These include:

- **Fall detection:** This feature can automatically detect falls and send out emergency alerts, ensuring help arrives quickly in case of an accident.
- **Real-time emergency alerts:** Apps and systems can leverage audio processing to identify dangerous situations, such as screams or cries for help, and immediately send out alerts to emergency services and designated contacts.
- **Voice-activated assistance:** Hands-free voice commands allow women to access help even when they are unable to physically use their devices. This can be especially crucial in situations where they are being threatened or attacked.

Beyond emergency response, audio processing empowers women in numerous ways:

- **Detecting and deterring stalking:** By analyzing audio patterns and identifying suspicious sounds, systems can alert authorities and deter potential stalkers.
- **Providing emotional support:** AI-powered chatbots and voice assistants can offer emotional support and guidance during stressful situations, helping women cope with anxiety and trauma.
- **Facilitating access to resources:** Audio-based information systems can provide women with access to resources and support services, empowering them to make informed decisions and access help whenever needed.

One of the most exciting applications of audio processing is the development of personal safety apps that use sophisticated algorithms to identify threats and take appropriate actions. These apps can analyze background noise, detect aggressive speech

patterns, and even recognize the sounds of weapons. In case of danger, they can automatically trigger alarms, send emergency alerts, and even record evidence.

## 1.2 Challenges

However, the use of audio processing technologies also raises important ethical concerns. Privacy is a major issue, as constantly monitoring audio data can have significant implications for individual privacy rights. Additionally, bias in algorithms can lead to discrimination against certain groups of women.

To ensure responsible and ethical use of audio processing for women's safety, it is crucial to:

- Develop clear and transparent policies regarding data collection, storage, and usage.
- Implement robust security measures to protect against data breaches and misuse.
- Involve diverse stakeholders in the development and deployment of these technologies.
- Conduct regular audits and assessments to identify and address potential biases.

## 1.3 Technology

Audio processing stands as a powerful tool in the fight for women's safety, and its effectiveness hinges on a robust foundation of machine learning models, libraries, and modules. These components work in tandem to analyze audio data, identify threats, and trigger appropriate responses, ultimately empowering women and safeguarding their well-being.

Several key machine learning models play crucial roles in this process:

- Sound event detection: Models like convolutional neural networks (CNNs) analyze audio data and classify sounds into various categories, including screams, cries for help, gunshots, and aggressive speech. This allows for the identification of potential threats and the initiation of appropriate actions.
- Speaker identification: By extracting unique features from a person's voice, models like Gaussian mixture models (GMMs) can identify who is speaking. This is particularly useful in recognizing potentially dangerous individuals or verifying the identity of someone in distress.
- Anomaly detection: Models like one-class support vector machines (OC-SVMs) can learn "normal" audio patterns and identify deviations from them. This



enables the detection of unusual sounds, which could indicate a dangerous situation or someone in need of assistance.

- Emotion recognition: Models like recurrent neural networks (RNNs) analyze speech patterns and intonation to identify emotions like fear, anger, or distress. This information can be invaluable for providing emotional support and tailoring responses to specific situations.

Libraries and modules form the backbone of these models, providing tools and functionalities for audio processing tasks. Some prominent examples include:

- Librosa: This Python library offers functionalities for audio and music analysis, including feature extraction, audio manipulation, and visualization. Its ease of use and comprehensive features make it a popular choice for developing audio processing applications.
- Scikit-learn: This widely used Python library provides a range of machine learning algorithms, including those used for sound event detection, anomaly detection, and speaker identification. Its versatility and extensive documentation make it a powerful tool for building ML models for audio processing.
- TensorFlow: This open-source software library provides high-performance tools for building and training complex machine learning models, including those used in audio processing. Its scalability and flexibility enable developers to tackle demanding tasks like speech recognition and natural language processing.
- PyAudio: This Python library allows for real-time audio input and output, enabling applications to interact with audio data directly. This is crucial for building applications that respond to sound events in real-time, such as emergency alert systems.

Modules like DeepSpeech and Vosk provide pre-trained models for specific tasks, such as automatic speech recognition. These readily available modules can be easily integrated into existing applications, reducing development time and effort.

The effectiveness of audio processing applications depends heavily on the quality of the underlying models, libraries, and modules. Constant research and development are crucial for improving the accuracy, efficiency, and robustness of these tools. Additionally, ensuring compatibility across different platforms and devices is essential to expand the reach of these applications and make them accessible to a wider audience.

As the field of audio processing for women's safety continues to evolve, we can expect to see even more innovative models, libraries, and modules emerge. By leveraging the power of machine learning and technology, we can create a safer world for women, one where they can live their lives free from fear and harm.

## Chapter: 2

# Fundamentals of Technology

---

Women's safety is a global concern, and technology offers promising avenues for creating safer environments. Artificial intelligence (AI) and machine learning (ML) are playing a crucial role in developing innovative applications that empower women and provide them with tools for protection.

While hunting for the best optimal approach to achieve our target, we tried and encountered with several blogs, publications, videos, modules, libraries etc. Here, we are stating few key points of them.

### 2.1 Object Detection and Recognition:

Object recognition is a general term to describe a collection of related computer vision tasks that involve identifying objects in digital photographs.

Image classification involves predicting the class of one object in an image. Object localization refers to identifying the location of one or more objects in an image and drawing abounding box around their extent. Object detection combines these two tasks and localizes and classifies one or more objects in an image.

When a user or practitioner refers to “object recognition“, they often mean “object detection“.

As such, we can distinguish between these three computer vision tasks:

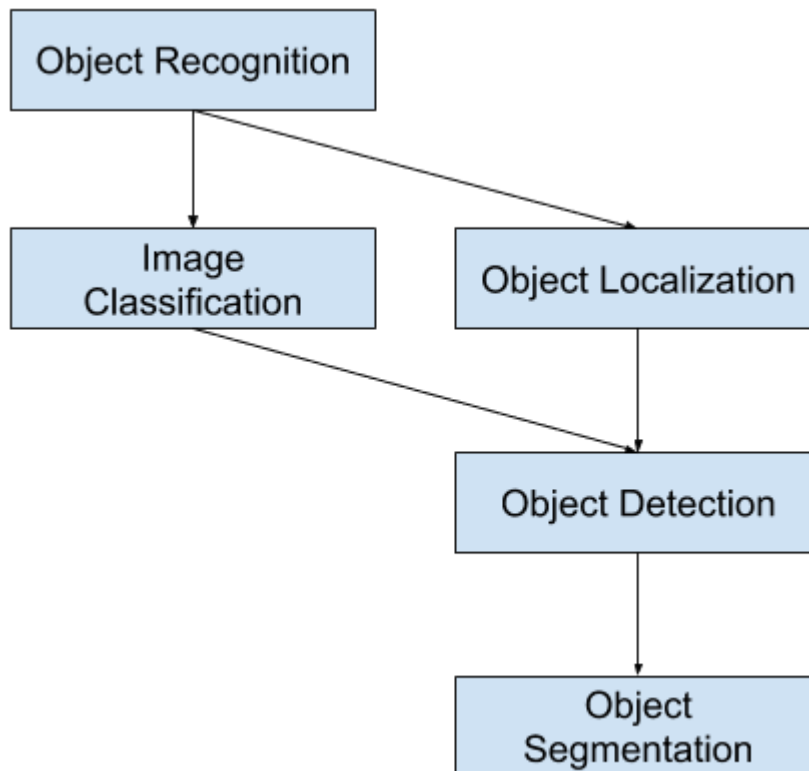
1. **Image Classification:** Predict the type or class of an object in an image.
  - Input: An image with a single object, such as a photograph.
  - Output: A class label (e.g. one or more integers that are mapped to class labels).

2. **Object Localization:** Locate the presence of objects in an image and indicate their location with a bounding box.

- Input: An image with one or more objects, such as a photograph.
- Output: One or more bounding boxes (e.g. defined by a point, width, and height).

3. **Object Detection:** Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

- Input: An image with one or more objects, such as a photograph.
- Output: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

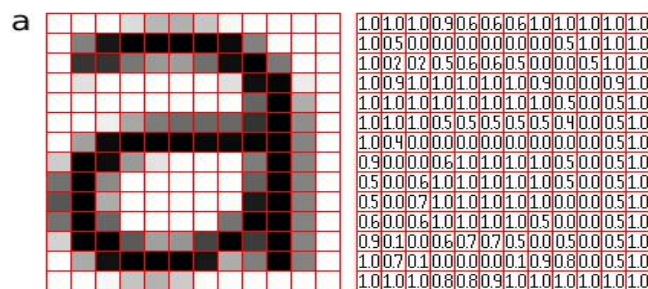


## 2.2 Classical Machine Learning Models

The classical machine learning models to train an audio dataset somehow revolves around Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM). A GMM (Gaussian mixture model) can be thought of as a single state HMM (Hidden Markov Model). In other words, a state in an HMM can be thought to have a mixture of distributions, with the probability of belonging to a distribution being represented by the emission probability (aka observation probability); each state in the HMM can have a unique set of emission probabilities. Therefore, each state in an HMM can be thought of as a GMM (with emission probabilities representing the probability of association to a distribution). Other methods include computing the out-of- sample log-likelihood upon the addition of each state and the number of states maximizing the log-likelihood is chosen as the reasonable number of states to work with. Though, GMM is preferably used because it is more reliable than HMM, although HMM possess more accuracy. Secondly, GMM yields output much faster than HMM. It consumes optimal resources than compared to HMM. Most of the times either of them are used independently or with a DNN (deep neural network). However, sometimes they both can be used together in combination. Choosing any of the above prescribed approach is Arts-more-than-Science. The Researcher's analytical abilities help him choose the better model for his project or work.

## 2.3 Convolutional Neural Network

A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.



