CHAPTER-1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction

Emotions are perceived by every human brain around the globe within its recognizable environment. Every action proposition is based on the inference of the perceived emotion. These directly affect the state of reality. To have success in every walk of life one needs to master the capacity to be aware of, control, and express one's emotions, and to handle interpersonal relationships judiciously and empathetically. Using the modern day computational tools emotion analysis can be a powerful aresenary.

The communication among people and PCs has gotten a great deal of consideration off late. It is a standout amongst the most well known zones of research and has extraordinary potential. Showing a PC the comprehension of human feelings is an essential part of this cooperation. A ton of effective applications identified with discourse acknowledgment are accessible in the market. Individuals can utilize their voice to offer directions to vehicle, PDAs, PC, TV and numerous electrical gadgets. Subsequently, to influence a PC to comprehend human feelings and give a superior cooperation experience turns into a fascinating test.

The most well-known approach to perceive any discourse feeling is extricating imperative highlights that are identified with different passionate states from the discourse flag (for example vitality is an essential element to recognize bliss from misery), feed these highlights to the info end of a classifier and acquire diverse feelings at the yield end. This procedure is appeared in the figure underneath.

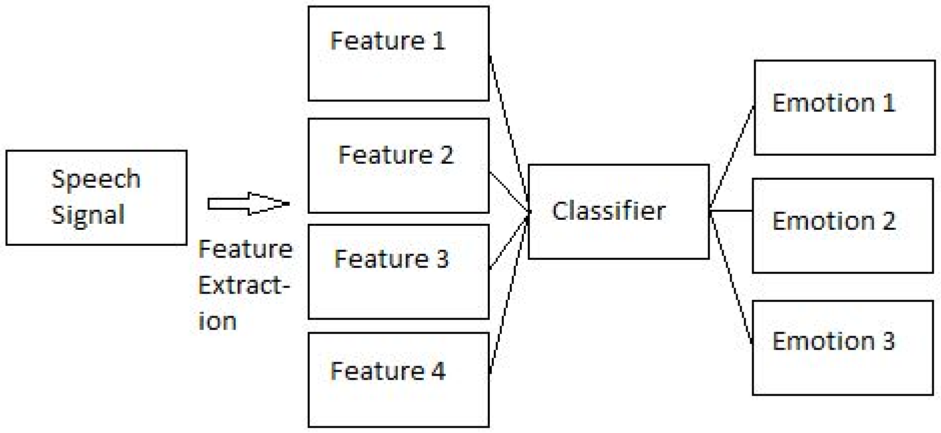


Figure 1.1 Flow of emotion recognition and classification

The point is to group a bunch of recorded discourse motion into conceivable classifications of feelings, in particular: upbeat, miserable, irate, common. Prior to extraction, pre-preparing is performed on the discourse signals. Tests are taken from the discourse and the simple flag is changed over to advanced flag. At that point each sentence is standardized to guarantee that every one of the sentences are in a similar volume go. Finally, division isolates motion in casings with the goal that discourse flag can keep up its qualities in brief term. Regularly utilized highlights are picked for study and in this manner removed. Vitality is the most fundamental component of discourse flag. Contribute is every now and again utilized this point and autocorrelation is utilized to recognize the contribute each casing. After autocorrelation, factual qualities are determined for discourse signals. Formant is another imperative component. Direct Predictive Coding (LPC) technique is utilized to extricate the first formant. Like pitch, factual qualities are determined for first formant. Mel recurrence cepstral coefficient (MFCC) is a portrayal of momentary power range on a human-like mel size of recurrence. Initial three coefficients of MFCCs are taken to infer means and differences. All highlights of the discourse tests are put into Artificial Neural Network (ANN), which comprises of an information lattice alongside an objective framework, which show the feeling state for each sentence made the contribution out of neural system.

