SPEECH EMOTION RECOGNITION

A Capstone Project report submitted

in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

ANUGAM SAIKIRAN	**(19K41A0592)**
GADDAM HULDAH GRACE	**(19K41A0507)**
ALUGUVELLI ANIRUDH REDDY	**(19K41A0531)**
MUNDRATHI NISHANTH KIRAN	**(18K41A0532)**

Under the guidance of

Dr. T. SAMPATH KUMAR

Associate Professor, Department of CSE.

SR Engineering College

Innovation . Creativity . Entrepreneurship

SR ENGINEERING COLLEGE

Ananthasagar, Warangal.

CERTIFICATE

This is to certify that this project entitled **"SPEECH EMOTION RECOGNITION"** is the bonafied work carried out by **ANUGAM SAIKIRAN , GADDAM HULDAH GRACE, ALUGUVELLI ANIRUDH REDDY, MUNDRATHI NISHANTH KIRAN** as a Capstone Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING** during the academic year 2022-2023 under our guidance and Supervision.

DR. T. SAMPATH KUMAR
Associate Professor,
S R Engineering College,
Ananthasagar, Warangal.
Warangal

Dr. M. Sheshikala
Assoc. PHOD(CSE),
Engineering College,
Ananthasagar,

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our project guide **Mr. DR. T. SAMPATH KUMAR, Associate. Prof.** as well as Head of the CSE Department **Dr. M. Sheshikala, Associate Professor** for guiding us from the beginning through the end of the Capstone Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project co-ordinators **Mr. Sallauddin Md, Asst. Prof., and R. Ashok Asst. Prof.** for their encouragement and support.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Principal, **Dr. V. MAHESH,** for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

Speech emotion recognition is an act of recognizing human emotions and pitch. By using this system we can predict emotions such as sad, angry, surprised, calm, fearful, neutral, regret, and many more using some audio files. Recognition of emotion from speech signals is called speech emotion recognition. Extracting features from speech dataset we can train a machine learning model to recognize the emotion of the speech we can make speech emotion recognizer (SER).The data used for this is a large sample of people and their emotion controlled speeches. 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.

CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

ACRONYM	ABBREVATION
SER	SPEECH EMOTION RECOGNITION
MLPC	MULTILAYER PERCEPTRONCLASSFIER
SVM	SUPPORT VECTOR MACHINE
LSTM	LONG SHORT TERM MEMORY
MFCC	MEL FREQUENCY CEPSTRAL COEFFICIENTS

CHAPTER - 1

INTRODUCTION

1.1 OVERVIEW

The importance of emotion recognition of human speech has increased in recent days to improve both the naturalness and efficiency of human - machine interactions. Recognizing human emotions is a very complex task in itself because of the ambiguity in classifying the acted and natural emotions. Speech Emotion Recognition is a task of speech processing and computational paralinguistics that aims to recognize and categorize the emotions expressed in spoken language. The goal is to determine the emotional state of a speaker, such as happiness, anger, calm, or disgusting, from their speech patterns, such as prosody, pitch, and rhythm. Among the different methods, the voice-based intelligent devices are gaining popularity in a wide range of applications. In a voice-based system, a computer agent is required to completely comprehend the human's speech percept in order to accurately pick up the commands given to it. This field of study is termed as Speech Processing and consists of three components: Speaker Identification Speech Recognition Speech Emotion Detection Speech Emotion Detection is challenging to implement among the other components due to its complexity.

Furthermore, the definition of an intelligent computer system requires the system to mimic human behavior. A striking nature unique to humans is the ability to alter conversations based on the emotional state of the speaker and the listener. Speech emotion detection can be built as a classification problem solved using several machine learning algorithms. This project discusses in detail the various methods and experiments carried out as part of implementing a Speech Emotion Detection system.

The aim of the paper is to detect the emotions which are elicited by the speaker while speaking. Emotion Detection has become a essential task these days. The speech which is in fear, anger, joy have higher and wider range in pitch whereas have low range in pitch. Detection of speech is useful in assisting human machine interactions. Here we are using different classification algorithms to recognize the emotions , Multi-layer perception, and the audio feature MFCC, MEL, chroma were used. These models have been trained to recognize these emotions (Calm, neutral, surprise, happy,

sad, angry, fearful, disgust). we will use the libraries sound file to build a model using an MLP Classifier and LSTM. This will be able to recognize emotion from sound files. We will load the data, extract features from it, then split the dataset into training and testing sets. Then, we'll initialize an MLP Classifier , LSTM and train the model. Finally, we'll calculate the accuracy of our model.

1.2 EXISTING METHODS

The speech emotion detection system is implemented as a Machine Learning (ML) model. The steps of implementation are comparable to any other ML project, with additional fine-tuning procedures to make the model function better. The flowchart represents a pictorial overview of the process (see Figure 1). The first step is data collection, which is of prime importance. The model being developed will learn from the data provided to it and all the decisions and results that a developed model will produce is guided by the data. The second step, called feature engineering, is a collection of several machine learning tasks that are executed over the collected data. These procedures addressthe several data representation and data quality issues. The third step is often considered the core of an ML project where an algorithmic based train itself to respond to any new data it is exposed to. The final step is to evaluate the functioning of the built model. Very often, developers repeat the steps of developing a model and evaluating it to compare the performance of different algorithms. Comparison results help to choose the appropriate ML algorithm most relevant to the problem.

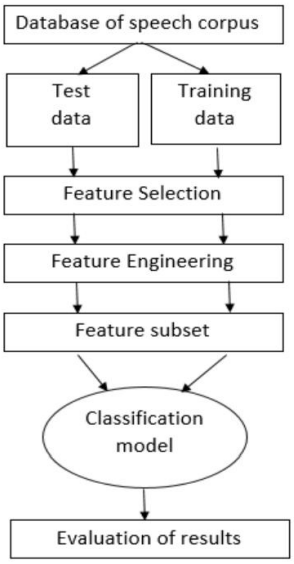

Fig 1.2.1 image depicting speech emotion recognition

1.3 PROPOSED METHODS

In our model we use libraries like MFCC, Chroma, mel spectrogram. The x-axis represents time, the y-axis represents frequency, and color represents amplitude. The brighter the color, the higher the amplitude. This spectrogram uses a special scale for frequency called the Mel scale. MFCC is one of the most common ways to perform feature extraction from audio waveforms. MFCC features are very efficient in preserving the overall shape of the audio waveform.Chroma is a specific kind of spectrogram based on the chromatic scale. The chromatic scale or 12 tone-scale is a set of 12 pitches used in tonal music. From Sklearn Neural Networks we use Multi-layer Perception Classifier and LSTM.

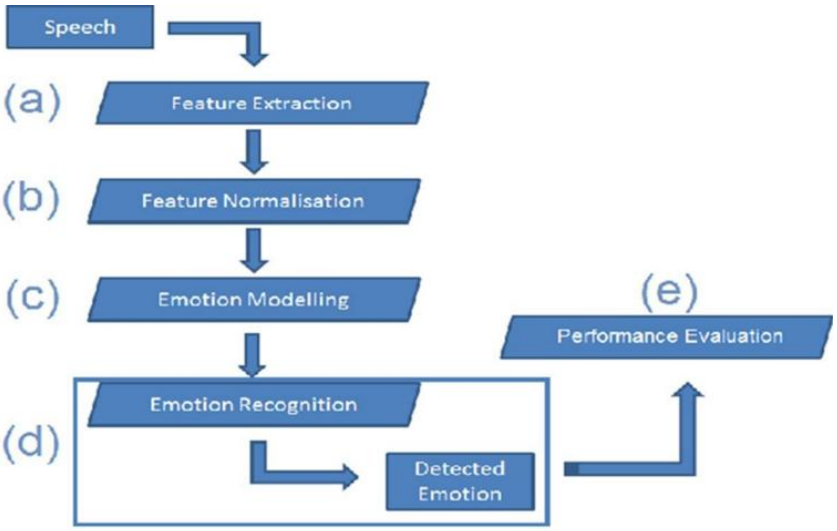

Fig 1.3.1 depicting MFCC algorithm

1.4 LITERATURE SURVEY

[1] Human emotion is a temporally dynamic event which can be inferred from both audio and video feature sequences. In this paper we investigate the long short term memory recurrent neural network (LSTM-RNN) based encoding method for category emotion recognition in the video. LSTM-RNN is able to incorporate knowledge about how emotion evolves over long range successive frames and emotion clues from isolated frame. After encoding, each video clip can be represented by a vector for each input feature sequence. The vectors contain both frame level and sequence level emotion information. These vectors are then concatenated and fed into support vector machine (SVM) to get the final prediction result.

[2] This paper critically analysed the literature on SER in terms of speech databases, speech features, traditional machine learning (ML) classifiers and DL approaches along with the areas for future directions. In recent years, there is a growing interest of researchers to use deep learning (DL) approaches for SER and get improvement in recognition rate.

[3] In this paper, we proposed a novel inference algorithm, a dialogical emotion decoding (DED) algorithm, that treats a dialog as a sequence and consecutively decode the emotion states of each utterance over time with a given recognition engine. This decoder is trained by incorporating intra- and inter-speakers emotion influences within a conversation. Our approach achieves a 70.1% in four class emotion on the IEMOCAP database, which is 3% over the state-of-art model. The evaluation is further conducted on a multi-party interaction database, the MELD, which shows a similar effect. Our proposed DED is in essence a conversational emotion rescoring decoder that can also be flexibly combined with different SER engines.

[4] In the proposed System Human emotion is a temporally dynamic event which can be inferred from both audio and video feature sequences. In this paper we investigate the long short term memory recurrent neural network (LSTM-RNN) based encoding method for category emotion recognition in the video. LSTM-RNN is able to incorporate knowledge about how emotion evolves over long range successive frames and emotion clues from isolated frame. After encoding, each video clip can be represented by a vector for each input feature sequence. The vectors contain both frame level and sequence level emotion information. These vectors are then

concatenated and fed into support vector machine (SVM) to get the final prediction result.

[5] This paper critically analysed the literature on SER in terms of speech databases, speech features, traditional machine learning (ML) classifiers and DL approaches along with the areas for future directions. In recent years, there is a growing interest of researchers to use deep learning (DL) approaches for SER and get improvement in recognition rate.

CHAPTER - 2

HARDWARE / SOFTWARE TOOLS

2.1 REQUIREMENT SPECIFICATION

Requirement identification is the first step of any software development project. Until the requirements of a client have been clearly identified, and verified, no other task (design, coding, testing) could begin. Usually business analysts having domain knowledge on the subject matter discuss with clients and decide what features are to be implemented. Categorization of Requirements Based on the target audience or subject matter, requirements can be classified into different types, as stated below:

User requirements:

They are written in natural language so that both customers can verify their requirements have been correctly identified

1. Consent to proceed for video access
2. Messaging.

System requirements:

They are written involving technical terms and/or specifications, and are meant for the development or testing teams.

21. Computer.
2. Processor.
3. Memory.
4. Python.

Functional Requirements (FRs):

These describe the functionality of a system - how a system should react to a particular set of inputs and what should be the corresponding output.

1. Access to the voice.

Non-functional requirements (NFRs):

They are not directly related to what functionalities are expected from the system. However, NFRs could typically define how the system should behave under certain situations. For example, an NFR could say that the system should work with 256MB RAM. Under such conditions, an NFR could be more critical than a FR. Non-functional requirement could be further classified into different types like:

• Product requirements:

1. Should be able to support Firefox 5 in Ubuntu.12

• Performance Requirements:

1. Works 24/7.
2. Supports huge network traffic.

Hardware Requirements:

Processor: Intel Core i5.

RAM: 256MB.

Software Requirements:

Operating System: Ubuntu.

Programming Languages: Python 3.

2.2.1 PYTHON

Python is a general purpose, dynamic, high-level, and interpreted programming language. It supports object oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures. Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. Python support multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles. Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc. We don't need to use data types to declare variable because it is dynamically typed. Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test debug cycle is very fast.

2.2.2 STREAMLIT

Streamlit is an open-source Python framework for building interactive web applications with minimal coding effort. It simplifies the process of creating and deploying data-driven apps by providing a clean and intuitive user interface. With Streamlit, users can build interactive dashboards, data visualizations, and machine learning models in just a few lines of code.

Streamlit is designed to be user-friendly, even for those with little or no experience in web development. Its simple syntax allows developers to create interactive web applications using Python, without the need to learn HTML, CSS, or JavaScript. It provides a range of pre-built widgets, such as sliders, buttons, and text input boxes, which can be easily integrated into the app.

One of the key features of Streamlit is its real-time app preview, which allows developers to see their changes instantly, without the need to restart the server or refresh the page. This makes the development process faster and more efficient.

Some of the popular uses of Streamlit include:

Data visualization: Streamlit provides a range of data visualization tools that allow users to create interactive charts, graphs, and maps. These visualizations can be easily customized to suit the needs of the user.

Machine learning models: Streamlit allows users to create and deploy machine learning models with ease. It provides a range of pre-built widgets for model selection, data input, and output visualization.

Dashboards: Streamlit can be used to create interactive dashboards that allow users to explore and analyze data in real-time. These dashboards can be easily shared with others, making it a great tool for collaboration.

Prototyping: Streamlit is an ideal tool for rapid prototyping of web applications. It allows developers to quickly build and test ideas, without the need for complex coding or server setup.

2.2.3 HTML

HTML (Hypertext Markup Language) HTML can be used for building web based applications that incorporate machine learning functionality. Here are a few ways that HTML might be used in the context of machine learning:

Building web-based user interfaces for machine learning models: HTML can be used to build the front-end of web-based user interfaces that allow users to interact with machine learning models. For example, an HTML form could be used to collect input data for a machine learning model, while JavaScript code could be used to send the data to the model for processing.

Visualizing machine learning results: HTML can be used to display the results of machine learning models in a visual and user-friendly way. For example, bar charts and graphs created using JavaScript libraries like D3.js can be embedded in an HTML page to display the results of a machine learning model.