The human brain processes a huge amount of information the second we see an image. Each neuron works in its own receptive field and is connected to other neurons in a way that they cover the entire visual field. Just as each neuron responds to stimuli only in the restricted region of the visual field called the receptive field in the biological vision system, each neuron in a CNN processes data only in its receptive field as well. The layers are arranged in such a way so that they detect simpler patterns first (lines, curves, etc.) and more complex patterns (faces, objects, etc.) further along. By using a CNN, one can enable sight to computers.

## **Layers used to build ConvNets**

A complete Convolution Neural Networks architecture is also known as convnets. A convnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

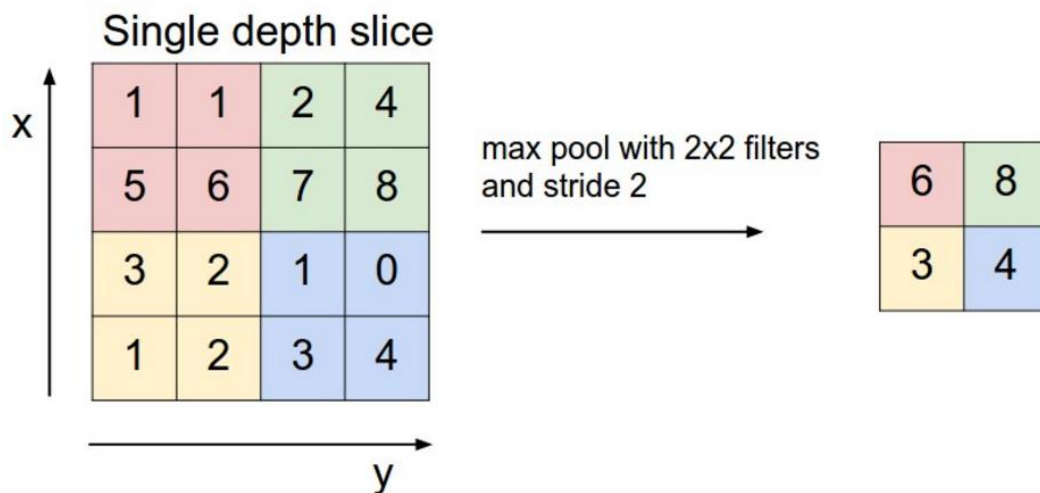
Types of layers: datasets

**Let's take an example by running a convnets on of image of dimension 32 x 32 x 3.**

- **Input Layers:** It's the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.
- **Convolutional Layers:** This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually  $2 \times 2$ ,  $3 \times 3$ , or  $5 \times 5$  shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred ad feature maps. Suppose we use a total of 12 filters for this layer we'll get an output volume of dimension  $32 \times 32 \times 12$ .
- **Activation Layer:** By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU:  $\max(0, x)$ , Tanh, Leaky RELU, etc.

The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.

- Pooling layer: This layer is periodically inserted in the convnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.
- Flattening: The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.
- Fully Connected Layers: It takes the input from the previous layer and computes the final classification or regression task.
- Output Layer: The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.



## 2.4 Data Handling in Audio domain

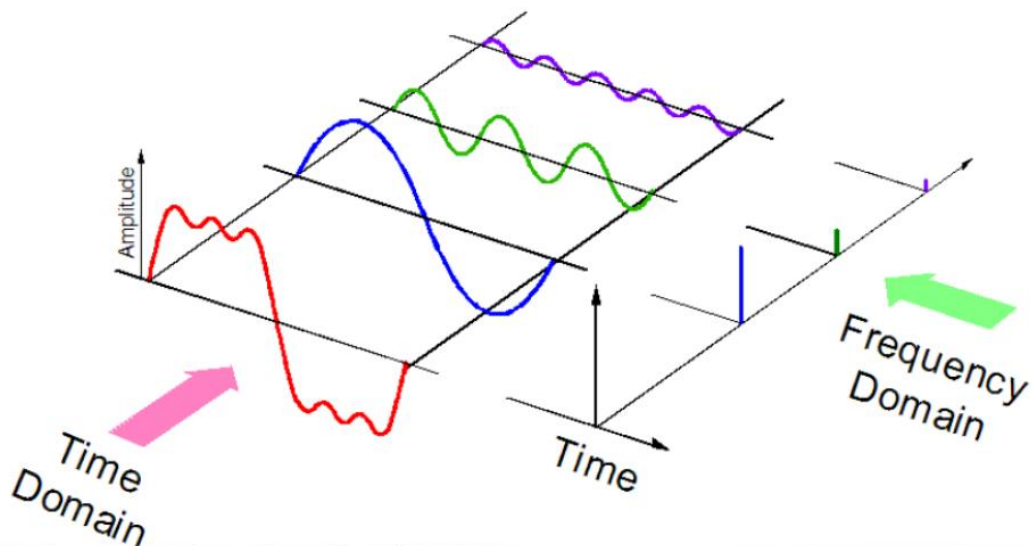
As with all unstructured data formats, audio data has a couple of preprocessing steps which have to be followed before it is presented for analysis. Here we will get an intuition on why this is done.

The first step is to actually load the data into a machine understandable format. For this, we simply take values after every specific time steps. For example; in a 2 second audio file, we extract values at half a second. This is called **sampling of audio data**, and the rate at which it is sampled is called the **sampling rate**.

Another way of representing audio data is by converting it into a different domain of data representation, namely the **frequency domain**. When we sample an audio data, we require much more data points to represent the whole data and also, the sampling rate should be as high as possible.

On the other hand, if we represent audio data in frequency domain, much less computational space is required. To get an intuition, take a look at the image below:

## The central idea

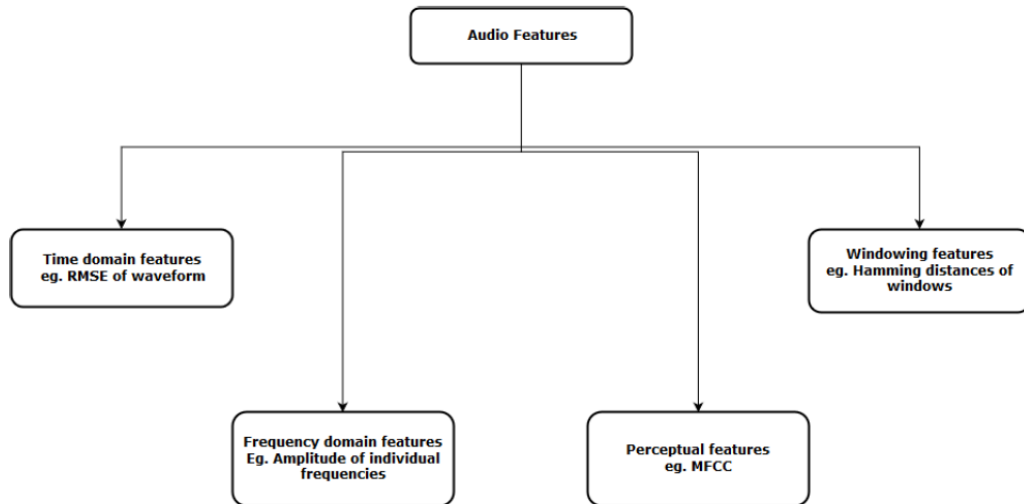


Here, we separate one audio signal into 3 different pure signals, which can now be represented as three unique values in frequency domain.

There are a few more ways in which audio data can be represented, for example using MFCs (Mel Frequency Cepstrum) [\[3\]](#).

### 2.5 Feature Extraction Techniques

The next step is to extract features from these audio representations, so that our algorithm can work on these features and perform the task it is designed for. Here's a visual representation of the categories of audio features that can be extracted.



Fg-6

After extracting these features, it is then sent to the machine learning model for further analysis <sup>[3]</sup>. Audio feature extraction underpins a massive proportion of audio processing, music information retrieval, audio effect design and audio synthesis. Design, analysis, synthesis and evaluation often rely on audio features, but there is a large and diverse range of feature extraction tools presented to the community. An evaluation of existing audio feature extraction libraries was undertaken <sup>[4]</sup>. Almost 7 + libraries were tried to perform the feature extraction task for retrieval of MFCC and delta MFCC. A summary of official documentation about those libraries with their features and application areas is illustrated below.

## 2.6 Libraries

### ▪ pyAudioAnalysis

PyAudioAnalysis is an open Python library that provides a wide range of audio-related functionalities focusing on feature extraction, classification, segmentation and visualization issues.

It lays 34 small and medium features and several other large features can also be extracted using those 34 features. *Dependencies:* Below you can find a list of library dependencies.

a.) NUMPY

b.) MATPLOTLIB



- c.) SCIPY
- d.) SKLEARN
- e.) Hmmlearn
- f.) Simplejson
- g.) eyeD3
- h.) pydub

Due to very large number of dependencies, it is quite tough to implement pyAudioAnalysis.

#### ▪ **YAFFE**

Yaafe is an audio features extraction toolbox. The Yaafe acronym, it's just *Yet Another Audio Feature Extractor* <sup>[5]</sup>.

##### *Easy to use:*

The user can easily declare the features to extract and their parameters in a text file. Features can be extracted in a batch mode, writing CSV or H5 files. The user can also extract features with Python or Matlab.

##### *Efficient:*

Yaafe automatically identifies common intermediate representations (spectrum, envelope, autocorrelation.) and computes them only once. Extraction is processed block per block so that arbitrarily long files can be processed, and memory occupation is low.