Building web-based dashboards for machine learning applications: HTML can be used to build dashboards that allow users to monitor and interact with machine learning applications in real-time. For example, an HTML page could display the current status of a machine learning model, such as the current training loss or accuracy.

Overall, it can be a useful tool for building web-based applications that incorporate machine learning functionality.

Some common applications of HTML include:

- Web Development
- Email Newsletters
- E-commerce
- Responsive Design
- Online Advertising
- Web Applications

2.2.4 CSS

CSS can be used to style the user interfaces of web-based applications that incorporate machine learning functionality. Here are a few ways that CSS might be used in the context of machine learning Styling the user interface of web-based applications: CSS can be used to style the user interface of web-based applications that incorporate machine learning functionality. For example, CSS can be used to set the font size and color of text, to change the background color of the page, or to adjust the layout of user interface elements.

Creating custom visualizations: CSS can be used to create custom

visualizations of machine learning data. For example, CSS animations can be used to create dynamic and interactive visualizations of machine learning model outputs.
Applying machine learning to CSS: While CSS itself is not used for machine learning, there have been some interesting projects that apply machine learning techniques to CSS. For example, researchers have used machine learning to automatically generate CSS layouts based on user preferences.
Here are some of the ways in which CSS is helpful in web applications:

- Styling: CSS provides a way to style the content of web pages, allowing developers to control the layout, typography, and colors of web pages.
- Consistency: By using CSS, developers can apply consistent styles across multiple web pages or even an entire website, making it easier for users to navigate and understand the content.
- Responsiveness: CSS allows developers to create responsive web designs that can adapt to different screen sizes and devices, making web applications accessible on desktops, laptops, tablets, and smartphones.
- Accessibility: CSS can be used to improve the accessibility of web pages by providing visual cues and information that can be used by screen readers and other assistive technologies.
- Performance: By separating the content and presentation of a web page, CSS can improve the performance of web applications by reducing the amount of code that needs to be downloaded and processed by the browser.

Overall, it can be a useful tool for styling the user interfaces of web-based applications that incorporate machine learning functionality.

2.2.5 JAVASCRIPT

JavaScript is a powerful scripting language used primarily for web development, but it can also be used to create interactive web-based applications that incorporate machine learning functionality. Here are a few ways that JavaScript might be used in the context of machine learning.

Building interactive user interfaces for machine learning models: JavaScript can be used to create dynamic and interactive user interfaces for web-based applications that incorporate machine learning functionality. For example, JavaScript

can be used to create interactive visualizations of machine learning model outputs or to create custom user interface elements for controlling machine learning algorithms.

Preprocessing and manipulating data: JavaScript can be used to preprocess and manipulate data before feeding it into machine learning models. For example, JavaScript can be used to load and preprocess image data before feeding it into a computer vision model.

Developing machine learning libraries: JavaScript can be used to develop machine learning libraries and frameworks. For example, TensorFlow.js is a popular machine learning library that provides a JavaScript interface to TensorFlow, allowing 12 developers to build and train machine learning models directly in the browser.

Deploying machine learning models: JavaScript can be used to deploy machine learning models to the web. For example, machine learning models can be deployed as JavaScript libraries that can be embedded in web pages or used in web-based applications.

Overall, it can be a powerful tool for building interactive web-based applications that incorporate machine learning functionality, and for preprocessing and manipulating data before feeding it into machine learning models.

CHAPTER - 3

PROJECT IMPLEMENTATION

3.1 DESCRIPTION

Multi level perceptron classifier algorithm: The Multilayer Perceptron Classifier (MLPC) is a type of artificial neural network that is used for supervised learning tasks such as classification and regression. In MLPC, the data is passed through a series of layers that contain neurons or units, and each unit performs a simple mathematical operation on the input data. The output of each layer is passed to the next layer until the final output is produced. In this way, MLPC can learn complex relationships between input and output data.

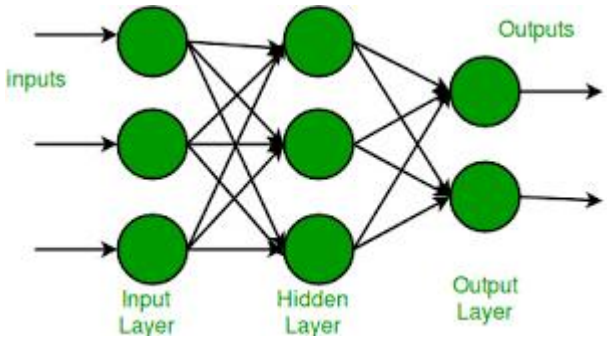

fig 3.1.1 diagram of simple MLPC architecture

In this diagram, the input layer consists of four input neurons, which represent the features of the input data. The hidden layer contains five neurons, and the output layer contains three neurons, which represent the possible classes or outputs. Each neuron in the hidden and output layers has a set of weights and biases that determine its output based on the inputs from the previous layer.

The MLPC algorithm uses a process called backpropagation to adjust the weights and biases of the neurons during training. Backpropagation works by calculating the error between the predicted output and the actual output, and then propagating that error back through the network to adjust the weights and biases of the neurons. This process is repeated many times during training until the network can accurately predict the output for new input data.

Here is a step-by-step overview of the MLPC algorithm:

- Initialize the weights and biases of the neurons in the network.
- Feed the input data through the network and calculate the output.
- Calculate the error between the predicted output and the actual output.
- Use backpropagation to adjust the weights and biases of the neurons in the network to reduce the error.
- Repeat steps 2-4 for many iterations or epochs until the network can accurately predict the output for new input data.
- Use the trained network to make predictions on new input data.
- Overall, MLPC is a powerful algorithm that can learn complex relationships between input and output data. It is widely used in machine learning applications such as image classification, speech recognition, and natural language processing.

Long Short Term Memory :

The Long Short-Term Memory (LSTM) algorithm is a type of recurrent neural network that is used for sequence modeling and prediction tasks, such as natural language processing, speech recognition, and stock price prediction. The LSTM algorithm is designed to overcome the vanishing gradient problem that can occur with standard recurrent neural networks, which makes it difficult to learn long-term dependencies in the input sequence.

The basic idea behind the LSTM algorithm is to use memory cells and gates to control the flow of information through the network. Each memory cell stores information over time, and the gates regulate the flow of information into and out of the cell. The gates are implemented as sigmoid activation functions, which allow the network to control the flow of information based on the input sequence.
Here are the main components of the LSTM algorithm:

- Input gate: This gate determines which information from the input should be stored in the memory cell. It takes the input sequence and the output of the previous time step as inputs and outputs a value between 0 and 1 for each element of the input sequence.

- Forget gate: This gate determines which information in the memory cell should be forgotten. It takes the input sequence and the output of the previous time step as inputs and outputs a value between 0 and 1 for each element of the memory cell.
- Output gate: This gate determines which information from the memory cell should be output as the final prediction. It takes the input sequence and the output of the previous time step as inputs and outputs a value between 0 and 1 for each element of the memory cell.
- Memory cell: This is where the LSTM stores the information over time. The cell has an internal state that is updated based on the input sequence, the output of the previous time step, and the input and forget gates.
- Cell state: This is the long-term memory of the LSTM, which is updated based on the input sequence and the forget and input gates.

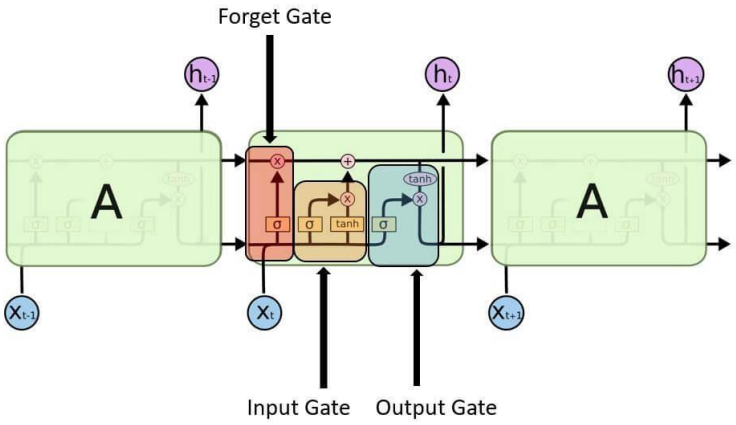

fig 3.1.2 Diagram of LSTM Architecture

In this diagram, the input sequence is fed into the LSTM network, and each element of the sequence is processed sequentially. The input gate determines which information from the input should be stored in the memory cell, the forget gate determines which information in the memory cell should be forgotten, and the output gate determines which information from the memory cell should be output as the final prediction. The memory cell stores the information over time, and the cell state is the long-term memory of the LSTM.

During training, the LSTM algorithm is optimized using backpropagation through time, which involves calculating the gradient of the loss function with respect to the LSTM parameters at each time step and updating the parameters in the direction of the negative gradient.

Support Vector Machine :

A Support Vector Machine (SVM) is a popular supervised learning algorithm used for classification, regression, and outlier detection analysis. It works by finding the optimal boundary, known as a hyperplane, that can separate different classes of data points.

The SVM algorithm is trained on labeled data to determine the hyperplane that maximizes the margin between the classes. The margin is the distance between the hyperplane and the closest data points from each class. The SVM finds the hyperplane that not only separates the data points but also maximizes this margin.

SVMs are particularly effective in dealing with high-dimensional feature spaces and non-linearly separable data. They can also use kernel functions to map the input data into higher-dimensional spaces, allowing for better separation of the data points.

Overall, SVMs are a powerful tool for solving a wide range of classification and regression problems and are widely used in fields such as image recognition, natural language processing, and bioinformatics.

here are some additional points about Support Vector Machines (SVMs):

SVMs are based on the idea of finding the hyperplane that maximally separates the data points of different classes in the feature space. The hyperplane is chosen so that the distance between the hyperplane and the closest data points of each class is maximized. This distance is known as the margin, and the points that lie on the margin are called support vectors.

SVMs can handle both linear and nonlinear decision boundaries by using different kernel functions that map the data to a higher-dimensional space where linear separation is possible. Popular kernel functions include polynomial kernels, Gaussian (RBF) kernels, and sigmoid kernels.

SVMs are sensitive to the choice of hyperparameters such as the regularization parameter (C) and the kernel parameters. Proper tuning of these parameters is essential for achieving good performance and avoiding overfitting.

SVMs have been successfully applied to a wide range of machine learning tasks such as classification, regression, and outlier detection.

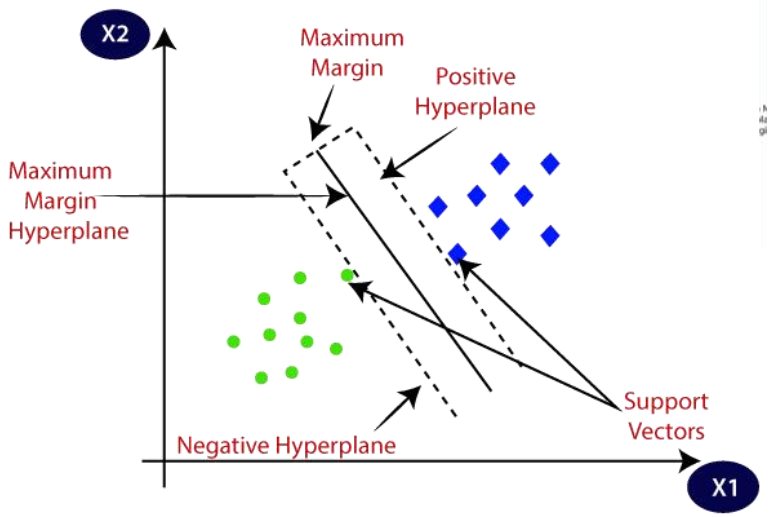

Fig 3.1.3 Diagram of Support vector machine

Random forest Algorithm:

Random Forest is a popular machine learning algorithm used for classification, regression, and other tasks. It works by constructing multiple Decision Trees on randomly sampled subsets of the data and combining their predictions.

The algorithm starts by selecting a subset of features and data points from the original dataset and building a Decision Tree on that subset. This process is repeated multiple times, with different subsets of features and data, resulting in a forest of Decision Trees.

When making predictions, each tree in the forest independently predicts the class or value of the target variable based on the input features. The final prediction is then made by aggregating the individual predictions.

Random Forest has several advantages over single Decision Trees, including reduced overfitting, improved accuracy, and the ability to handle high-dimensional data with many features. It also has the added benefit of feature importance measures, which can be used for feature selection and ranking.

Random Forest is widely used in fields such as finance, healthcare, and image processing for tasks such as fraud detection, disease diagnosis, and object recognition. Overall, Random Forest is a powerful and flexible algorithm that can be applied to a wide range of machine learning problems.

here are some additional points about the Random Forest algorithm:

- Random Forest can handle both categorical and continuous data, and it is robust to outliers and missing values.
- It is computationally efficient and can handle large datasets with many features.
- Random Forest can be used for both binary and multi-class classification problems, as well as for regression tasks.
- The algorithm can be tuned by adjusting hyperparameters such as the number of trees in the forest, the maximum depth of each tree, and the size of the subset of features used at each split.
- Random Forest is less interpretable than single Decision Trees because it generates a large number of trees, but feature importance measures can be used to understand the relative importance of each feature.
- The Random Forest algorithm is also used in other ensemble methods such as Gradient Boosting and Bagging.

MFCC:

MFCC (Mel Frequency Cepstral Coefficients) is a widely used feature extraction technique in audio signal processing, especially in speech recognition and speaker identification. It is a representation of the short-term power spectrum of a sound, based on the human auditory system's response to different frequencies. The following points detail the key aspects of MFCC:

- Mel-frequency: MFCC utilizes the mel-frequency scale, which is a perceptual scale of frequency based on the way humans hear sound. It is a non-linear scale

that maps frequencies to the mel scale, which more accurately reflects the way we perceive the relative differences between frequencies.

- Pre-emphasis: Before applying the MFCC algorithm, a pre-emphasis filter is applied to the audio signal, which boosts higher frequencies and reduces low frequencies to balance the spectrum.
- Frame blocking: The audio signal is divided into small frames of typically 20-30ms, which are processed independently. This ensures that the spectral characteristics of the sound can be tracked over time.
- Windowing: A windowing function (such as Hamming or Hanning) is applied to each frame to reduce spectral leakage caused by abrupt changes at the frame boundaries.
- Fourier Transform: A fast Fourier transform (FFT) is applied to each frame to obtain its frequency-domain representation.
- Mel-frequency filterbank: The Mel-frequency filterbank is used to filter the power spectrum obtained from the Fourier transform. The filterbank consists of a set of triangular filters, whose width and height are based on the mel scale. Each filter's output represents the energy in that frequency range.
- Logarithmic scaling: The log of the filterbank output is taken, which compresses the dynamic range of the signal and reflects the way we perceive loudness.
- Discrete Cosine Transform (DCT): The DCT is applied to the log-filterbank energies, producing a set of coefficients that are decorrelated and represent the spectral envelope of the sound.
- Cepstral coefficients: The first few DCT coefficients, typically between 12 and 20, are selected as the MFCCs, which capture the most relevant spectral features of the sound.
- Feature normalization: The MFCCs are often normalized to have zero mean and unit variance to reduce the impact of scaling on the feature values.
- Feature selection: In some cases, not all MFCC coefficients are equally important for classification. Feature selection techniques can be applied to identify the most relevant coefficients and reduce the dimensionality of the feature space.
- Applications: MFCC has numerous applications in speech and audio signal processing, such as speech recognition, speaker identification, speech synthesis, language identification, and emotion recognition.

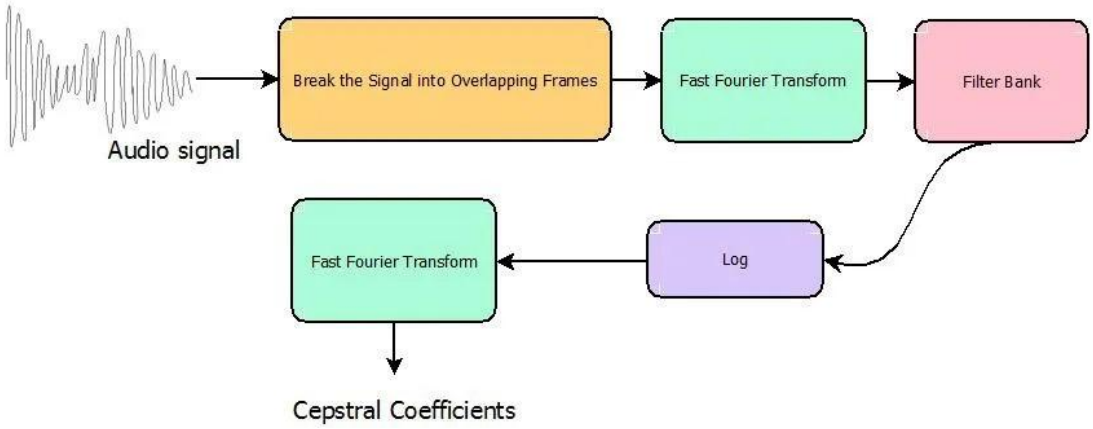

Fig 3.1.4 mfcc flowchart

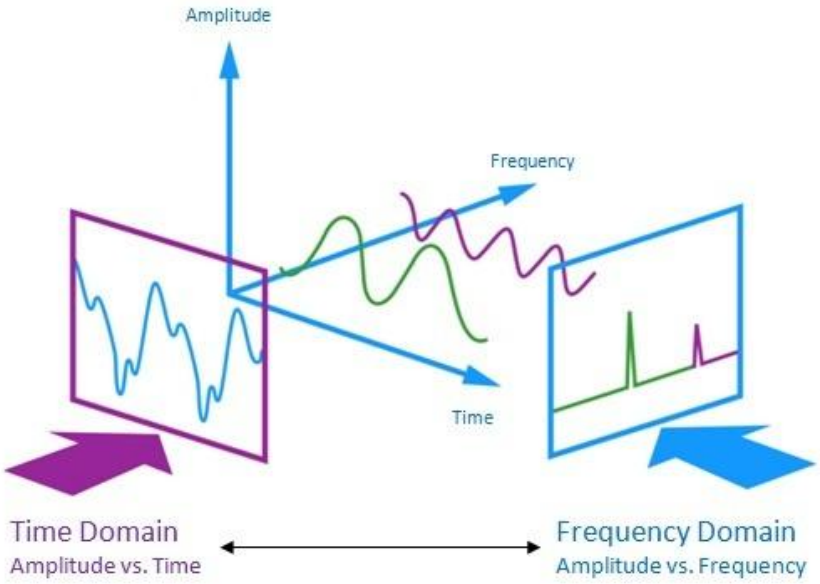

Fig 3.1.5 Time domain vs Frequency domain

- Time Domine : From the above figure we can observe that Time Domine nothing but Amplitude vs Time.

  Frequncy Domine : Frequncy Domine nothing but Amplitude vs Frequncy

- To extract features from raw audio we need to convert raw audio form Time Domine to Frequncy Domine.

CHROMA:

Chroma is a feature extraction technique in audio signal processing that represents the pitch content of a musical audio signal. It is widely used in music information retrieval tasks such as music genre classification, audio content-based retrieval, and chord recognition. Here are some key points about chroma:

- Pitch class: Chroma is based on the concept of pitch class, which is a musical term that refers to the set of notes that share the same pitch class, regardless of their octave. There are 12 pitch classes in the Western musical system.
- Pitch class profile: The chroma representation of an audio signal is called the pitch class profile (PCP). It is a 12-dimensional vector that represents the distribution of the 12 pitch classes in the audio signal.
- Short-time Fourier transform (STFT): The chroma representation is obtained by first computing the STFT of the audio signal. The STFT represents the time-varying frequency content of the audio signal.
- Harmonic filtering: The STFT is then filtered to emphasize the harmonic components of the audio signal. This is achieved by applying a bank of band-pass filters that correspond to the harmonics of each pitch class.
- Summation: The outputs of the filters for each pitch class are then summed to obtain the PCP. The resulting PCP is normalized to have unit magnitude.
- Octave equivalence: Chroma has the property of octave equivalence, which means that the PCP is invariant to the octave of the pitch classes. This property reflects the way humans perceive pitch, where the same pitch class sounds similar across different octaves.
- Transposition invariance: Chroma also has the property of transposition invariance, which means that the PCP is invariant to the key of the musical piece. This property allows chroma to capture the tonal content of the music without being affected by the specific key or tuning system used.
- Applications: Chroma is a powerful technique for music information retrieval tasks such as music genre classification, audio content-based retrieval, and chord recognition. It is also used in music analysis and synthesis, such as melody extraction, harmony analysis, and automatic transcription.

- Limitations: Chroma has some limitations, such as its sensitivity to the tuning and timbre of the musical instrument, and its inability to capture the rhythm or tempo of the music. Additionally, chroma is not suitable for non-musical audio signals or speech signals, where the pitch content is less important.

Overall, chroma is a valuable technique for representing the tonal content of musical audio signals. Its properties of octave equivalence and transposition invariance make it a robust and flexible tool for a wide range of music information retrieval tasks.

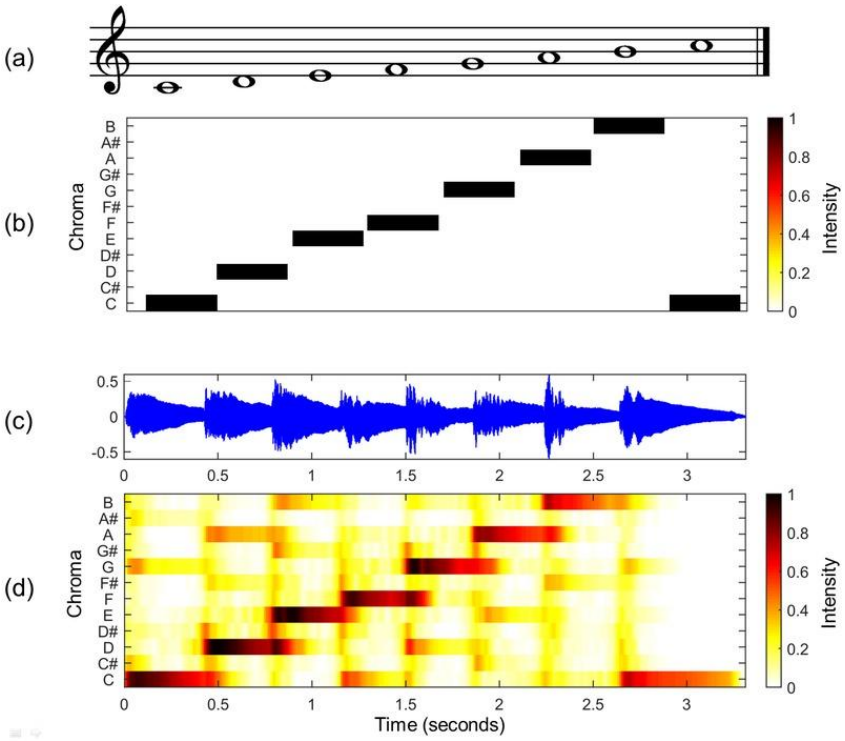

Fig 3.1.6 Depicting Chroma

MEL:

MEL (Mel-Frequency Cepstral Coefficients) is a feature extraction technique in audio signal processing that is similar to MFCC but uses a different frequency scale. Here are some key points about MEL:

- Frequency scale: MEL uses a non-linear frequency scale that is based on the mel scale, which is a perceptual scale of pitch that is based on the way humans perceive sound.
- Mel scale: The mel scale is defined such that a change of 1000 Hz in frequency is perceived as the same distance in pitch for all frequencies. The mel scale is commonly divided into 24 or 40 mel frequency bands.
- Filterbank: Like MFCC, MEL uses a filterbank to extract spectral features from the audio signal. The filterbank consists of a set of triangular filters that are evenly spaced on the mel scale.Energy calculation: The filterbank output is calculated by taking the energy of the audio signal within each filterbank band.
- Logarithmic compression: The filterbank output is then compressed using a logarithmic function to emphasize the lower frequencies and reduce the impact of high-frequency noise.
- Discrete cosine transform (DCT): The compressed filterbank output is then transformed using a DCT to obtain the MEL frequency cepstral coefficients (MFCCs).
- Delta and delta-delta coefficients: As with MFCC, the first and second derivatives of the MFCCs can also be calculated as delta and delta-delta coefficients.
- Feature normalization and selection: MEL also benefits from feature normalization and selection techniques, such as zero-mean normalization and feature selection algorithms, to improve the accuracy of classification tasks.
- Applications: MEL is used in a wide range of speech and audio processing applications, such as speech recognition, speaker identification, and emotion recognition.
- Limitations: Like MFCC, MEL assumes that the spectral envelope of the sound is stationary over time, which may not be true for all audio signals. Additionally,

the performance of MEL can be sensitive to the choice of parameters, such as the number and spacing of the mel frequency bands and the number of MFCCs.

Overall, MEL is a powerful technique for spectral feature extraction in audio signal processing, particularly for speech and spoken language applications. Its use of a non-linear frequency scale based on the mel scale makes it a valuable complement to MFCC in many speech processing tasks.

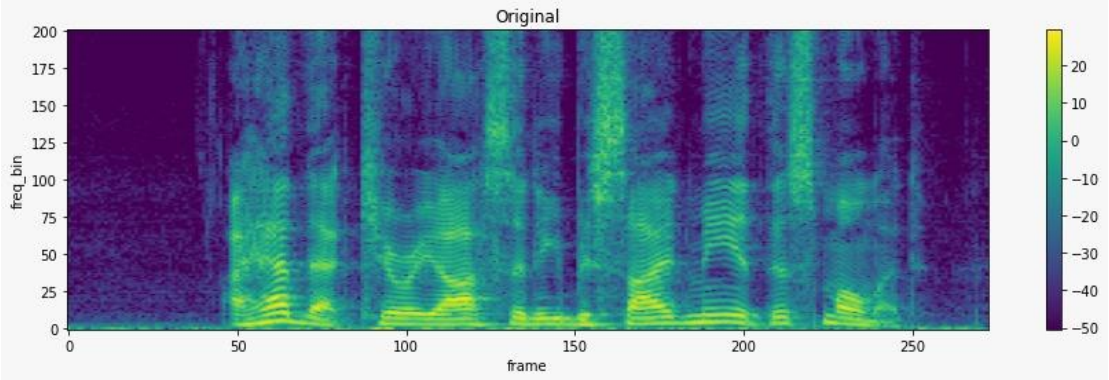

Fig 3.1.7 depicting about MEL

CNN:

A Convolutional Neural Network (CNN) is a type of neural network that is commonly used in computer vision tasks, such as image and video recognition. Here are some key points about CNNs:

- Convolutional layers: CNNs have convolutional layers, which apply a set of filters to the input image to extract features. Each filter is a small matrix that slides over the image, computing the dot product of its values with the corresponding input values at each position. The result is a feature map that highlights certain patterns in the input image.
- Pooling layers: After each convolutional layer, CNNs typically have a pooling layer, which reduces the size of the feature maps by taking the maximum or average value within a small window. This helps to make the network more robust to variations in the input image, such as translations and rotations.Non-linear activation functions: CNNs also have non-linear activation functions, such

as ReLU (Rectified Linear Unit), which introduce non-linearity into the network and allow it to learn more complex features.