#### ▪ **LibROSA**

LibROSA is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. It can extract both spectral as well as rhythmic features. Few examples are:

##### *Spectral features-*

chroma_cens([y, sr, C, hop_length, fmin, ...])	Computes the chroma variant “Chroma Energy Normalized” (CENS).
melspectrogram([y, sr, S, n_fft, ...])	Compute a mel-scaled spectrogram.
mfcc([y, sr, S, n_mfcc])	Mel-frequency cepstral coefficients

<code>rmse([y, S, frame_length, hop_length, ...])</code>	Compute root-mean-square (RMS) energy for each frame, either from the audio samples <code>y</code> or from a spectrogram <code>S</code> .
<code>spectral_centroid([y, sr, S, n_fft, ...])</code>	Compute the spectral centroid.

### ***Rhythm Features-***

<code>tempogram([y, sr, onset_envelope, ...])</code>	Compute the tempogram: local autocorrelation of the onset strength envelope.
--	--

LibROSA is computationally the worst efficient python library. That's why we decided to leave exploring LibROSA any further.

### **▪ Essentia**

Essentia is an open-source C++ library for audio analysis and audio-based music information retrieval. It contains an extensive collection of algorithms including audio input/output functionality, standard digital signal processing blocks, statistical characterization of data, and a large set of spectral, temporal, tonal and high-level music descriptors. Essentia is cross-platform and it is designed with a focus on optimization in terms of robustness, computational speed and low memory usage, which makes it effective for many industrial applications. The library is also wrapped in Python and includes a number of command-line tools and third-party extensions, which facilitate its use for fast prototyping and allow setting up research experiments very rapidly.

### **Specialties:**

- ➔ Extensive collection of reusable algorithms
- ➔ Cross-platform
- ➔ Fast prototyping
- ➔ Optimized for computational speed

But, We didn't go with Essentia because its libraries was written in C++, and we faced very difficulty in making the platform ready for its implementation; since cross language libraries were causing hectic. We could not even set up the environment for the Essentia; it was demanding some Visual Studio C++ build tool 14.0; which caused some error while installation repeatedly.

## ▪ **python\_speech\_features**

This library provides common speech features for ASR including MFCCs and filterbank energies. You will need numpy and scipy to run these files.

### Supported features:

- `python_speech_features.mfcc()` - Mel Frequency Cepstral Coefficients
- `python_speech_features.fbank()` - Filterbank Energies
- `python_speech_features.logfbank()` - Log Filterbank Energies
- `python_speech_features.ssc()` - Spectral Subband Centroids

We have used this particular library in our project. It fulfilled all of our requirements in the feature engineering without causing any trouble. We have even achieved a remarkable accuracy in identification task.

## ▪ **Scikits.Talkbox**

Talkbox is set of python modules for speech/signal processing. The goal of this toolbox is to be a sandbox for features which may end up in scipy at some point. The following features are planned before version 1.0 release:

- Spectrum estimation related functions: both parametric (lpc, high resolution methods like music and co), and non-parametric (Welch, periodogram)
- Fourier-like transforms (DCT, DST, MDCT, etc...)
- Basic signal processing tasks such as resampling
- Speech related functionalities: mfcc, Mel spectrum, etc.

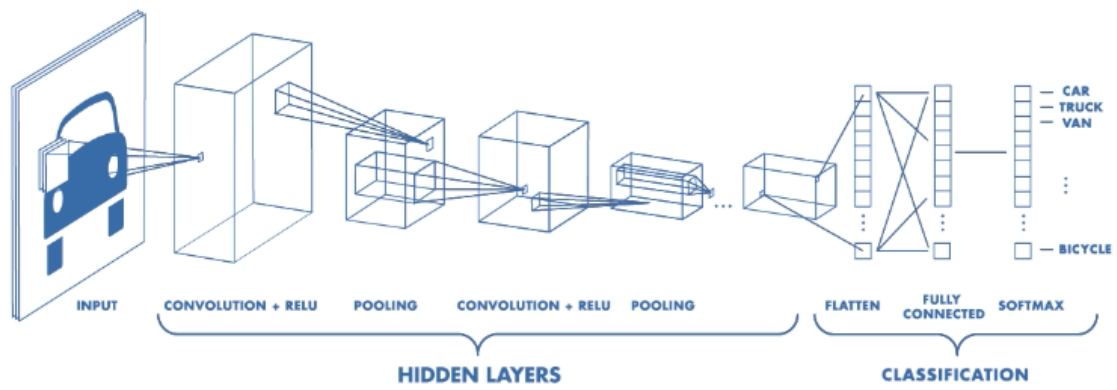
Since, this module is still under massive development. So, there are few bugs with its current build. Secondly, we didn't get proper documentation and any supporting tutorial. That's why we even planned to drop this library. It can be used as an alternate to LibROSA. They both can never be used together.

## Chapter: 3

# Proposed Methodology

---

After breaching through various approaches and different libraries, we finally stuck to using **Tensorflow** library. Our main methodology includes feature extraction through MFCC and then Model Training by CNN. The picture below illustrates our methodology pictorially.



As we can see, this can be summarized in **5 basic Phases**:

1. Data Acquisition
2. Data pre-processing
3. Feature Extraction
4. Model Training
5. Perform Testing (identification)

Here, we are going to explain each phase in little detail. Starting from the very first phase;

### ▪ Data Acquisition:

Though we tested our model's working and accuracy on a downloaded dataset; viz. VoxForge DATASET. But, to check its reliability and correctness; we also prepared a dataset by ourselves. It was voice recording of our friends, colleagues, and relatives.

So, we had two datasets in particular:

1. VoxForge Dataset
2. Manual self-made Dataset

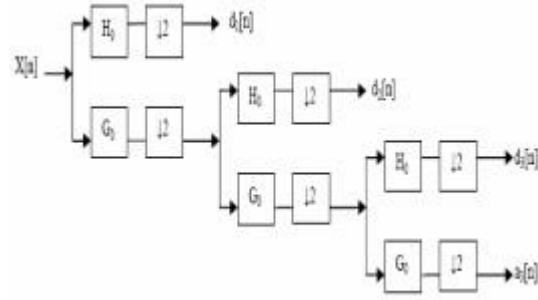
▪ **Data preprocessing:**

The data must be pre-processed in order to achieve better outputs and prediction results. This is to ensure that the model is trained with minimum errors.

- VoxForge dataset was already clean and noise free. So, we just skipped the pre-processing part in its case.

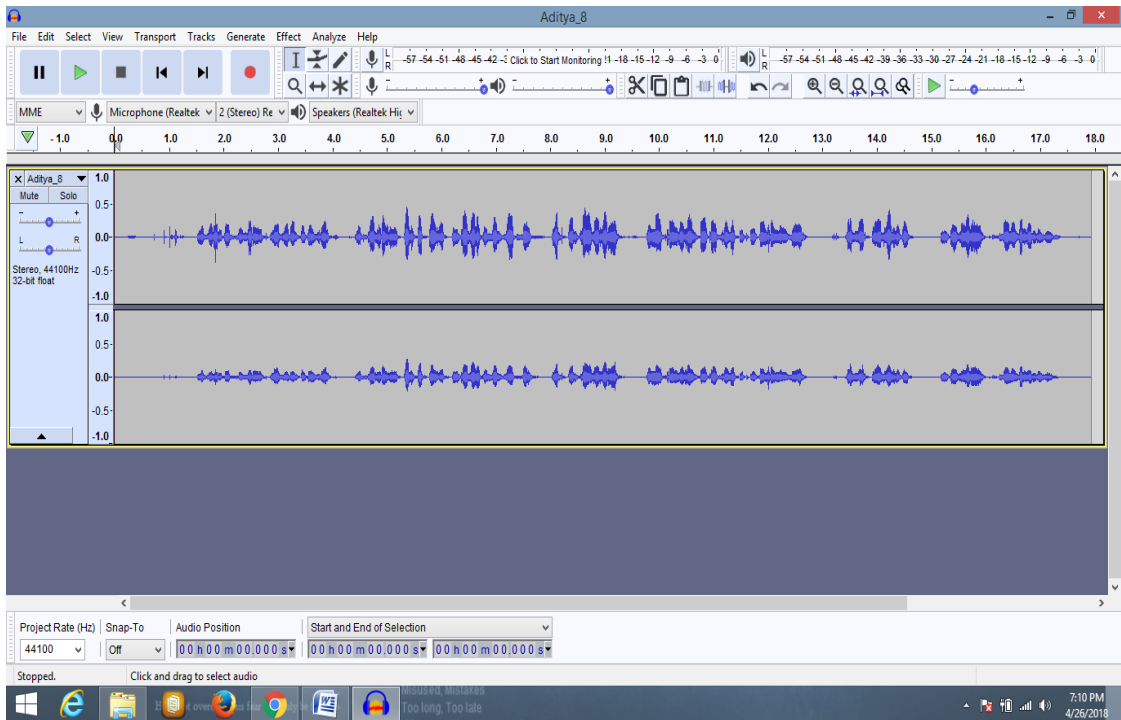
**1. Noise Reduction + Silence Removal:**

Noise reduction and Silence removal technique was applied on the audio samples. Noise leads to degrade the performance of speaker identification system .the de-noise process done by wavelet decomposition technique. The de-noising process consists of decomposing the original signal, thresholding the detail coefficients, and reconstructing the signal. The decomposition portion of de-noising is accomplished via the DWT. The Discrete Wavelet Transform (DWT) is commonly employed using dyadic multirate filter banks, which are sets of filters that divide a signal frequency band into sub bands. These filter banks are comprised of low pass, high-pass, or band pass filters. If the filter banks are wavelet filter banks that consist of special low-pass and high-pass wavelet filters, then the outputs of the low-pass filter are the approximation coefficients. Also, the outputs of the high-pass filter are the detail coefficients. The process of obtaining the approximation and detail coefficients is called decomposition. If a threshold operation is applied to the output of the DWT and wavelet coefficients that are below a specified value are removed, then the system will perform a “de-noising” function. There are two different threshold operations. In the first, hard thresholding, coefficients whose absolute values are lower than the threshold are set to zero. Hard thresholding is extended by the second technique, soft thresholding, by shrinking the remaining nonzero coefficients toward zero.