- Fully connected layers: At the end of the network, CNNs typically have one or more fully connected layers, which combine the features extracted by the convolutional layers and produce a final output. These layers can also include dropout, which randomly removes some of the connections between neurons to prevent overfitting.
- Training: CNNs are typically trained using backpropagation and stochastic gradient descent, where the weights of the filters and fully connected layers are updated to minimize a loss function, such as cross-entropy or mean squared error.
- Transfer learning: CNNs can also be used for transfer learning, where a pre-trained network is used as a starting point for a new task. The pre-trained network is fine-tuned on the new task by adjusting the weights of the fully connected layers.
- Applications: CNNs have many applications in computer vision, such as image classification, object detection, and semantic segmentation. They are also used in other fields, such as natural language processing and audio processing.
- Limitations: CNNs have some limitations, such as their high computational cost, the need for large amounts of labeled data for training, and their limited ability to reason about the context and semantics of an image.
- Convolutional filters: The filters in the convolutional layers can be designed to capture different types of features, such as edges, corners, and textures. These filters are learned during training, and the network can adapt to different types of images and tasks.
- Spatial invariance: CNNs are designed to be spatially invariant, meaning that they can recognize the same pattern regardless of its location in the image. This is achieved through the use of pooling layers and shared weights in the convolutional layers.
- Data augmentation: To overcome the limitations of small labeled datasets, CNNs can be trained with data augmentation techniques, such as flipping, rotating, and cropping the input images. This creates additional training examples and helps to prevent overfitting.

- Architecture design: CNN architecture design is an active area of research, and many variations and improvements have been proposed over the years. Some popular architectures include AlexNet, VGG, ResNet, and Inception.
- Transfer learning applications: Transfer learning with CNNs has become a popular technique for many computer vision applications, especially when labeled training data is scarce. Pre-trained CNNs can be used as feature extractors, or the weights can be fine-tuned for a specific task.
- Interpretability: One challenge with CNNs is that they can be difficult to interpret, meaning it can be unclear why the network made a particular prediction. Researchers are exploring methods to visualize and explain the features learned by CNNs.
- Hardware optimization: As CNNs require a significant amount of computation, researchers and engineers are exploring hardware optimization techniques to speed up the inference and training processes. This includes the use of specialized chips, such as GPUs, TPUs, and FPGAs.

Overall, CNNs are a powerful tool for computer vision tasks, and their ability to automatically extract features from images makes them well-suited for a wide range of applications.

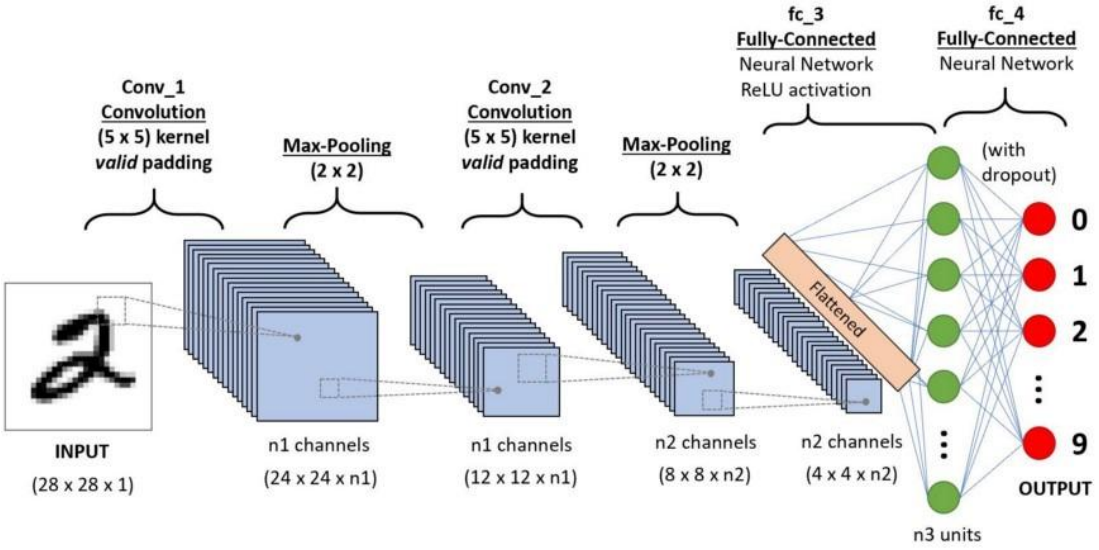

Fig 3.1.8 depicting CNN

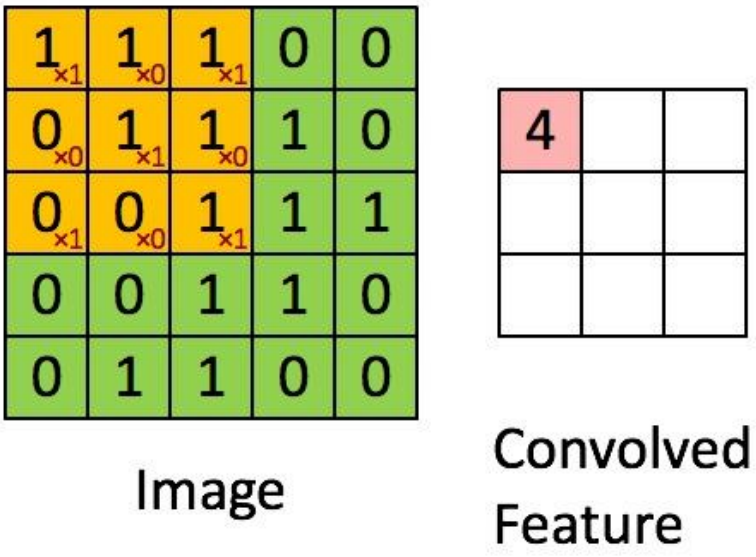

Fig 3.1.9 Depicting CNN convolved feature

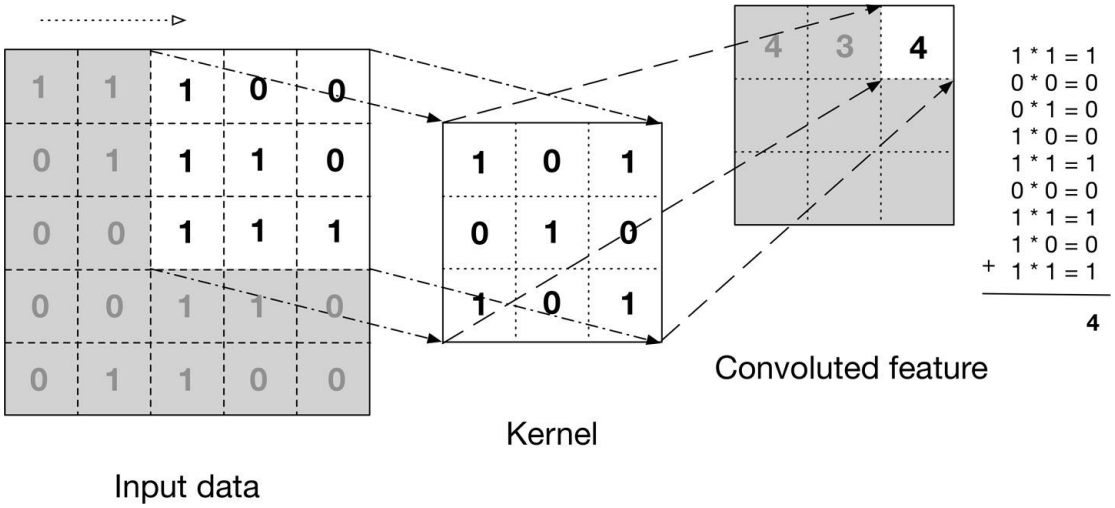

Fig 3.1.10 Depicting CNN

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

3.2 UML DIAGRAMS

3.2.1 SEQUENCE DIAGRAM

A sequence diagram is a type of UML (Unified Modeling Language) diagram that shows interactions between objects or components in a system over time. It illustrates the flow of messages or events between objects or components, and the order in which these interactions occur. The sequence diagram is particularly useful for visualizing complex scenarios involving multiple objects or components, as it allows developers to identify potential bottlenecks or inefficiencies in the system, and to design solutions to address these issues. Overall, the sequence diagram is a valuable tool for software engineers to plan, analyze, and communicate the behavior of a system.

- Audio Input: The first step in the sequence diagram is the audio input, which can be obtained through a microphone or an audio file.
- Feature Extraction: The next step is feature extraction, where acoustic features such as pitch, amplitude, and spectral density are extracted from the audio input. These features are then converted into a numerical format suitable for analysis by the machine learning algorithm.
- Pre-processing: The feature vector is then pre-processed to normalize or standardize the data and to remove any unwanted noise or artifacts that could affect the accuracy of the model.
- Classification: The pre-processed feature vector is then fed into a machine learning algorithm, which has been trained on a labeled dataset of speech samples with known emotional states. The algorithm uses this training data to classify the emotional state of the input speech sample as one of several possible emotions such as happy, sad, angry, or neutral.
- Output: The final step in the sequence diagram is the output, which represents the predicted emotion state of the input speech sample. The output can be displayed as a graphical representation or as a textual output indicating the predicted emotional state.

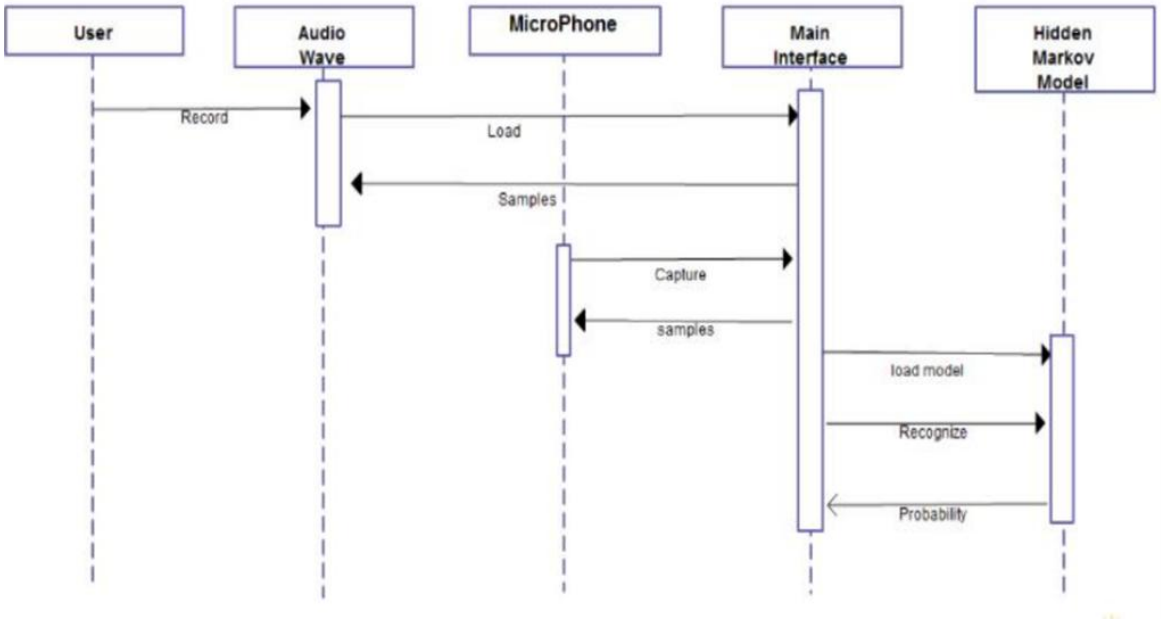

fig 3.2.1 Sequence diagram of speech emotion recognition

3.2.2 USE CASE DIAGRAM

A use case diagram is a type of UML (Unified Modeling Language) diagram that represents the functionality of a system in terms of actors, use cases, and their relationships.

Actors are external entities that interact with the system, while use cases represent specific tasks or functions that the system performs. Use cases are depicted as ovals, while actors are represented as stick figures or rectangles.

Use case diagrams are typically used in the early stages of software development to clarify the requirements of the system and to establish a common understanding among stakeholders. They help to identify the different types of users and the tasks they perform, as well as the interactions between the users and the system.

Overall, use case diagrams are a useful tool for understanding and documenting the requirements of a system, and for communicating those requirements to stakeholders. They can also help to identify potential issues or ambiguities in the requirements, which can be addressed before development begins

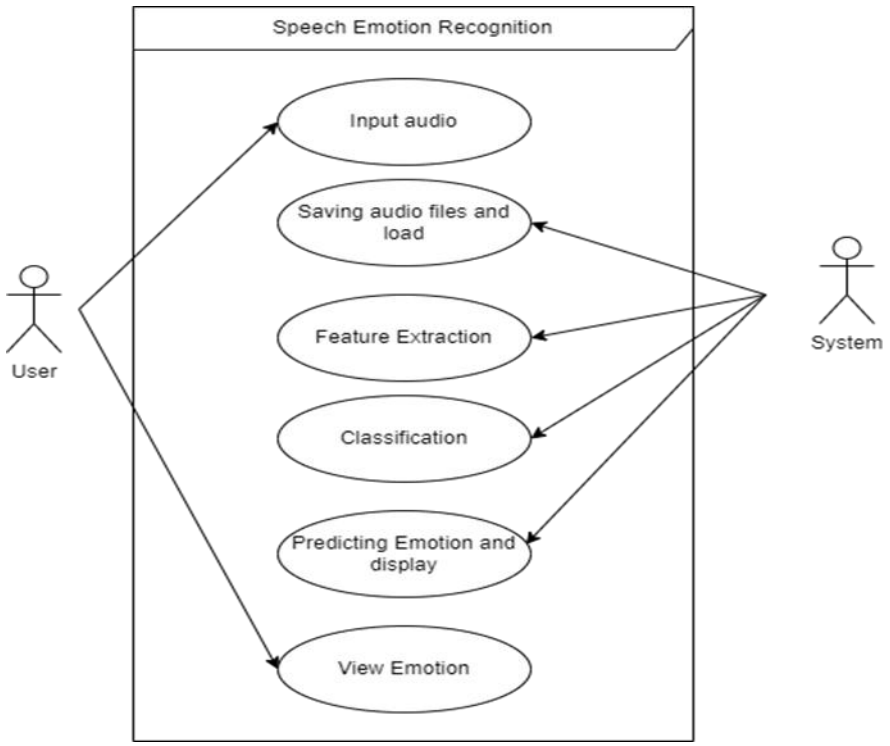

fig 3.2.2 Use case diagram of speech emotion recognition

The use case diagram provides a visual representation of the main functions of the Speech Emotion Recognition system, as well as the interactions between the system and its users. The diagram consists of five use cases: Record Speech, Preprocessing, Feature Extraction, Feature Selection, Emotion Classification, and Output.