Fg-8

But, since pre-processing technique was not our main concern; we achieved this task completion by using a music/ audio amplification software; named **Audacity**. We performed noise removal and Silence removal manually one by one through a GUI interface of the software.



Fg-9

## 2. Conversion from .mp3 to .wav format:

We further converted all of the mp3 files into .wav file format. Since we are using *Scipy.iofile.wav* module for reading only .wav files in our project.

### ▪ Feature Extraction:

Here we are focusing on two main features **MFCCs** and their Derivatives, say **Delta-MFCC**. We calculated 20 MFCCs and 20 Delta-MFCCs. So, totally we had **40 features** in hand. Delta MFCC was calculated by a custom defined function under **featureextraction.py** module.

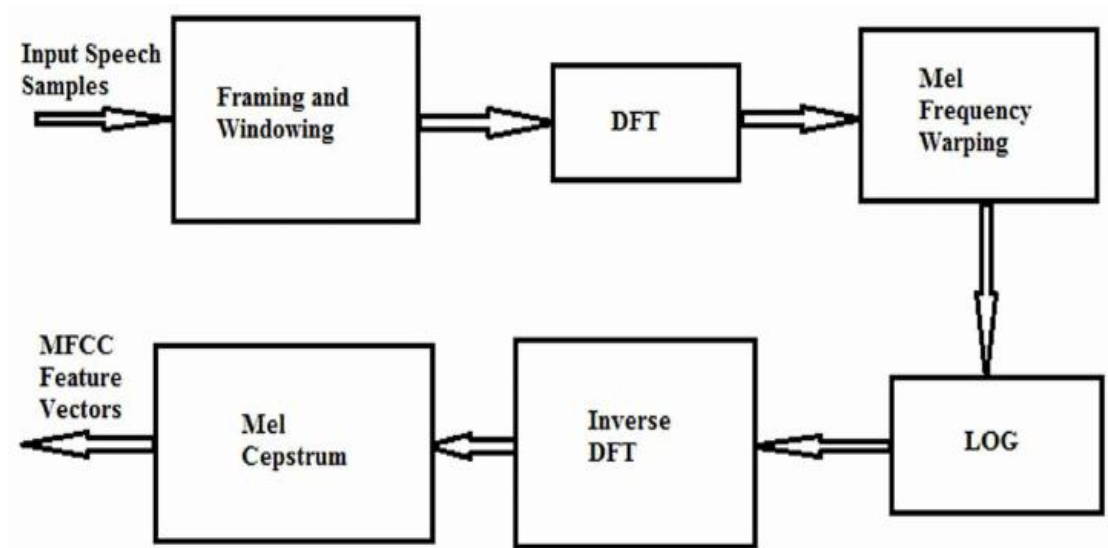
MFCC is one of the important feature extraction techniques for speaker identification technique. The goal of feature extraction is to find a set of properties of an utterance that have acoustic correlations to the speech signal which are parameters that can somehow be computed or estimated through processing of the signal waveform. Such parameters are termed as features.

### *Mel frequency Cepstral coefficient (MFCC) estimation*

After the process of removing background noises from voice signal has finish, the process of feature extraction will begin. Feature extraction is a process of obtaining different features of voice signal such as amplitude, pitch and the vocal tract. It is a task of finding parameter set obtained from the input voice signal. The extracted features should have some criteria in dealing with the speech signal such as: Stable over time

- Stable over time
- Should occur frequently and naturally in speech
- Should not be susceptible to mimicry
- Easy to measure extracted speech features
- Shows little fluctuation from one speaking environment to another
- Discriminate between speakers while being tolerant of intra speaker variability's

Mel Frequency Cepstrum Coefficients (MFCC) to extract features in the voice signal. MFCC focuses on series of calculation that uses Cepstrum with a nonlinear frequency axis following Mel scale. To obtain melcepstrum, the voice signal is windowed first using analysis window and then Discrete Fourier Transform is computed. The main purpose of MFCC is to mimic the behavior of human ears. MFCC estimation includes following process. MFCC process subdivided into six phases or blocks.



Fg-10

MFCC through python speech features:

`python_speech_features.base.mfcc`

`(signal, samplerate=16000, winlen=0.025, winstep=0.01, numcep=13, nfilt=26, nfft=512, lowfreq=0, highfreq=None, preemph=0.97, ceplifter=22, appendEnergy=True, winfunc=<function <lambda>>)`

Parameters:

- **signal** – the audio signal from which to compute features. Should be an N\*1 array
- **samplerate** – the samplerate of the signal we are working with.
- **winlen** – the length of the analysis window in seconds. Default is 0.025s (25 milliseconds)
- **winstep** – the step between successive windows in seconds. Default is 0.01s (10 milliseconds)
- **numcep** – the number of cepstrum to return, default 13
- **nfilt** – the number of filters in the filterbank, default 26.
- **nfft** – the FFT size. Default is 512.
- **lowfreq** – lowest band edge of mel filters. In Hz, default is 0.
- **highfreq** – highest band edge of mel filters. In Hz, default is samplerate/2
- **preemph** – apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97.
- **ceplifter** – apply a lifter to final cepstral coefficients. 0 is no lifter. Default is 22.



**Returns:**

- **appendEnergy** – if this is true, the zeroth cepstral coefficient is replaced with the log of the total frame energy.
- **winfunc** – analysis window to apply to each frame. By default no window is applied. Use numpy window functions here e.g. `winfunc=numpy.hamming`

A numpy array of size (NUMFRAMES by numcep) containing features. Each row holds 1 feature vector.

### ▪ **Model Training:**

We implemented the GMM approach for model training. The speaker identification system module can be separated into four modules:

- Front-end processing
- Speaker modeling
- Speaker database
- Decision logic

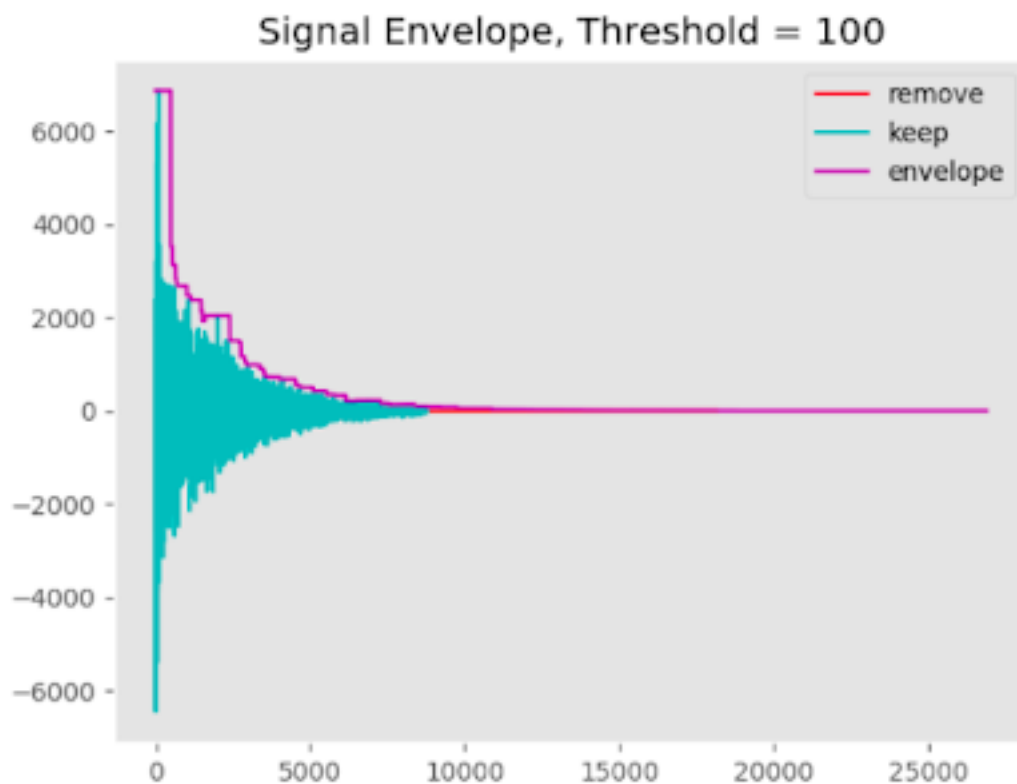
#### *Front- end processing*

- This step is the first step to create feature vectors. It is the “signal processing” part, which converts the sampled speech signal into set of feature vectors, which characterize the properties of speech that can separate different speakers. Front-end processing is performed both in training and testing phases. The objective in the front-end processing is to modify the speech signal, so that it will be more suitable for feature extraction analysis. The front-end processing operation based on noise cancelling, framing, windowing and pre-emphasis. The goal of feature extraction is to find a set of properties of an utterance that have acoustic correlations to the speech-signal, that is parameters that can somehow be computed or estimated through processing of the signal waveform. Such parameters are termed as features. It includes the process of measuring some important characteristic of the signal such as energy or frequency response, augmenting these measurements with some perceptually meaningful derived measurements and statically conditioning these numbers to form observation vectors.

### *Modeling*

- The objective of modeling technique is to generate models for each speaker using specific feature vector extracted from each speaker. It performs a reduction of feature data by modeling the distributions of the feature vectors. The speaker reorganization is also divided into two parts that means speaker dependant and speaker independent. In the speaker independent mode of the speech reorganization the computer should ignore the speaker specific characteristics of the speech signal and extract the intended message .on the other hand in case of speaker dependent mode speech reorganization machine should extract speaker characteristics in the acoustic signal.

### *Convolutional Neural Network (CNN)*

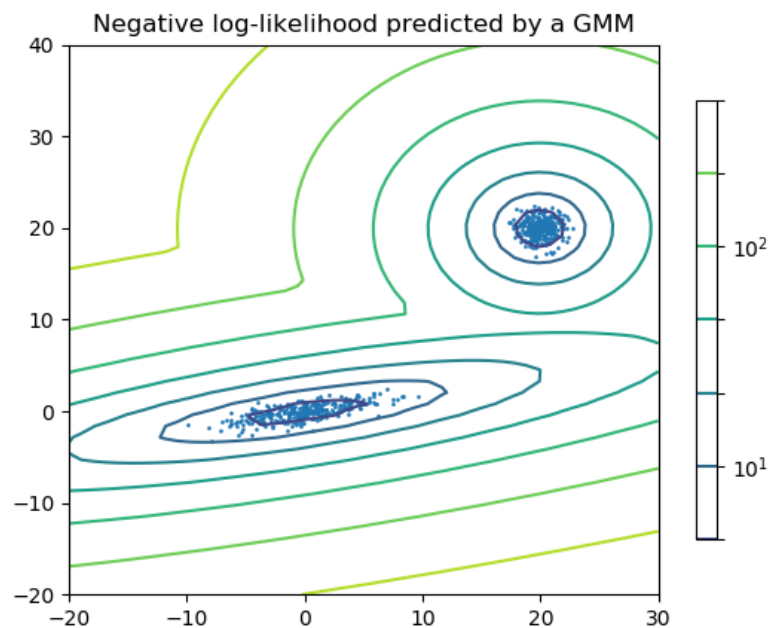


A signal envelope threshold graph in a Convolutional Neural Network visualizes the activation levels of neurons throughout the network. Imagine a series of peaks and valleys, where each peak represents a neuron responding strongly to an input feature. A threshold line separates the "activated" peaks from the "silent" ones. This helps understand which neurons are most responsive to specific features and how they contribute to the network's overall decision-making process. Analyzing this graph can

help identify potential biases, overfitting, and areas for improvement within the network architecture.

### ***Gaussian Mixture model (GMM)***

`sklearn.mixture` is a package which enables one to learn Gaussian Mixture Models (diagonal, spherical, tied and full covariance matrices supported), sample them, and estimate them from data. Facilities to help determine the appropriate number of components are also provided.



**Two-component Gaussian mixture model:** *data points, and equi-probability surfaces of the model.*

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

Scikit-learn implements different classes to estimate Gaussian mixture models that correspond to different estimation strategies.

### *Speaker database*

- The speaker models are stored here. These models are obtained for each speaker by using feature vector extracted from each speaker. These models are used for identification of unknown speaker during the testing phase.

### *Decision logic*

- It makes the final decision about the identity of the speaker by comparing unknown speaker to all models in the data base and selecting the best matching model.

### ▪ **Perform Identification:**

The log-likelihood for each .gmm model of every speaker was calculated in the model training phase. It was stored as a database in a separate folder. This data dictionary is used for matching 1: N speaker's gmm file. The speaker with the highest score is chosen and identified.

### 4.1 Data Dictionary

We gathered two different datasets:

#### 1. VoxForge:

- **340 cleaned and pre-processed** voice sample Data. Each voice sample was around **4 sec.** in length. So default value of **nfft = 512** in **mfcc()** just worked fine.

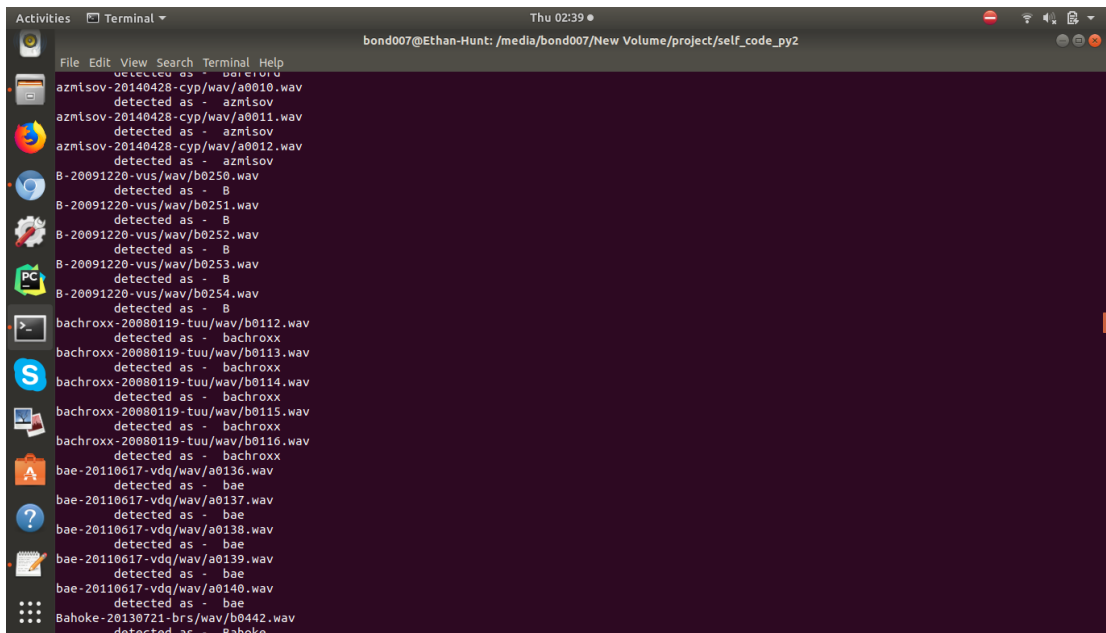
#### 2. Self-Made Dataset:

- **2 speakers** each accompanied 15 voice samples for training data. And 5 voice samples for testing data. So, total of **340 voice sample** Dataset. Each voice sample was around **30 sec. to 1 minute 40 sec** in length. So, we had to **tune parameters** in **mfcc()**. We adjusted **nfft** values from **512 to 2000 to 1800 to 1500**, finally.

### 4.3 Screenshots

```
Activities Terminal
bond007@Ethan-Hunt: /media/bond007/New Volume/project/self_code_py2
File Edit View Search Terminal Help
+ modeling completed for speaker: Apple_Eater.gmm with data point = (2877, 40)
Ara-20091020-vnt/wav/a0517.wav
Ara-20091020-vnt/wav/a0518.wav
Ara-20091020-vnt/wav/a0519.wav
Ara-20091020-vnt/wav/a0520.wav
Ara-20091020-vnt/wav/a0521.wav
+ modeling completed for speaker: Ara.gmm with data point = (3644, 40)
argall-20090722-gau/wav/rb-12.wav
argall-20090722-gau/wav/rb-13.wav
argall-20090722-gau/wav/rb-14.wav
argall-20090722-gau/wav/rb-15.wav
argall-20090722-gau/wav/rb-16.wav
+ modeling completed for speaker: argall.gmm with data point = (2468, 40)
ariyan-20120801-gra/wav/b0265.wav
ariyan-20120801-gra/wav/b0266.wav
ariyan-20120801-gra/wav/b0267.wav
ariyan-20120801-gra/wav/b0268.wav
ariyan-20120801-gra/wav/b0269.wav
+ modeling completed for speaker: ariyan.gmm with data point = (2586, 40)
arjuan-20100820-acz/wav/b0217.wav
arjuan-20100820-acz/wav/b0218.wav
arjuan-20100820-acz/wav/b0219.wav
arjuan-20100820-acz/wav/b0220.wav
arjuan-20100820-acz/wav/b0221.wav
+ modeling completed for speaker: arjuan.gmm with data point = (2787, 40)
Arten-20121123-ebd/wav/b0131.wav
Arten-20121123-ebd/wav/b0132.wav
Arten-20121123-ebd/wav/b0133.wav
Arten-20121123-ebd/wav/b0134.wav
Arten-20121123-ebd/wav/b0135.wav
+ modeling completed for speaker: Arten.gmm with data point = (2794, 40)
arthur-20130509-yjj/wav/ar-01.wav
arthur-20130509-yjj/wav/ar-02.wav
arthur-20130509-yjj/wav/ar-03.wav
arthur-20130509-yjj/wav/ar-04.wav
arthur-20130509-yjj/wav/ar-05.wav
+ modeling completed for speaker: arthur.gmm with data point = (3782, 40)
artk-20101225-kvd/wav/b0302.wav
```