The final use case in the diagram is "Output", which represents the predicted emotional state of the input speech sample. The output can be displayed as a graphical representation or as a textual output indicating the predicted emotional state.

Overall, the use case diagram for Speech Emotion Recognition provides a high-level view of the system's functionality and the interactions between the user and the system. It highlights the key use cases involved in the process of emotion recognition, from capturing the audio input to the final output of the predicted emotional state. By identifying the different use cases, stakeholders can better understand the requirements of the system and design a more effective solution.

3.2.3 FLOWCHART

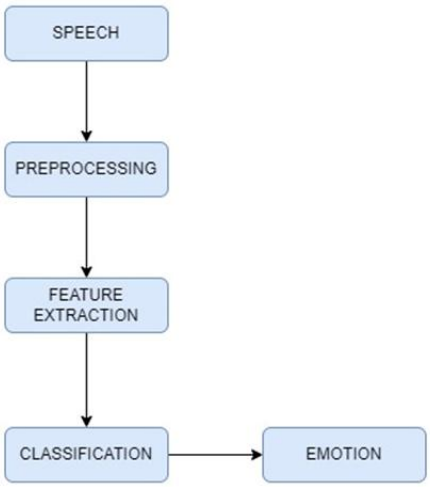

fig 3.2.3 FLowchart of speech emotion recognition

- Audio Input: The first step in the flowchart is the audio input, which can be obtained through a microphone or an audio file.
- Preprocessing: The audio input is then preprocessed to remove any unwanted noise or artifacts that could affect the accuracy of the model. This could include filtering, normalization, or resampling of the audio signal.
- Feature Extraction: The preprocessed audio signal is then processed to extract relevant acoustic features such as pitch, intensity, and spectral content. These features are then converted into a numerical format suitable for analysis by the machine learning algorithm.
- Feature Selection: The extracted features are then selected based on their relevance to the task of emotion recognition. This could involve techniques such as principal component analysis or feature ranking algorithms.
- Emotion Classification: The selected features are then fed into a machine learning algorithm trained on a labeled dataset of speech samples with known emotional states. The algorithm uses this training data to classify the emotional state of the input speech sample as one of several possible emotions such as happy, sad, angry, or neutral.
- Output: The final step in the flowchart is the output, which represents the predicted emotional state of the input speech sample. The output can be displayed

as a graphical representation or as a textual output indicating the predicted emotional state.

Overall, Speech Emotion Recognition is a complex process that involves several stages of data processing and machine learning. The flowchart provides a visual representation of the steps involved in the process, highlighting the flow of data from the audio input to the final output of the predicted emotional state. The flowchart also emphasizes the importance of feature extraction and selection in achieving accurate emotion recognition, as well as the critical role of machine learning algorithms in learning from labeled data to classify emotional states.

3.3 WORKING OF PROPOSED METHOD

3.3.1 DATASET

These are the audio files of emotions like calm, fearful, happy, disgust etc. and these audio files are used to train the algorithm.

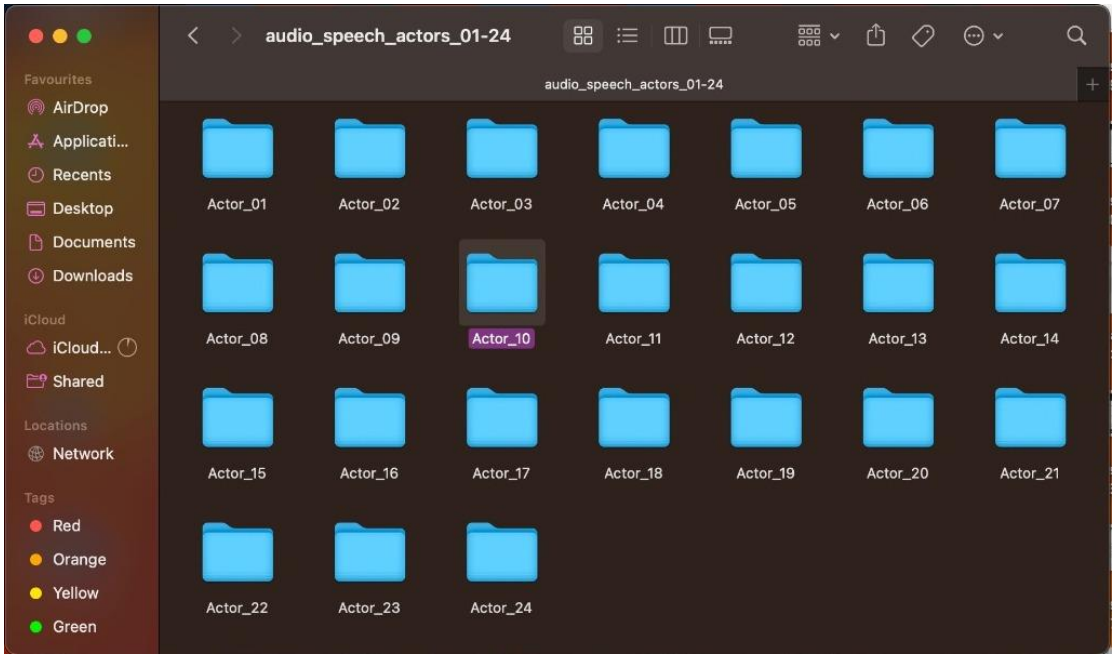

fig 3.3.1: Dataset

Description:

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) contains 7356 files (total size: 24.8 GB). The database contains 24 professional actors

(12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression. All conditions are available in three modality formats: Audio-only (16bit, 48kHz .wav), Audio-Video (720p H.264, AAC 48kHz, .mp4), and Video-only (no sound). Note, there are no song files for Actor_18.

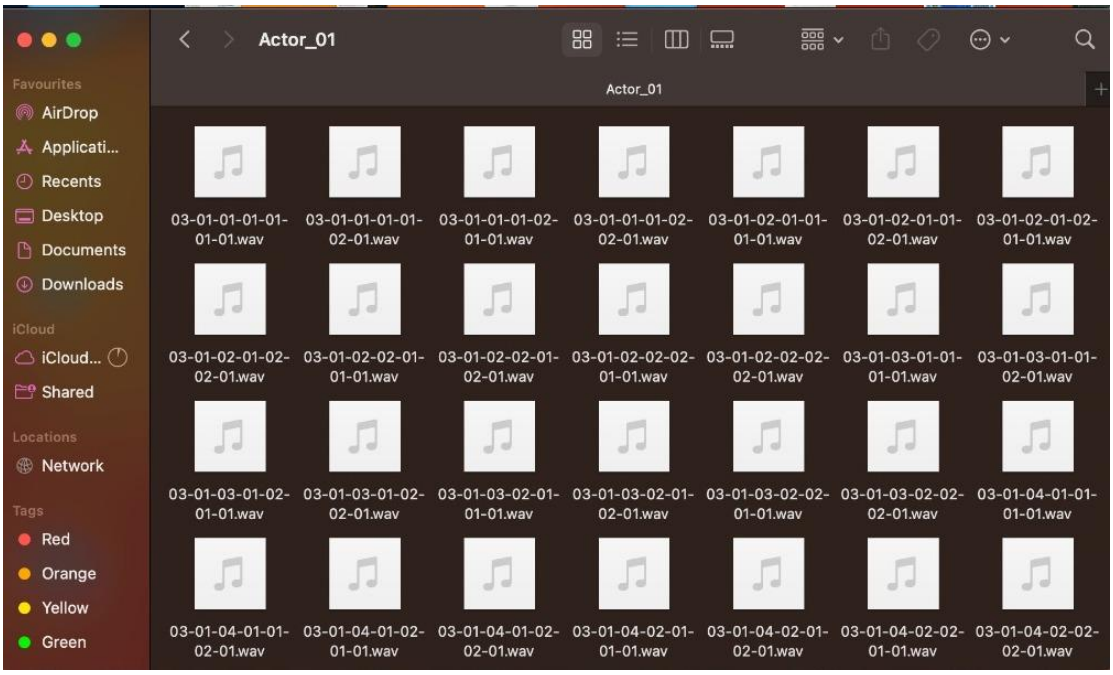

fig 3.3.2 Audio files of individual Actor

Filename identifiers

Modality (01 = full-AV, 02 = video-only, 03 = audio-only).

Vocal channel (01 = speech, 02 = song).

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

Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.

Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").

Repetition (01 = 1st repetition, 02 = 2nd repetition).

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

Filename example: 02-01-06-01-02-01-12.mp4

3.3.2 DATA PREPROCESSING

Real-world data collection has its own set of problems. It is often very messy which includes missing data, presence of outliers, unstructured manner, etc. Before looking for any insights from the data, we have to first perform preprocessing tasks which then only allow us to use that data for further observation and train our machine learning model. We use missing values treatment, outliers detection, normalization and data split to process our data before feeding it to the machine learning model. we used librosa and noisereduce libraries inorder to clean the speech.

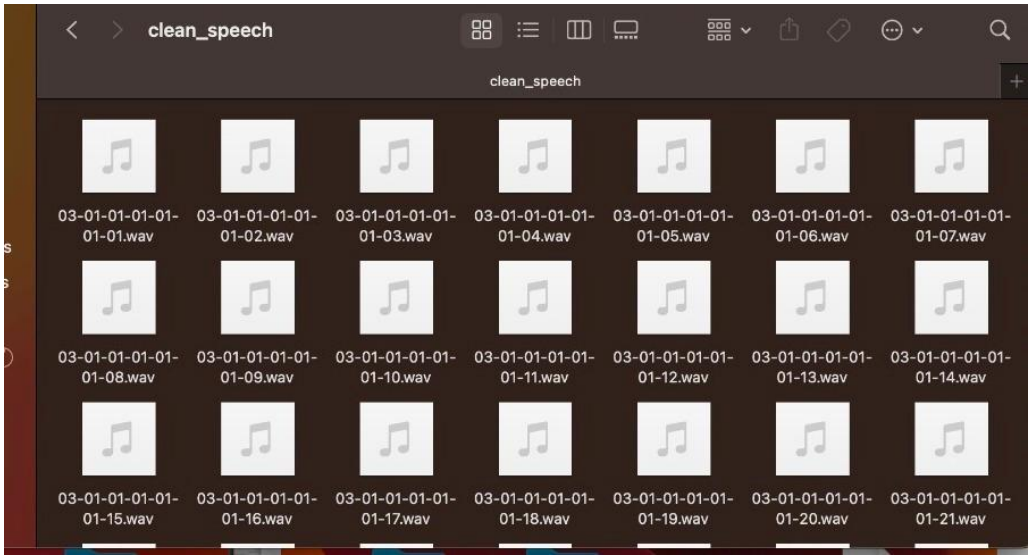

fig 3.3.3 Data after Preprocessing

Classifcleaning step is performed where down sampling of audio files is done and put mask over it and direct into clean folder and mask is to remove unnecessary empty voives around the main audio voice.

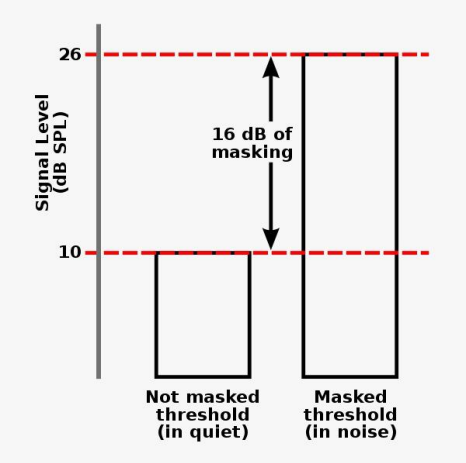

Fig 3.3.4 depicting masked and not masked threshold

In the presence of a masking noise (for example, a vacuum cleaner that is running simultaneously) that same individual cannot detect the sound of the cat scratching unless the level of the scratching sound is at least 26 dB SPL. We would say that the unmasked threshold for that individual for the target sound (i.e., the cat scratching) is 10 dB SPL, while the masked threshold is 26 dB SPL. The amount of masking is simply the difference between these two thresholds: 16 dB.

CHAPTER - 4

SIMULATION RESULTS AND ANALYSIS

4.1 Simulation results and ananysis

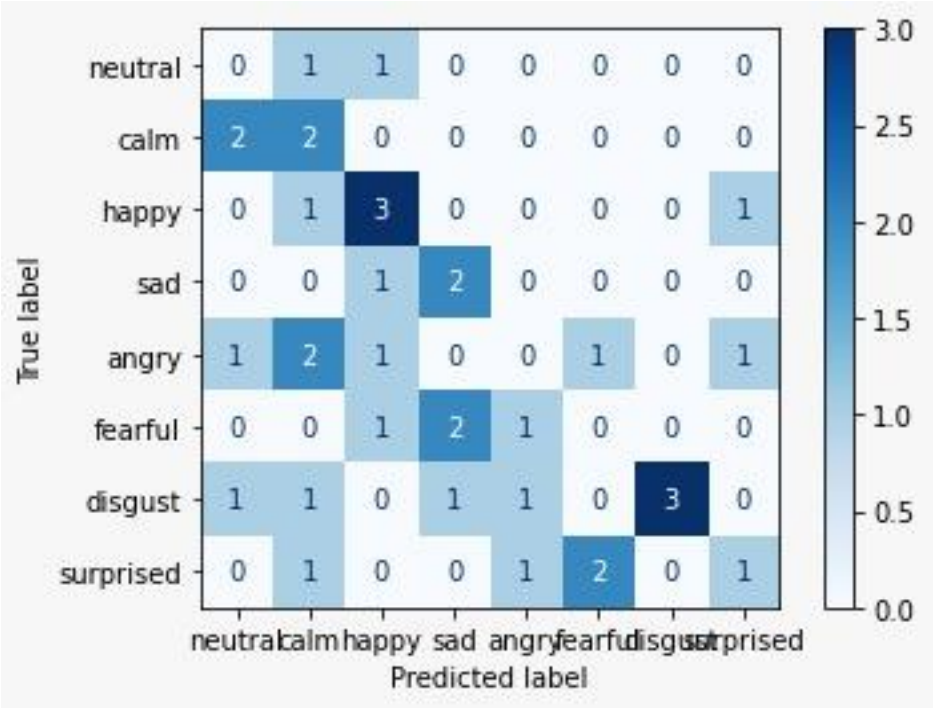

Fig 4.1.1 Confusion matrix

This is the confusion matrix , it visualizes and summarizes the performance of a classification algorithm where the Predicted label is on the x-axis and True label is on the y-axis.

```
#Calculate the accuracy of our model
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

#Print the accuracy
print("Accuracy: {:.2f}%".format(accuracy*100))
```

[80] ✓ 0.0s Python Python

... Accuracy: 91.19%

Fig 4.1.2 Accuracy using MLPC

Image showing the accuracy using Multilayer Perceptron Model and it is 91%.