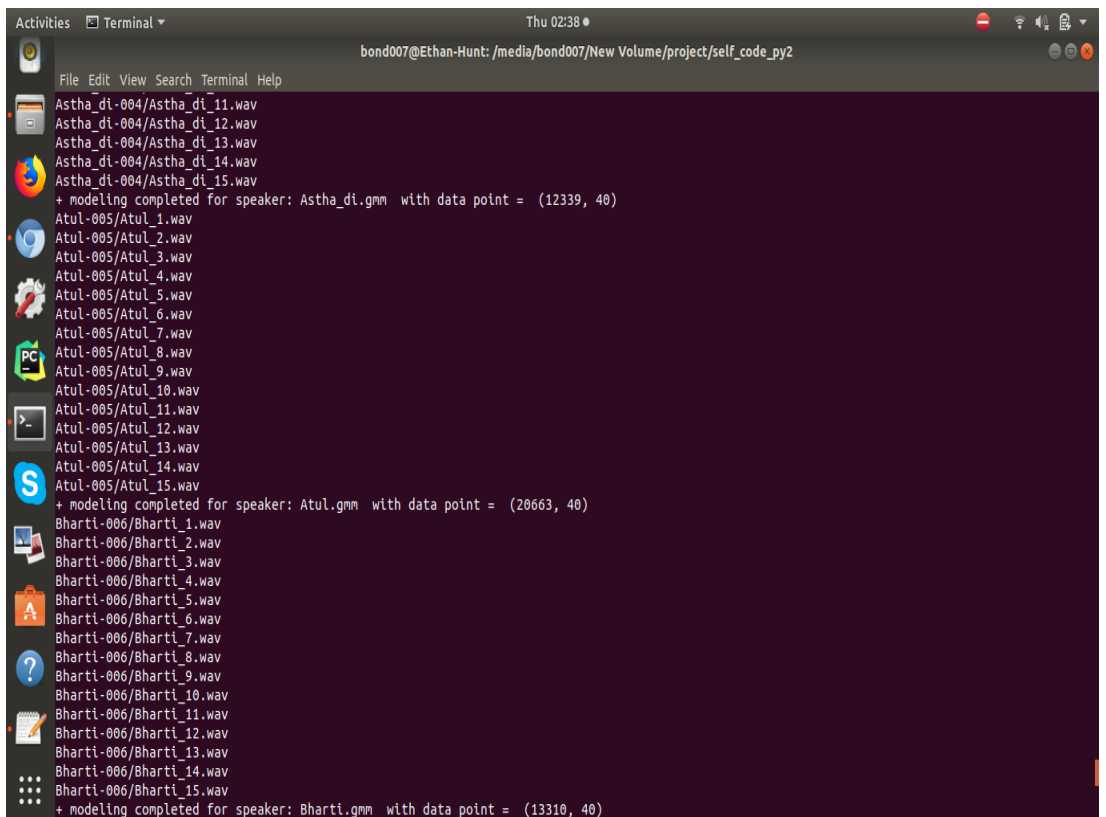
#### 1. VoxForge – Model Training



A terminal window titled 'bond007@Ethan-Hunt: /media/bond007/New Volume/project/self\_code\_py2' displays a list of audio files and their detected speaker names. The files are listed on the left, and the detected names are on the right. The detected names are: aznlsov, B, bachroxx, and bae. The files are: aznlsov-20140428-cyp/wav/a0010.wav, aznlsov-20140428-cyp/wav/a0011.wav, aznlsov-20140428-cyp/wav/a0012.wav, B-20091220-vus/wav/b0250.wav, B-20091220-vus/wav/b0251.wav, B-20091220-vus/wav/b0252.wav, B-20091220-vus/wav/b0253.wav, B-20091220-vus/wav/b0254.wav, bachroxx-20080119-tuu/wav/b0112.wav, bachroxx-20080119-tuu/wav/b0113.wav, bachroxx-20080119-tuu/wav/b0114.wav, bachroxx-20080119-tuu/wav/b0115.wav, bachroxx-20080119-tuu/wav/b0116.wav, bae-20110617-vdq/wav/a0136.wav, bae-20110617-vdq/wav/a0137.wav, bae-20110617-vdq/wav/a0138.wav, bae-20110617-vdq/wav/a0139.wav, bae-20110617-vdq/wav/a0140.wav, and Bahoke-20130721-brs/wav/b0442.wav.

```
File Edit View Search Terminal Help
aznlsov-20140428-cyp/wav/a0010.wav detected as - aznlsov
aznlsov-20140428-cyp/wav/a0011.wav detected as - aznlsov
aznlsov-20140428-cyp/wav/a0012.wav detected as - aznlsov
B-20091220-vus/wav/b0250.wav detected as - B
B-20091220-vus/wav/b0251.wav detected as - B
B-20091220-vus/wav/b0252.wav detected as - B
B-20091220-vus/wav/b0253.wav detected as - B
B-20091220-vus/wav/b0254.wav detected as - B
bachroxx-20080119-tuu/wav/b0112.wav detected as - bachroxx
bachroxx-20080119-tuu/wav/b0113.wav detected as - bachroxx
bachroxx-20080119-tuu/wav/b0114.wav detected as - bachroxx
bachroxx-20080119-tuu/wav/b0115.wav detected as - bachroxx
bachroxx-20080119-tuu/wav/b0116.wav detected as - bachroxx
bae-20110617-vdq/wav/a0136.wav detected as - bae
bae-20110617-vdq/wav/a0137.wav detected as - bae
bae-20110617-vdq/wav/a0138.wav detected as - bae
bae-20110617-vdq/wav/a0139.wav detected as - bae
bae-20110617-vdq/wav/a0140.wav detected as - bae
Bahoke-20130721-brs/wav/b0442.wav detected as - Bahoke
```

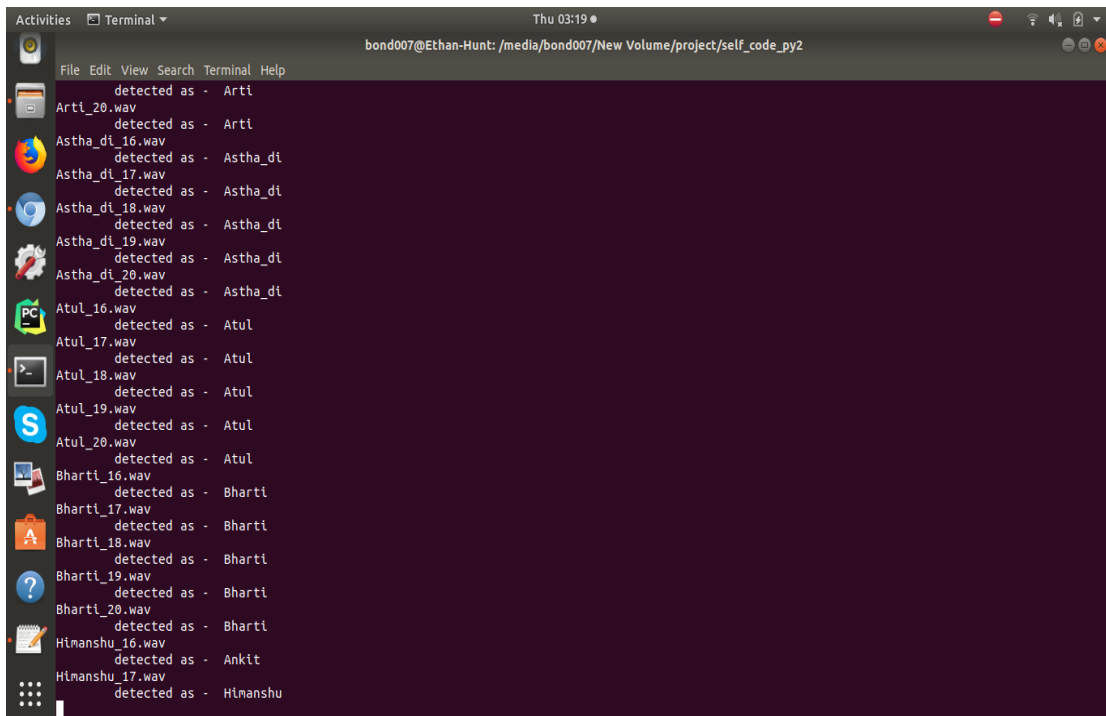
## 2. VoxForge – Testing : identification



A terminal window titled 'bond007@Ethan-Hunt: /media/bond007/New Volume/project/self\_code\_py2' displays a list of audio files and their identified speaker names. The files are listed on the left, and the identified names are on the right. The identified names are: Astha\_di, Atul, and Bharti. The files are: Astha\_di-004/Astha\_di\_11.wav, Astha\_di-004/Astha\_di\_12.wav, Astha\_di-004/Astha\_di\_13.wav, Astha\_di-004/Astha\_di\_14.wav, Astha\_di-004/Astha\_di\_15.wav, Atul-005/Atul\_1.wav, Atul-005/Atul\_2.wav, Atul-005/Atul\_3.wav, Atul-005/Atul\_4.wav, Atul-005/Atul\_5.wav, Atul-005/Atul\_6.wav, Atul-005/Atul\_7.wav, Atul-005/Atul\_8.wav, Atul-005/Atul\_9.wav, Atul-005/Atul\_10.wav, Atul-005/Atul\_11.wav, Atul-005/Atul\_12.wav, Atul-005/Atul\_13.wav, Atul-005/Atul\_14.wav, Atul-005/Atul\_15.wav, Bharti-006/Bharti\_1.wav, Bharti-006/Bharti\_2.wav, Bharti-006/Bharti\_3.wav, Bharti-006/Bharti\_4.wav, Bharti-006/Bharti\_5.wav, Bharti-006/Bharti\_6.wav, Bharti-006/Bharti\_7.wav, Bharti-006/Bharti\_8.wav, Bharti-006/Bharti\_9.wav, Bharti-006/Bharti\_10.wav, Bharti-006/Bharti\_11.wav, Bharti-006/Bharti\_12.wav, Bharti-006/Bharti\_13.wav, Bharti-006/Bharti\_14.wav, and Bharti-006/Bharti\_15.wav. The terminal also shows modeling completion for each speaker with data points: Astha\_di.gmm (12339, 40), Atul.gmm (20663, 40), and Bharti.gmm (13310, 40).

```
File Edit View Search Terminal Help
Astha_di-004/Astha_di_11.wav
Astha_di-004/Astha_di_12.wav
Astha_di-004/Astha_di_13.wav
Astha_di-004/Astha_di_14.wav
Astha_di-004/Astha_di_15.wav
+ modeling completed for speaker: Astha_di.gmm with data point = (12339, 40)
Atul-005/Atul_1.wav
Atul-005/Atul_2.wav
Atul-005/Atul_3.wav
Atul-005/Atul_4.wav
Atul-005/Atul_5.wav
Atul-005/Atul_6.wav
Atul-005/Atul_7.wav
Atul-005/Atul_8.wav
Atul-005/Atul_9.wav
Atul-005/Atul_10.wav
Atul-005/Atul_11.wav
Atul-005/Atul_12.wav
Atul-005/Atul_13.wav
Atul-005/Atul_14.wav
Atul-005/Atul_15.wav
+ modeling completed for speaker: Atul.gmm with data point = (20663, 40)
Bharti-006/Bharti_1.wav
Bharti-006/Bharti_2.wav
Bharti-006/Bharti_3.wav
Bharti-006/Bharti_4.wav
Bharti-006/Bharti_5.wav
Bharti-006/Bharti_6.wav
Bharti-006/Bharti_7.wav
Bharti-006/Bharti_8.wav
Bharti-006/Bharti_9.wav
Bharti-006/Bharti_10.wav
Bharti-006/Bharti_11.wav
Bharti-006/Bharti_12.wav
Bharti-006/Bharti_13.wav
Bharti-006/Bharti_14.wav
Bharti-006/Bharti_15.wav
+ modeling completed for speaker: Bharti.gmm with data point = (13310, 40)
```

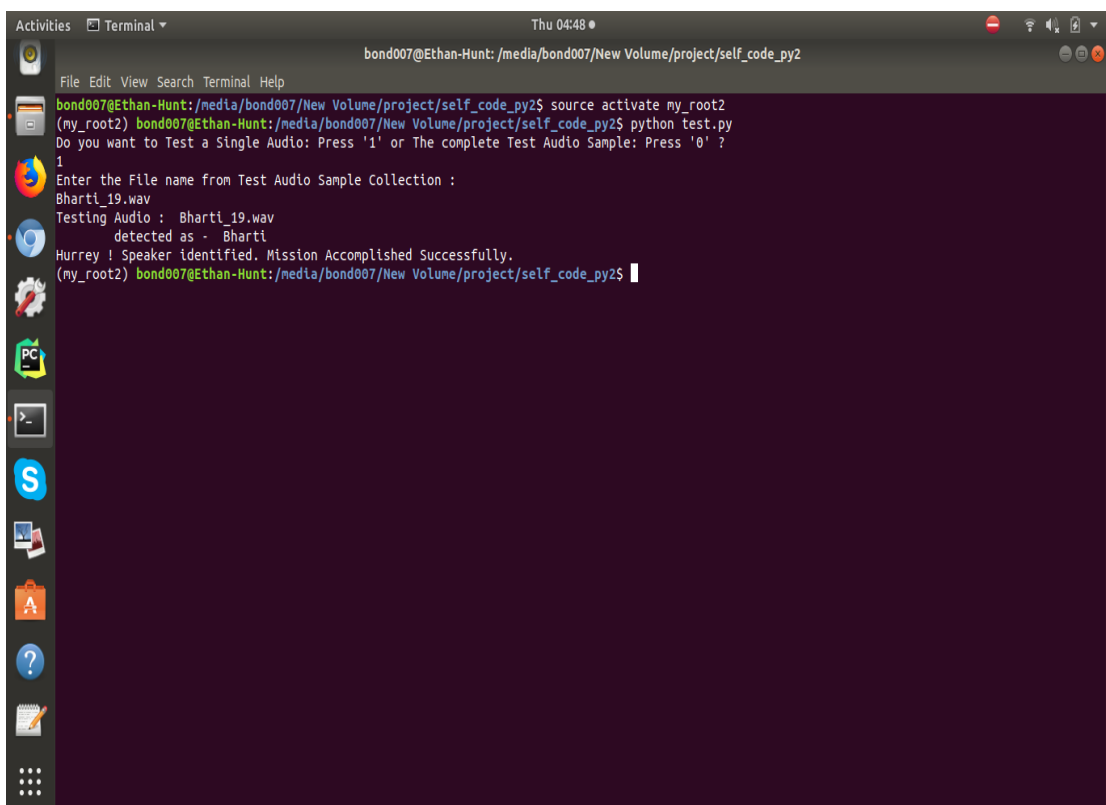
## 3. Self Made Dataset – Model Training



A terminal window titled 'bond007@Ethan-Hunt: /media/bond007/New Volume/project/self\_code\_py2' showing the output of a script. The script has detected 20 audio files and their corresponding speaker names. The output is as follows:

```
detected as - Arti
Arti_20.wav
detected as - Arti
Asthadi_16.wav
detected as - Asthadi
Asthadi_17.wav
detected as - Asthadi
Asthadi_18.wav
detected as - Asthadi
Asthadi_19.wav
detected as - Asthadi
Asthadi_20.wav
detected as - Asthadi
Atul_16.wav
detected as - Atul
Atul_17.wav
detected as - Atul
Atul_18.wav
detected as - Atul
Atul_19.wav
detected as - Atul
Atul_20.wav
detected as - Atul
Bharti_16.wav
detected as - Bharti
Bharti_17.wav
detected as - Bharti
Bharti_18.wav
detected as - Bharti
Bharti_19.wav
detected as - Bharti
Bharti_20.wav
detected as - Bharti
Himanshu_16.wav
detected as - Ankit
Himanshu_17.wav
detected as - Himanshu
```

#### 4. Self Made Dataset – testing: identification



A terminal window titled 'bond007@Ethan-Hunt: /media/bond007/New Volume/project/self\_code\_py2' showing the execution of a script for testing a single audio file. The user has entered 'Bharti\_19.wav' as the file name. The script has detected the speaker as 'Bharti' and displayed the message: 'Hurray ! Speaker identified. Mission Accomplished Successfully.'

```
bond007@Ethan-Hunt:/media/bond007/New Volume/project/self_code_py2$ source activate my_root2
(my_root2) bond007@Ethan-Hunt:/media/bond007/New Volume/project/self_code_py2$ python test.py
Do you want to Test a Single Audio: Press '1' or The complete Test Audio Sample: Press '0' ?
1
Enter the File name from Test Audio Sample Collection :
Bharti_19.wav
Testing Audio : Bharti_19.wav
detected as - Bharti
Hurray ! Speaker identified. Mission Accomplished Successfully.
(my_root2) bond007@Ethan-Hunt:/media/bond007/New Volume/project/self_code_py2$
```

#### 5. Single Audio file – Testing : identification

```

bond007@Ethan-Hunt: /media/bond007/New Volume/project/self_code_py2
File Edit View Search Terminal Help
bond007@Ethan-Hunt: /media/bond007/New Volume/project/self_code_py2$ source activate my_root2
(my_root2) bond007@Ethan-Hunt: /media/bond007/New Volume/project/self_code_py2$ python test.py
Do you want to Test a Single Audio: Press '1' or The complete Test Audio Sample: Press '0' ?
1
Enter the File name from Test Audio Sample Collection :
Bharti_19.wav
Testing Audio : Bharti_19.wav
detected as - Bharti
Hurray ! Speaker identified. Mission Accomplished Successfully.
(my_root2) bond007@Ethan-Hunt: /media/bond007/New Volume/project/self_code_py2$ python test.py
Do you want to Test a Single Audio: Press '1' or The complete Test Audio Sample: Press '0' ?
0
Testing Audio : Aditya_16.wav
detected as - Aditya
Testing Audio : Aditya_17.wav
detected as - Aditya
Testing Audio : Aditya_18.wav
detected as - Aditya
Testing Audio : Aditya_19.wav
detected as - Rajeev
Testing Audio : Aditya_20.wav
detected as - Aditya
Testing Audio : Ankit_16.wav
detected as - Ankit
Testing Audio : Ankit_17.wav
detected as - Ankit
Testing Audio : Ankit_18.wav
detected as - Ankit

```

**Fg-17**

## 6. Complete test data – testing: identification

```
[ ] Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 40, 64)	384
activation_1 (Activation)	(None, 40, 64)	0
dropout_1 (Dropout)	(None, 40, 64)	0
max_pooling1d_1 (MaxPooling1D)	(None, 10, 64)	0
conv1d_2 (Conv1D)	(None, 10, 128)	41088
activation_2 (Activation)	(None, 10, 128)	0
dropout_2 (Dropout)	(None, 10, 128)	0
max_pooling1d_2 (MaxPooling1D)	(None, 2, 128)	0
conv1d_3 (Conv1D)	(None, 2, 256)	164096
activation_3 (Activation)	(None, 2, 256)	0
dropout_3 (Dropout)	(None, 2, 256)	0
flatten_1 (Flatten)	(None, 512)	0
dense_1 (Dense)	(None, 8)	4104
activation_4 (Activation)	(None, 8)	0
Total params: 209,672		
Trainable params: 209,672		
Non-trainable params: 0		



## Chapter: 5

# Result

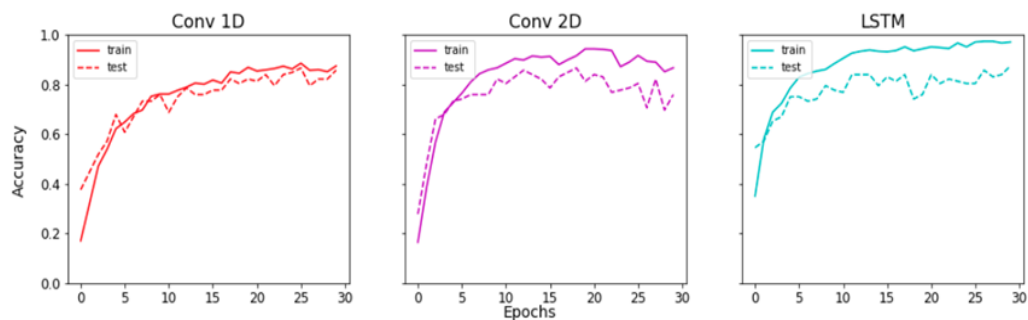
Thus, speaker identification was successfully conducted with an outstanding result on both of those datasets. The accuracy was **100%** in case of **VoxForge Dataset**; and **95.29 %** in case of **self made Dataset**. Thus, CNN- GMM model gives satisfactory results.

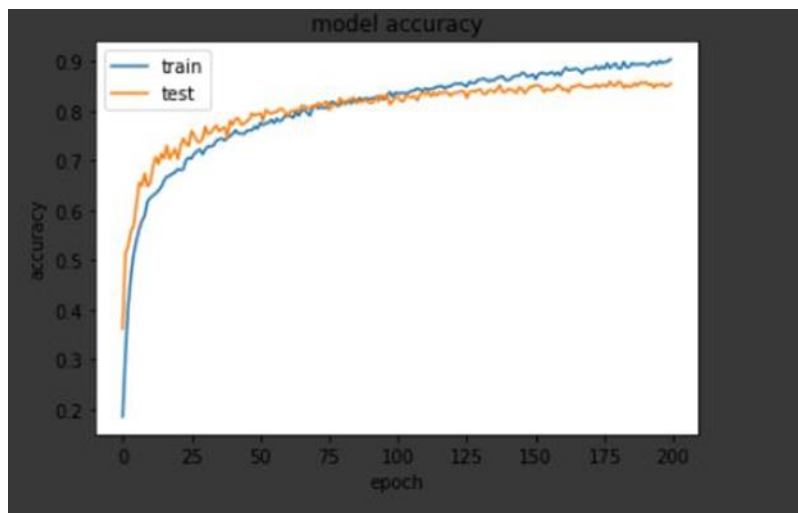
```
Activities Terminal Thu 04:50
bond007@Ethan-Hunt: /media/bond007/New Volume/project/self_code_py2