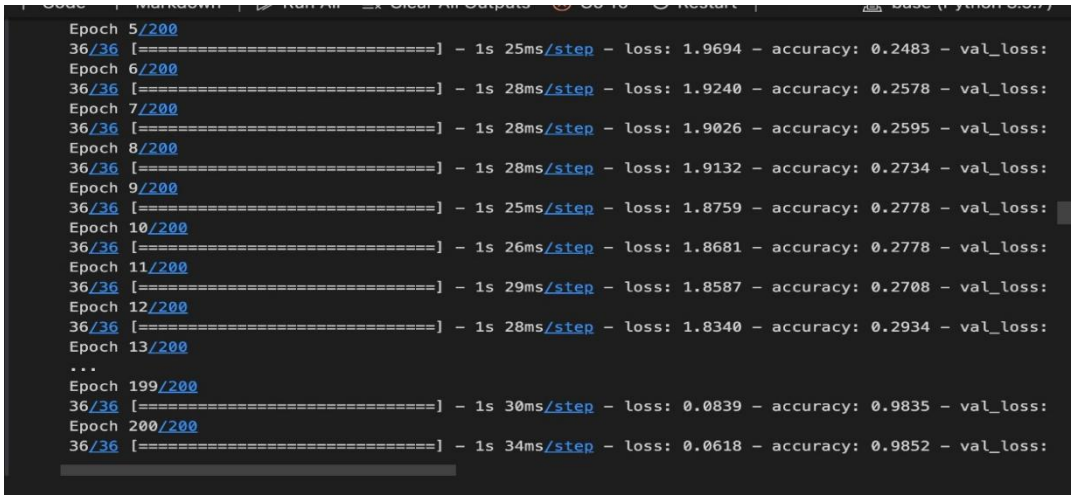

Fig 4.1.3 Accuracy using LSTM

This image shows accuracy using LSTM model by running 200 epochs where loss is 0.06 and value loss is 9.9, value accuracy is 0.30.

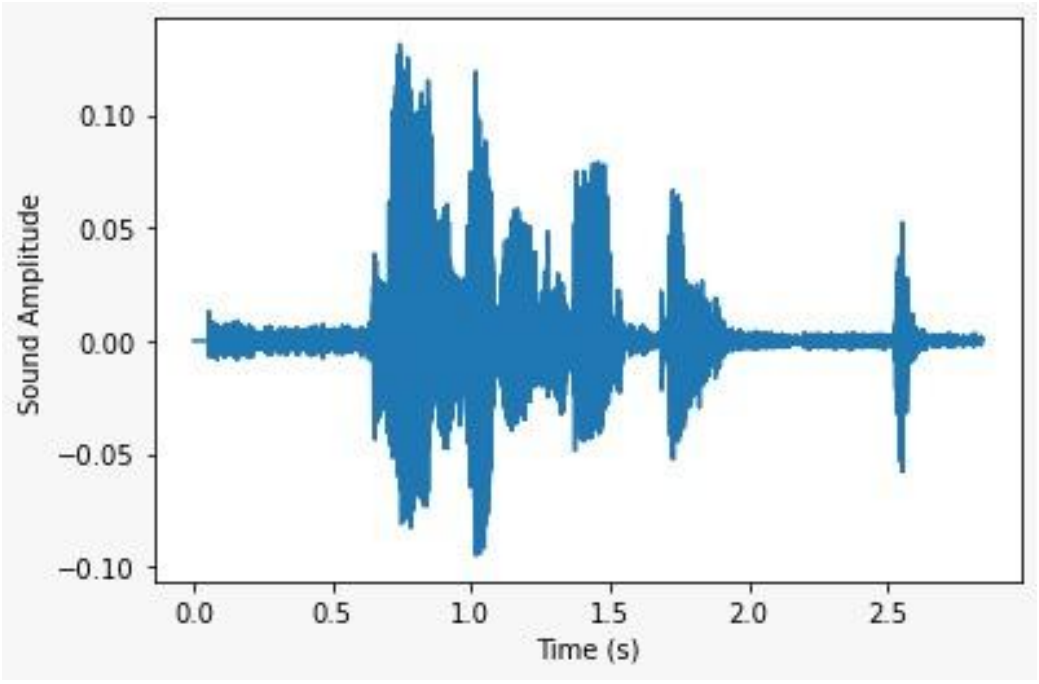

Fig 4.1.4 Sound Amplitude vs Time Graph

This is the graph displaying sound Amplitude vs Time of an audio file.

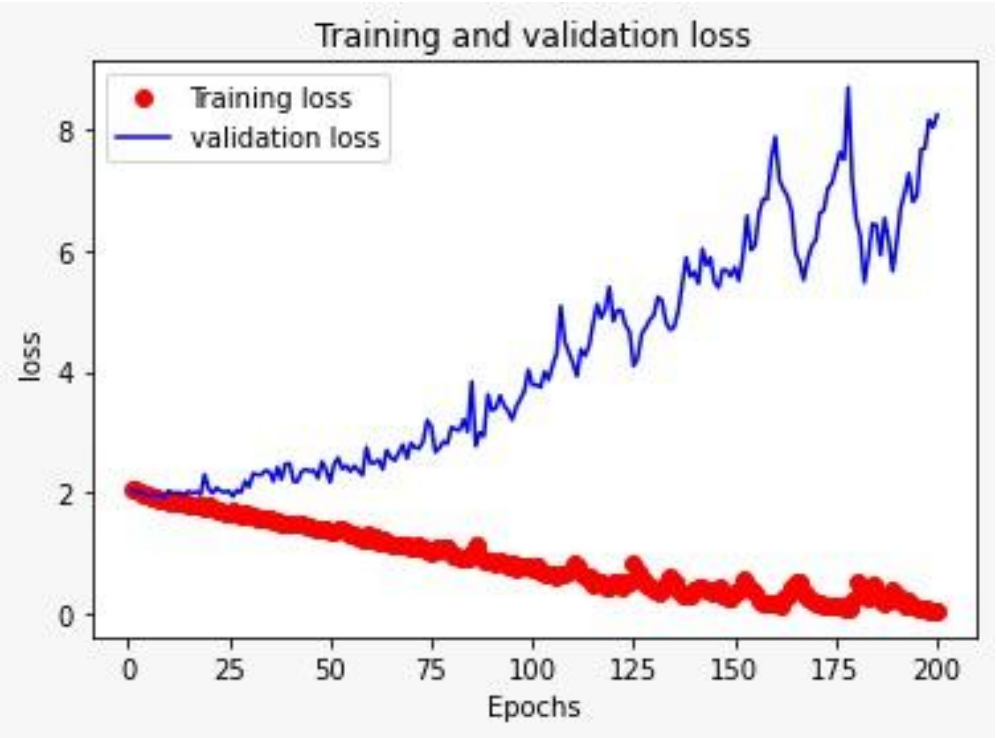

Fig4.1.5 Training and validation loss

It is a graph displaying training and validation loss where epochs are taken on X-axis and loss is taken on Y-axis.Here training accuracy is represented by red dot and validation accuracy is represented in blue line.

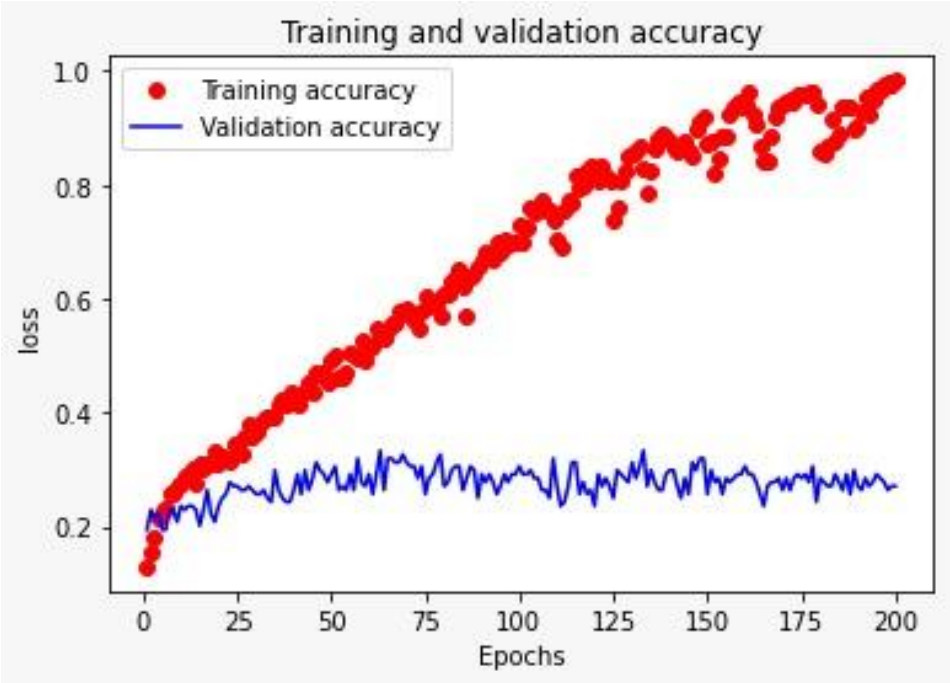

Fig 4.1.6 Training and validation accuracy

It is a graph displaying training and validation accuracy where epochs are taken on X-axis and loss is taken on Y-axis. Here training accuracy is represented by red dot and validation accuracy is represented in blue line.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 128)               66560

 dense (Dense)               (None, 64)                8256

 dropout (Dropout)           (None, 64)                0

 activation (Activation)     (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 dropout_1 (Dropout)         (None, 32)                0

 activation_1 (Activation)   (None, 32)                0

 dense_2 (Dense)             (None, 8)                 264

 activation_2 (Activation)   (None, 8)                 0

=================================================================
Total params: 77,160
Trainable params: 77,160
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.1.7 Model Summary

Model summary of LSTM model showing Layers, output shape and total trainable,non-trainable parameters.

	precision	recall	f1-score	support
calm	0.69	0.92	0.79	38
disgust	0.71	0.65	0.68	46
fearful	0.44	0.54	0.48	26
happy	0.76	0.50	0.60	44
accuracy			0.66	154
macro avg	0.65	0.65	0.64	154
weighted avg	0.67	0.66	0.65	154

```
[[35  1  2  0]
 [ 9 30  3  4]
 [ 3  6 14  3]
 [ 4  5 13 22]]
```

Fig 4.1.8 Classification report

It is the classification report which shows the performance evaluation of the model. It is used to show the precision, recall, F1 Score, and support.

	Actual	Predicted
0	calm	calm
1	fearful	fearful
2	disgust	disgust
3	disgust	calm
4	happy	fearful
5	fearful	fearful
6	calm	calm
7	happy	happy
8	disgust	disgust
9	calm	calm
10	happy	disgust
11	disgust	fearful
12	disgust	disgust
13	calm	calm
14	happy	happy
15	disgust	disgust
16	fearful	disgust
17	happy	fearful
18	disgust	disgust

Fig 4.1.9 Actual emotion vs Predicted emotion Table

This is the table displaying actual vs predicted emotions

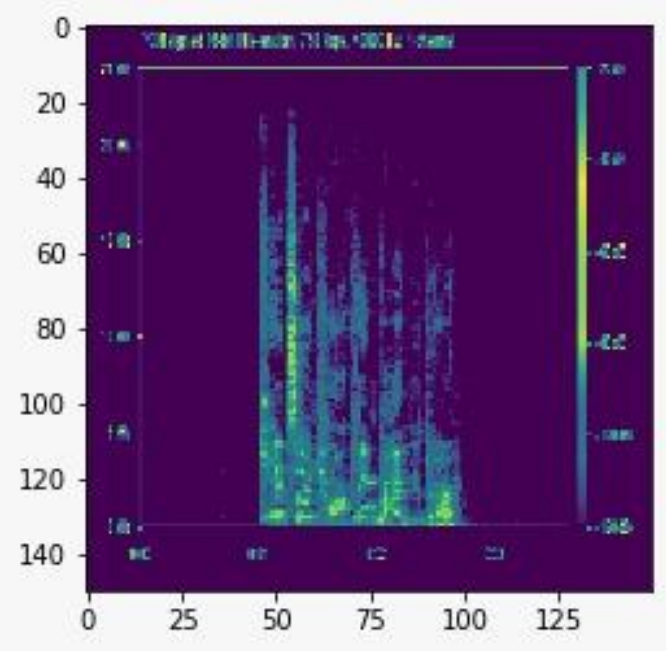

Fig 4.1.10 matplotlib function

The matplotlib function imshow() creates an image from a 2-dimensional numpy array. The image will have one square for each element of the array. The color of each square is determined by the value of the corresponding array element and the color map used by imshow() .

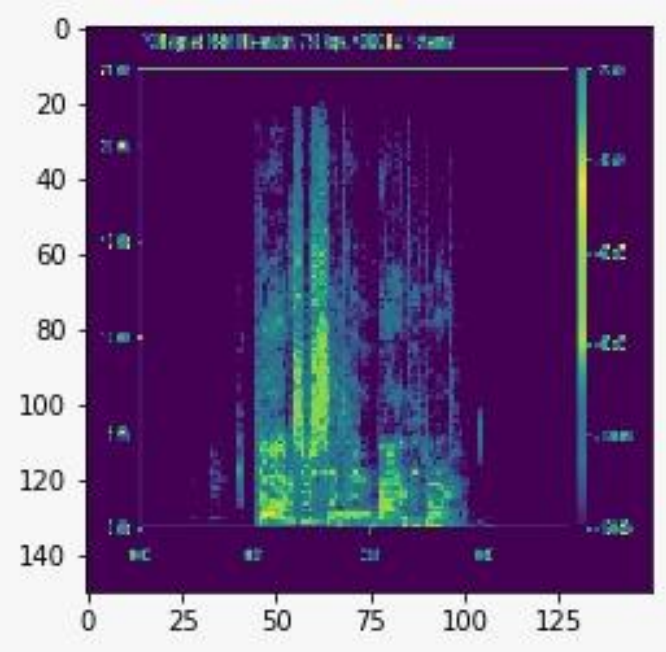

Fig 4.1.11 Matplotlib function inshow

- There are two common representations for RGB images with an alpha channel:

- Straight (unassociated) alpha: R, G, and B channels represent the color of the pixel, disregarding its opacity.
- Premultiplied (associated) alpha: R, G, and B channels represent the color of the pixel, adjusted for its opacity by multiplication.
- .imshow expects RGB images adopting the straight (unassociated) alpha representation.

Model summary of convolutional neural network consisting of number of layers, output shape and parameter.

```
...   Output exceeds the size limit. Open the full output data in a text editor
      Model: "sequential_3"
      _________________________________________________________________
      Layer (type)                 Output Shape              Param #
      =================================================================
      conv1d_5 (Conv1D)            (None, 180, 128)          768
      _________________________________________________________________
      activation_7 (Activation)    (None, 180, 128)          0
      _________________________________________________________________
      dropout_5 (Dropout)          (None, 180, 128)          0
      _________________________________________________________________
      max_pooling1d_3 (MaxPooling1 (None, 22, 128)           0
      _________________________________________________________________
      conv1d_6 (Conv1D)            (None, 22, 128)           82048
      _________________________________________________________________
      activation_8 (Activation)    (None, 22, 128)           0
      _________________________________________________________________
      max_pooling1d_4 (MaxPooling1 (None, 2, 128)            0
      _________________________________________________________________
      dropout_6 (Dropout)          (None, 2, 128)            0
      _________________________________________________________________
      conv1d_7 (Conv1D)            (None, 2, 128)            82048
      _________________________________________________________________
      activation_9 (Activation)    (None, 2, 128)            0
      _________________________________________________________________
      dropout_7 (Dropout)          (None, 2, 128)            0
      ...
      Total params: 166,920
      Trainable params: 166,920
      Non-trainable params: 0
      _________________________________________________________________
```

Fig 4.1.12 Model summary of convolutional neural network

```
              precision    recall  f1-score   support

         0.0       0.77      0.66      0.71        94
         1.0       0.81      0.70      0.75       101
         3.0       0.67      0.77      0.72        44
         4.0       0.75      0.91      0.82        90

    accuracy                           0.76       329
   macro avg       0.75      0.76      0.75       329
weighted avg       0.76      0.76      0.75       329

[[62  9  5 18]
 [ 9 71 12  9]
 [ 4  6 34  0]
 [ 6  2  0 82]]
```

Fig 4.1.13 Classification report of cnn.

```
              precision    recall  f1-score   support

        calm       0.69      0.92      0.79        38
     disgust       0.71      0.65      0.68        46
     fearful       0.44      0.54      0.48        26
       happy       0.76      0.50      0.60        44

    accuracy                           0.66       154
   macro avg       0.65      0.65      0.64       154
weighted avg       0.67      0.66      0.65       154

[[35  1  2  0]
 [ 9 30  3  4]
 [ 3  6 14  3]
 [ 4  5 13 22]]
```

Fig 4.1.14 Classification report of randomforest.

- The accuracy is 65.

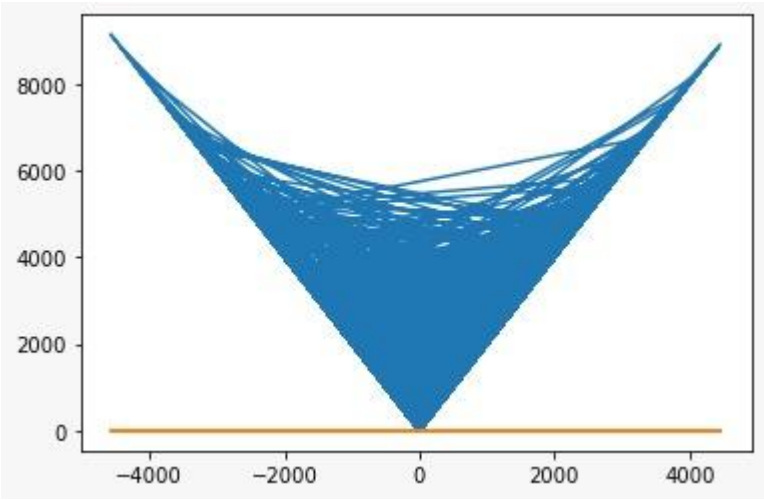

Fig 4.1.15 Mfcc graph after masking.

From the above audio/speech file we can observe that is more complex then simple sine wave. This is also combination of different sine waves with different Amplitude.

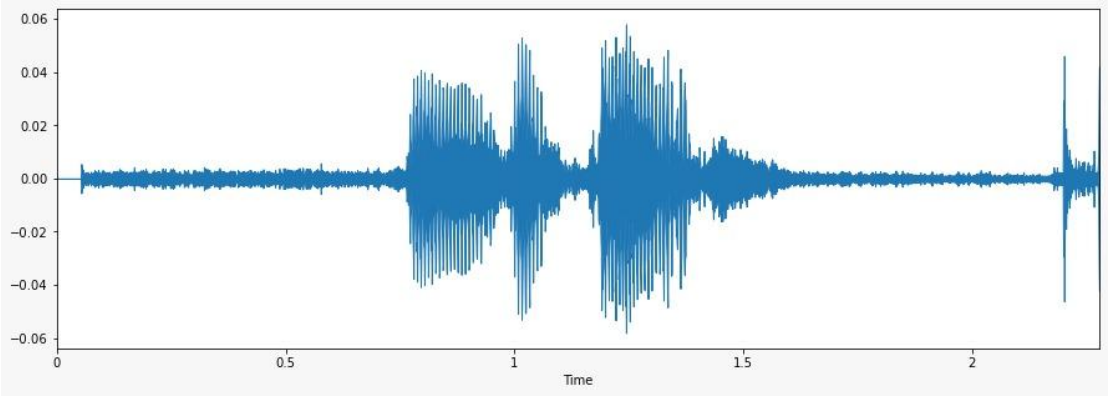

Fig 4.1.16 Amplitude vs time of recorded sound file.

4.2 USER INTERFACE :

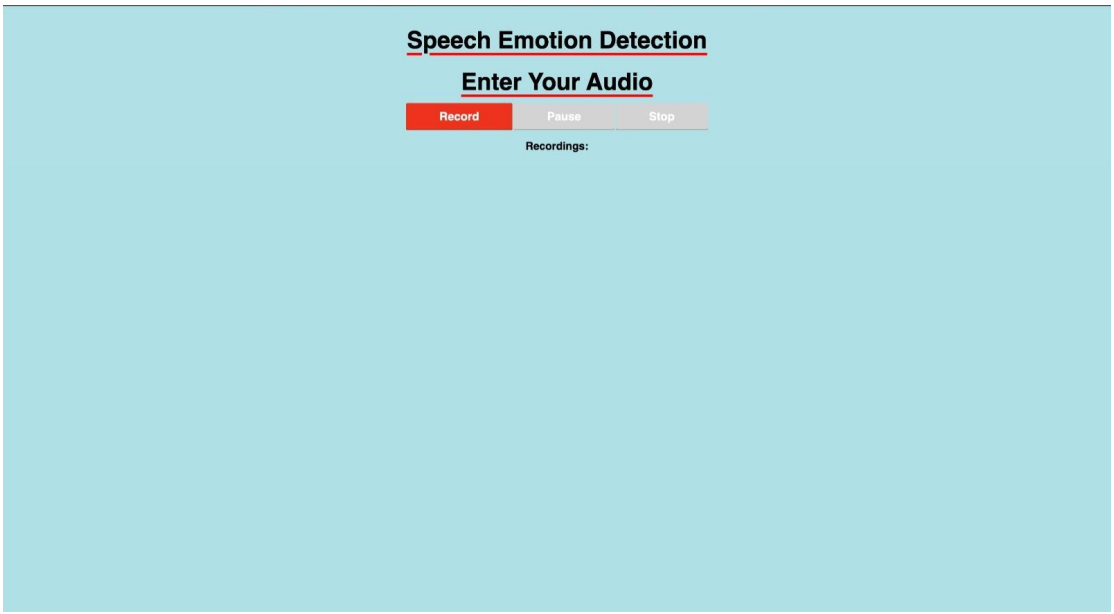

Fig 4.2.1 UI To record and save custom audio file.

This is the output after the running the html page consisting of Record,Pause and Stop buttons. Record button is in the red colour and its function is to capture the audio file, pause button is used to halt the current recording and stop button will be saving the audio file whenever it is clicked. We have used the cdn to convert our original audio file into .wav format

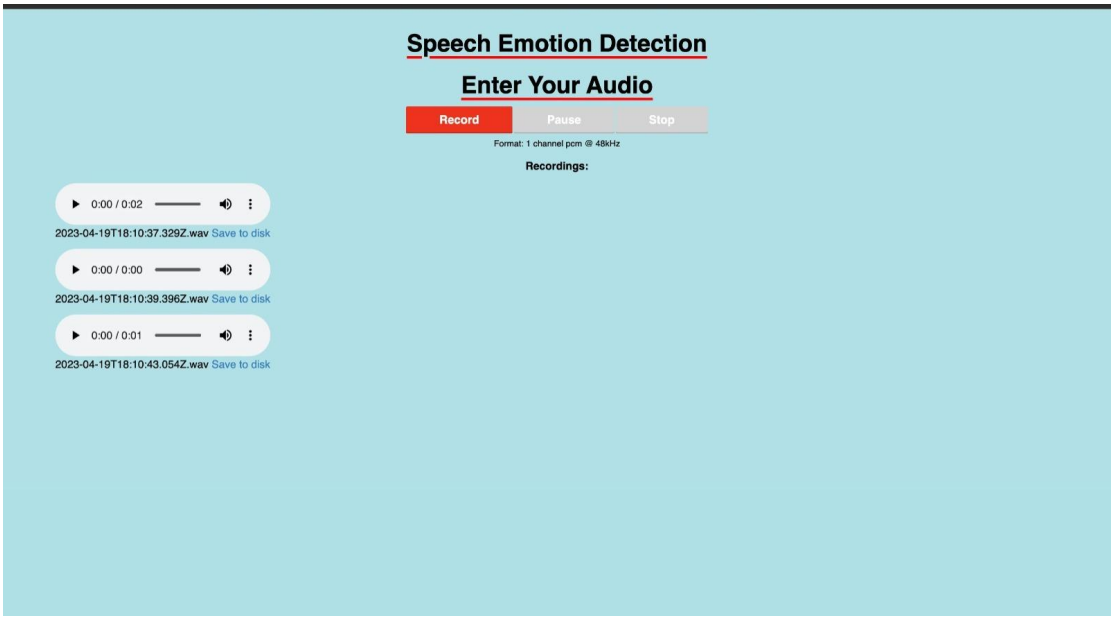

Fig 4.2.2 Recorded audio files

This image displays the saved audio files after clicking the stop button.These audio files in will be saved as .wav format which is used to test the emotion. We have the chance to listen these saved audio files and shows the options like control the playback speed, download and save to disk that is to the internal storage.

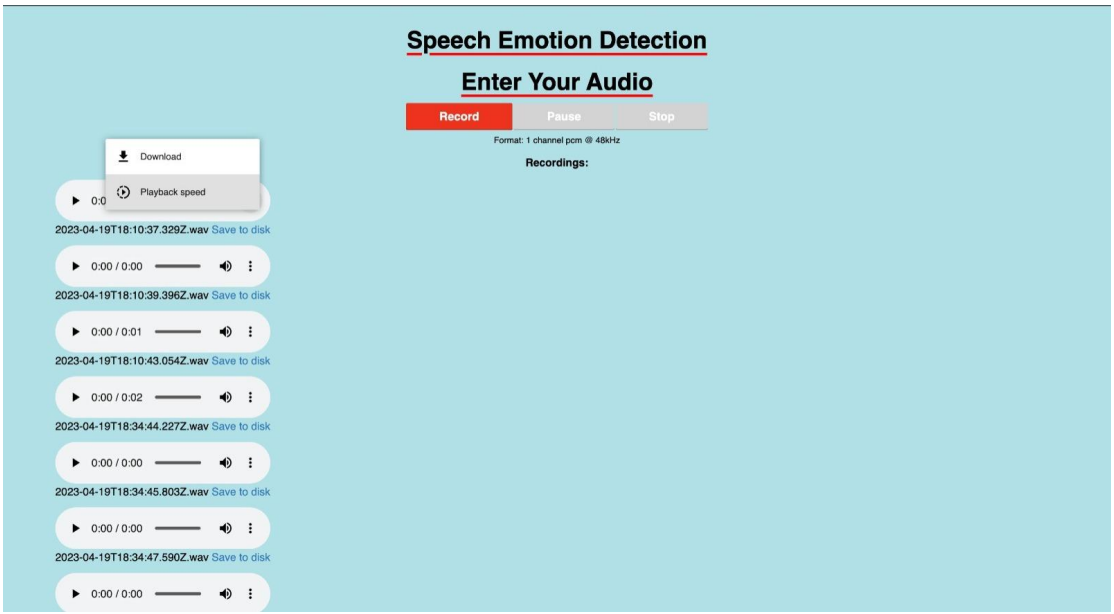

Fig 4.2.3 Added Recorded files

The User Interface which takes the input audio .wav format file.After submitting the audio file in.wav format via the user interface, we will receive the result of the identified emotion. The maximum size of the audio file to be dropped is upto 200Mb .