File Edit View Search Terminal Help

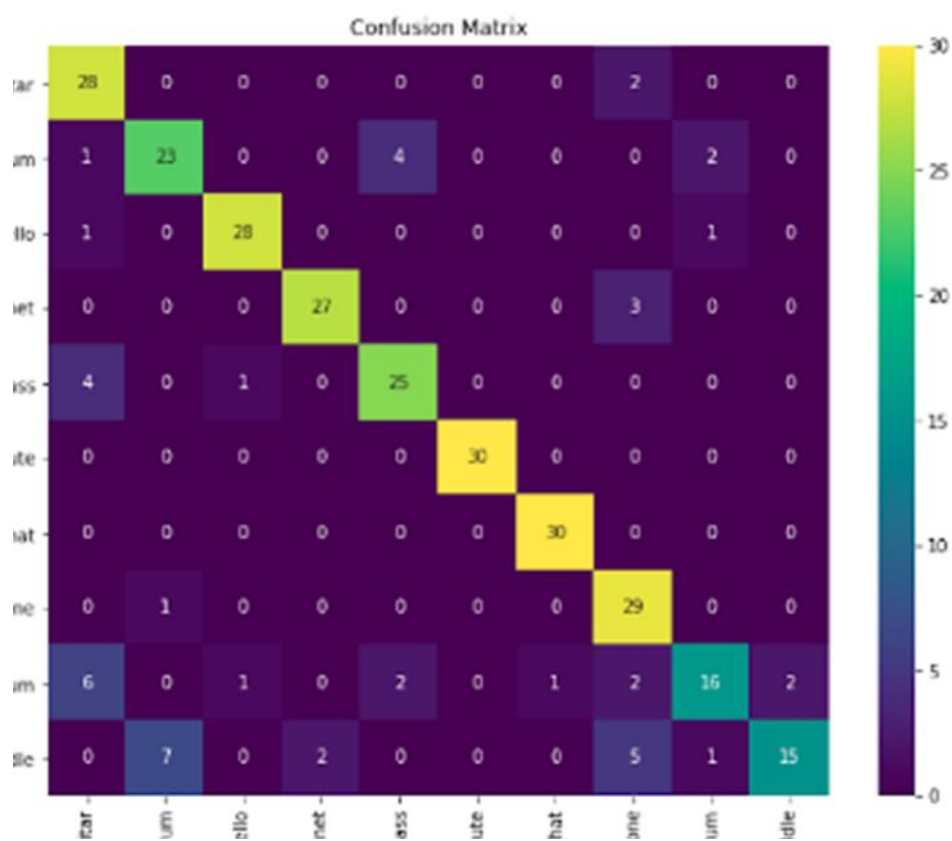
Testing Audio : Pooja_19.wav
detected as - Pooja
Testing Audio : Pooja_20.wav
detected as - Pooja
Testing Audio : Rajeev_16.wav
detected as - Rajeev
Testing Audio : Rajeev_17.wav
detected as - Aditya
Testing Audio : Rajeev_18.wav
detected as - Rajeev
Testing Audio : Rajeev_19.wav
detected as - Rajeev
Testing Audio : Rajeev_20.wav
detected as - Rajeev
Testing Audio : Raunak_16.wav
detected as - Raunak
Testing Audio : Raunak_17.wav
detected as - Raunak
Testing Audio : Raunak_18.wav
detected as - Raunak
Testing Audio : Raunak_19.wav
detected as - Raunak
Testing Audio : Raunak_20.wav
detected as - Raunak
Testing Audio : Shradha_16.wav
detected as - Shradha
Testing Audio : Shradha_17.wav
detected as - Shradha
Testing Audio : Shradha_18.wav
detected as - Shradha
Testing Audio : Shradha_19.wav
detected as - Shradha
Testing Audio : Shradha_20.wav
detected as - Shradha

4 85.0
The Accuracy Percentage for the current testing Performance with MFCC + GMM is : 95.2941176471 %
Hurray ! Speaker identified. Mission Accomplished Successfully.
(my_root2) bond007@Ethan-Hunt: /media/bond007/New Volume/project/self_code_py2$
```





**Model Accuracy Graph**



**Confusion Matrix**

## Chapter: 6

# Conclusion and Future Scope

---

### **Conclusion:**

The performances of Women Safety Application using various feature extraction and matching techniques. MFCC algorithm is used in our system because it has least false acceptance ratio. In order to improve system performance and also to achieve high accuracy CNN model can be used in feature matching technique. The speakers were trained and tested by using CNN and GMM model. They give better identification rate for speaker features. In future work, this technique integrated the pitch information with GMM and also to analyze the speaker identification system performance in the presence of noise.

Audio processing holds immense promise for creating a world where women feel safe, empowered, and free to live their lives without fear. By embracing the potential of this technology while ensuring its responsible development and implementation, we can build a safer future for all women. As technology continues to evolve, audio processing has the potential to become a powerful shield, protecting women and empowering them to live with confidence and courage.

### **Future Scope:**

Audio processing has already made significant strides in enhancing women's safety, but its potential extends far beyond current applications. As technology continues to evolve, we can expect even more innovative and effective solutions to emerge, shaping a future where women can live free from fear and harm.

Here are some exciting areas of exploration in the future of audio processing for women's safety:

1. Enhanced Sound Detection and Analysis:

- Real-time emotion recognition: By analyzing speech patterns and intonation with greater accuracy, systems can detect subtle emotions like fear or distress even when the individual is not explicitly calling for help. This can be crucial in identifying victims of domestic violence or other forms of hidden abuse.
- Context-aware threat identification: Systems will be able to analyze not only the sounds themselves but also the context in which they occur. This will allow them to distinguish between genuine threats and harmless noises, reducing false alarms and ensuring a more efficient response.
- Multi-modal analysis: Combining audio processing with other sensors, such as cameras and accelerometers, will provide a more comprehensive understanding of the situation. This will enable systems to identify dangers more accurately and take appropriate actions, even if the sounds are ambiguous or masked by background noise.

2. Personalized Safety Solutions:

- Biometric identification: By integrating voice recognition with other biometric data like facial recognition, systems can personalize safety measures to the individual. This can be particularly helpful for individuals with disabilities or specific needs.
- Adaptive learning: Systems will be able to learn and adapt to individual user preferences and behavior patterns. This will allow them to provide more personalized and effective protection, tailored to each woman's unique needs and risk profile.

- Psychological support: AI-powered chatbots and voice assistants will not only offer emotional support but also provide guidance and resources specific to the individual's situation and needs. This personalized support can be a powerful tool for coping with trauma and navigating the complexities of safety threats.

### 3. Expanding Accessibility and Inclusivity:

- Offline and low-resource solutions: Technologies will be developed to work effectively even in areas with limited internet connectivity or access to powerful devices. This will ensure that everyone, regardless of their location or resources, has access to safety solutions.
- Multilingual support: Systems will support a wider range of languages, making them accessible to women from diverse backgrounds and cultures. This will ensure that language barriers do not hinder access to protection.
- Culturally sensitive design: Technologies will be developed with sensitivity to different cultural contexts and beliefs. This will ensure that safety solutions are inclusive and respectful of diverse values and perspectives.

## Bibliography

1. Women Security Safety System using Artificial Intelligence Recognition, 2020.
2. N. Bhardwaj and N. Aggarwal, "Design and Development of “Suraksha”-A Women Safety", International Journal of Information & Computation Technology, vol. 4, no. 0974-2239, pp. 787-792, 2014.
3. Mitchell McLaren; Advances in deep neural network approaches to speaker recognition, **ICASSP** – April, 2015.
4. United Nations. Declaration on the elimination of violence against women. New York : UN, 1993.
5. WHO, LSHTM, SAMRC. Global and regional estimates of violence against women: prevalence and health impacts of intimate partner violence and non-partner sexual violence. WHO: Geneva, 2013.
6. Muskan, T. Khandelwal, M. Khandelwal and P. S. Pandey, "Women Safety Device Designed Using IoT and Machine Learning," 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Guangzhou, China, 2018, pp. 1204-1210, doi: 10.1109/SmartWorld.2018.00210.
7. Getting Started with Audio Data Analysis using Deep Learning (with case study) - **AnalyticsVidhya.com**
8. David Moffat, David Ronan, Joshua D. Reiss ;AN EVALUATION OF AUDIO FEATURE EXTRACTION TOOLBOXES; Nov 30 - Dec 3, 2015
9. Women Security Safety System using Artificial Intelligence Recognition, 2020.