The images of the predicted emotions (i.e happy,sad,calm,disgust) after uploading different audio files are shown below:

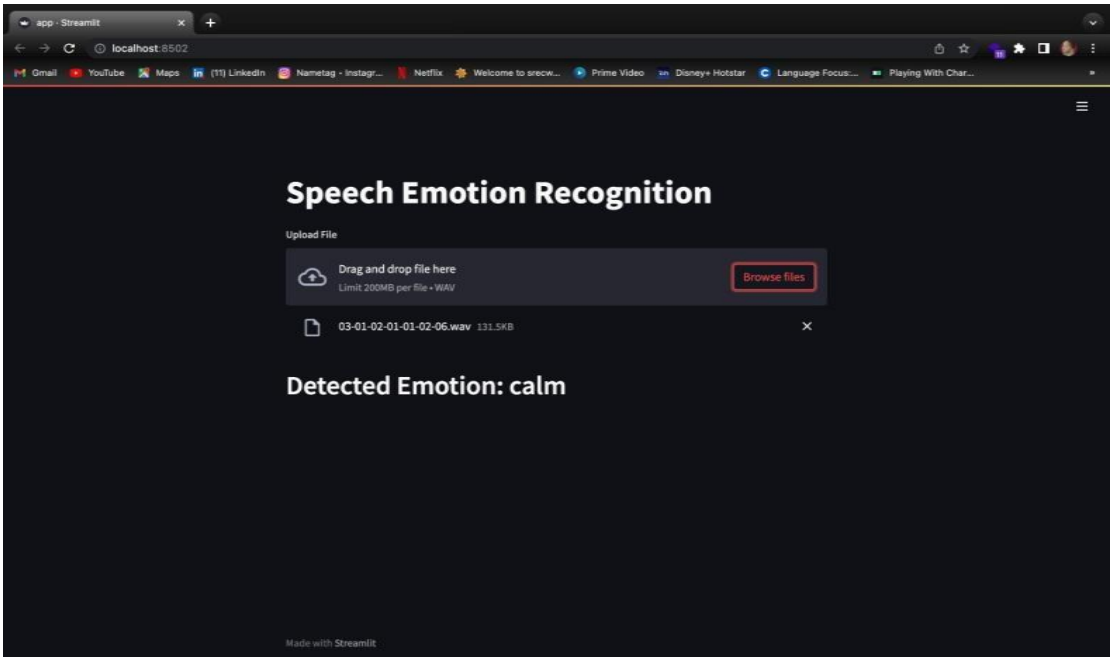

Fig 4.2.4 Emotion detetcion calm

- Detected emotion is "calm".

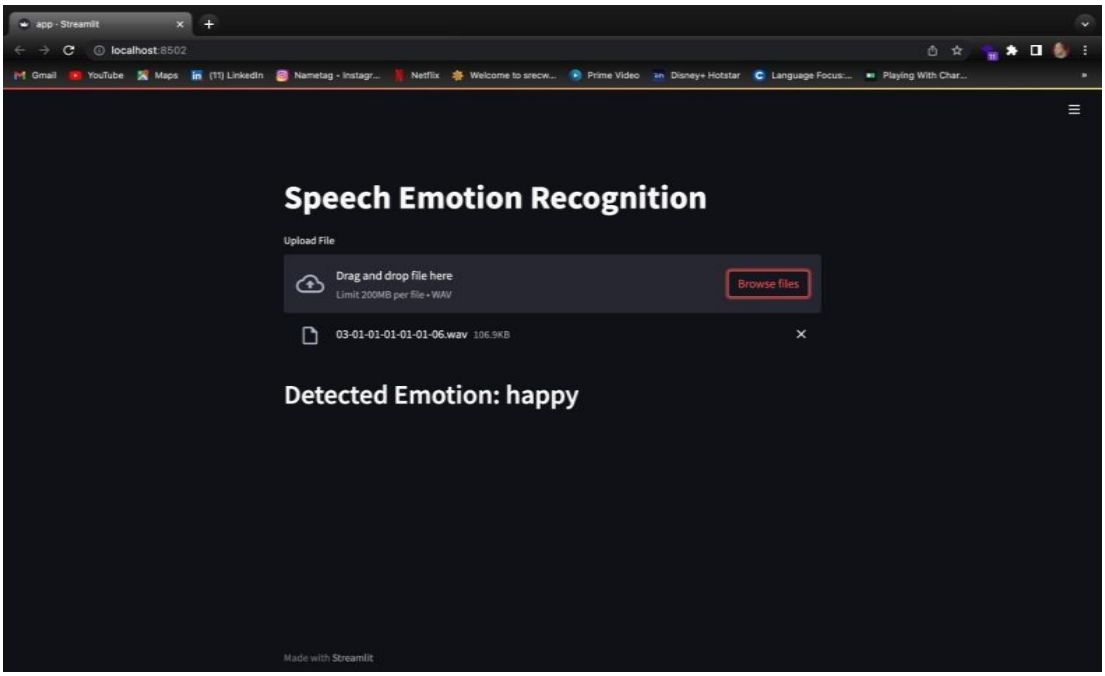

Fig 4.2.5 emotion detection happy

- Detected emotion is "Happy".

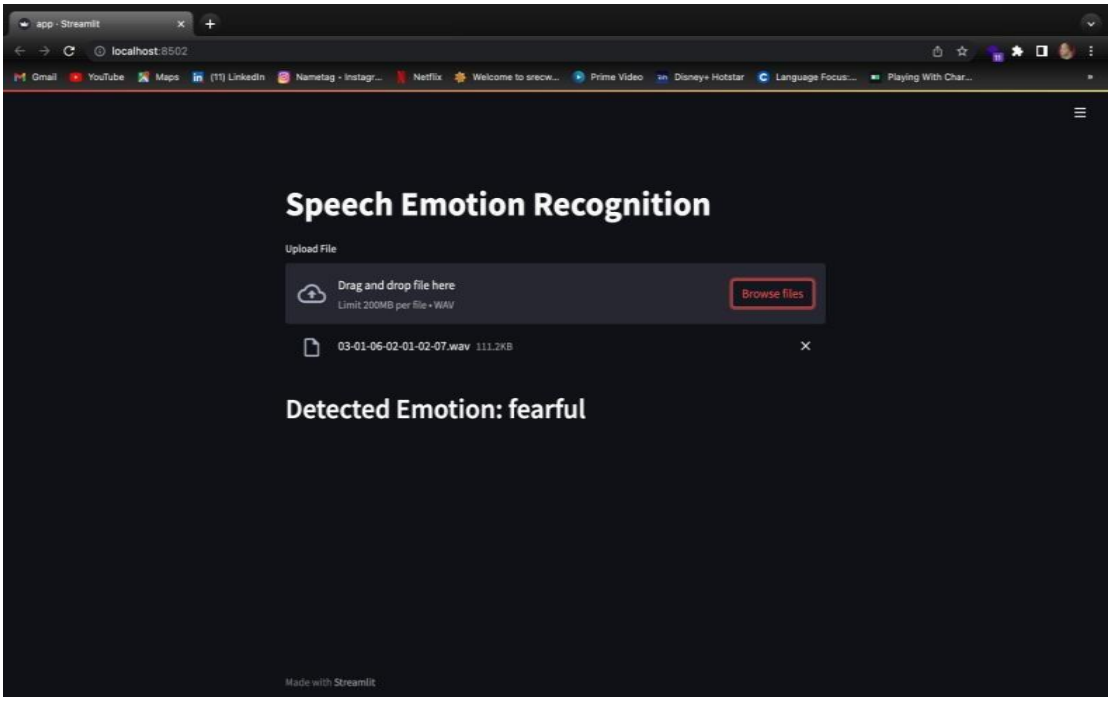

Fig 4.2.6 emotion detection Fearful

- Detected emotion is “Fearful”.

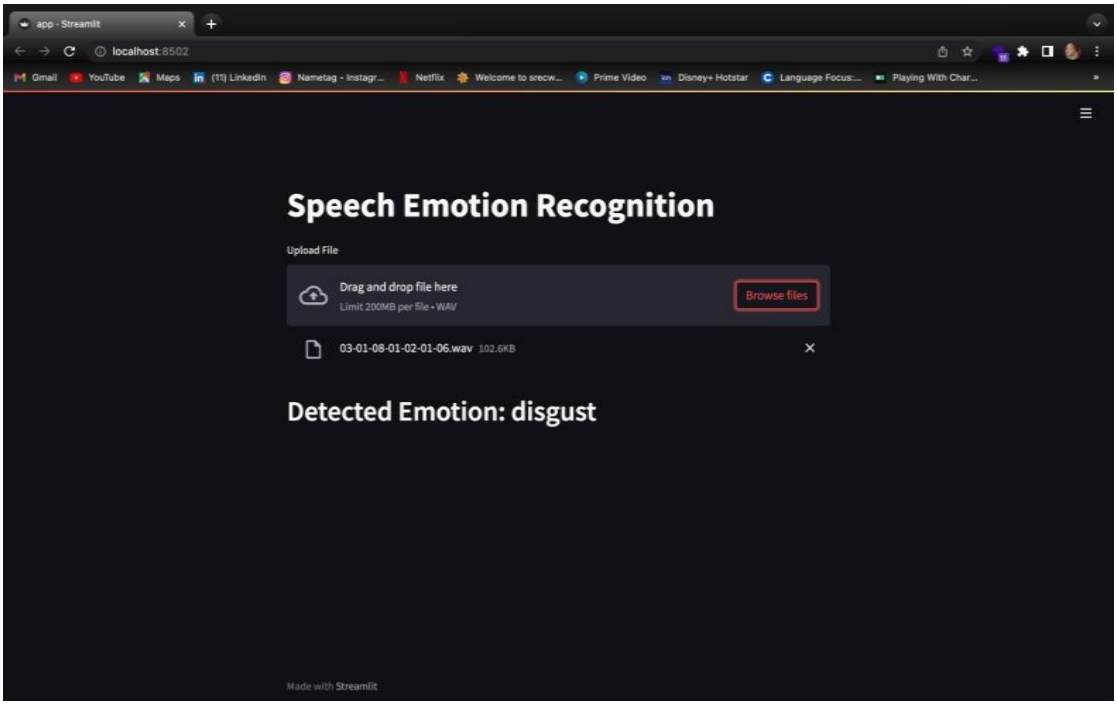

Fig4.2.7 Emotion detetction disgust

- Detected emotion is “Disgust”.

CHAPTER - 5
FUTURE SCOPE AND CONCLUSION

To understand the emotions of people will help in achieving greater success in the lives of people.The Proposed Model is used to detect emotion of a person based on his speech. In this process the model was trained with different emotions to make sure it gives the proper results. The model used is MLP. This could help communicate and build good relationship with people. Because emotions in a speech play a vital role in analyzing the actual message of the speaker.

Testing on a voice platform is a great learning experience for a test engineer due to its unique testing process and the unexpected findings that come with it. Every project adds value to the platform and creating testing strategies. Testing bleeding edge technology requires patience, different levels of creativity and techniques to make sure the skill is ready for the most intense user.

Voice recognition technology is the future and by finding out and solving problems with through testing various voice applications, we will enable this technology to be very useful and dependable for its users.

BIBILOGRAPHY

[1] Björn Schuller;Felix Burkhardt, “Learning with synthesized speech for automatic emotion recognition”Schuller, B., & Burkhardt, F. (2010). Learning with synthesized speech for automatic emotion recognition. 2010 IEEE International Conference on Acoustics, Speech and Signal Processing. doi:10.1109/icassp.2010.549501

[2] .C. Vásquez-Correa; N. García.,Emotion recognition from speech under environmental noise conditions using wavelet decomposition,Emotion recognition from speech under environmental noise conditions using wavelet pp. 247-252, doi: 10.1109/CCST.2015.7389690

[3] Speech based human emotion recognition using MFCC,. S. Likitha, S. R. R. Gupta, K. Hasitha and A. U. Raju, "Speech based human emotion recognition using MFCC," 2017 International Conference on Wireless Communications, S ignal Processing and Networking (WiSPNET), Chennai, India, 2017, pp. 2257-2260, doi: 10.1109/WiSPNET.2017.8300161

[4] Danai Styliani Moschona,An Affective Service based on Multi-Modal Emotion Recognition, using EEG enabled Emotion Tracking and Speech Emotion Recognition,https://doi.org/10.1109/ICCE-Asia49877.2020.9277291

[5] Linlin Chao,Jianhua Tao,Long short term memory recurrent neural network based encoding method for emotion recognition in video,https://doi.org/10.1109/ICASSP.2016.7472178

[6] Woo-Seok Lee, Yong-Wan Roh, Dong-Ju Kim,Speech Emotion Recognition Using Spectral Entropy,https://doi.org/10.1007/978-3-540-88518-4_6

[7] Sung-Lin-Yen, Chi-Chun-Lee “A Dialogical Emotion Decoder for Speech Emotion Recognition in Spoken Dialog” In 2020 International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020.

[8] Linlin Chao,Jianhua Tao “Long short term memory recurrent neural network based encoding method for emotion recognition in video” In 2016 International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2016

[9] Woo-Seok Lee, Yong-Wan Roh, Dong-Ju Kim Speech Emotion Recognition Using Spectral Entropy" Intelligent Robotics and Applications. ICIRA 202008 Springer-Verlag Berlin Heidelberg.

[10] J. G. Rázuri, D. Sundgren, R. Rahmani, A. Moran, I. Bonet, and A. Larsson, "Speech emotion recognition in emotional feedback for human-robot interaction," International Journal of Advanced Research in Artificial Intelligence (IJARAI), vol. 4, no. 2, pp. 20–27, 2015.

[11] D. Le and E. M. Provost, "Emotion recognition from spontaneous speech using hidden markov models with deep belief networks," in 2013 IEEE Workshop on Automatic Speech Recognition and Understanding. IEEE, 2013, pp. 216–221.

[12] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," IEEE Access, vol. 7, pp. 19 143–19 165, 2019.