1.2 Proposed System

The emotions of a person influence physical aspects like muscular tension, skin elasticity, blood pressure, heart rate, breath, tone of voice etc. This Project presents the implementation of emotion detection from voice with a deep Convolutional Neural Network architecture (CNN) that process and classifies voice samples. The architecture is an adaptation of an image processing CNN, programmed in Python.

- **Package Extraction Module:**

  Different packages which achieve the successful implementation of this project are extracted.

  Functions:

  1.Packages like Librosa, TensorFlow, numpy etc.. are extracted in the anaconda navigator for training module.

  2.The same packages are extracted in the system's command prompt for testing module.

- **Training Module:**

  Using the dataset ravedess build a approximate model for emotion analysing.

  Functions:

  1. Preprocessing of data sets for building the model.

  2.Classifying the data set into training and testing using decision tree classifier.

  3. Extracting MFCC Coefficients from the dataset.

  4. Construct a CNN Neural Network Model.

  5.Save the model.

  6.Reload the model for testing module.

- **Prediction Module:**

  Analyses the input voice data and predicts the emotion

Functions:

1. After reloading the model , using the GUI load a data file.
2. Submit this data file for analysis.
3. In the output section the emotion analysed is printed.

- **Recording Module:** Records user voice inputs

  Functions:

  1. Provides a feasibility to record different voice samples.
  2. Records the voice sample and stores them for testing module.

Key Factors Influencing the Proposed System

- Reliability

The solution should provide reliability to the user that it will run with all the features mentioned available and executing perfectly.

- Accuracy

The solution should be able to reach the desired level of accuracy. But also keeping in mind that this version is for proving the concept of the project.

- Generalization

The effectiveness of the model to generalize for real time data sets is questionable as the performance criteria might be not up to the mark

CHAPTER-2

FEASIBILITY STUDY

2. FEASIBILITY STUDY

A feasibility study involves taking a judgment call on whether a project is doable. The two criteria to judge feasibility are cost required and value to be delivered. A well-designed study should offer a historical background of the business or project, a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, such studies precede technical development and project implementation.

A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions.

2.1 Technical Feasibility

Technical feasibility involves evaluation of the hardware and the software requirements of the proposed system. In this project, the technologies involved are Anaconda Navigator, Python 3.6 and Command Prompt.The system should have intel i5 or higher processor and minimum of 8Gb RAM with a speed of 2.1GHZ. These play a major role in fulfilling the project requirements with good accuracy and within minimal time. They can convert ideas into working systems. The language used is Python for Front end development, packages like Librosa, TensorFlow are used for interaction to the modules.

2.2 Economic Feasibility

Economic Feasibility helps in assessing the viability, cost, and benefits associated with projects before financial resources are allocated. This assessment typically involves a cost/ benefits analysis of the project.

The application is so designed that it requires minimal cost and eliminates costs as there would minimal need for manual work. The technologies used helps in understanding the user without any investment. The Prediction module helps in receiving the input from the users easily and it can be easily analyzed and displays the Predicted result. Also, the Recording module, helps in recording unique speeches for the users and test them in the Prediction Module.

2.3 Legal Feasibility

The proposed system doesn't conflict with legal requirements like data protection acts or social media laws. It ensures legal data access and gives prominence to data security.

2.4 Operational Feasibility

The application involves design-dependent parameters such as reliability, maintainability, supportability, usability, disposability, sustainability, affordability, and others. It minimizes the drawbacks of the current system by building an application that automatically resolves the user queries and helps to analyze the user data.

2.5 Scheduling Feasibility

This assessment is the maximum essential for assignment fulfillment; in any case, a mission will fail if not completed on time. In scheduling feasibility and corporation estimates how a whole lot time the undertaking will take to complete.

CHAPTER-3
LITERATURE SURVEY

3. LITERATURE SURVEY

3.1 Literature Survey

3.1.1 Survey from A Proposed Approach with Analysis of Speech Signals for Sentiment Detection(Atul Tyagi,Nidhi Chandra)

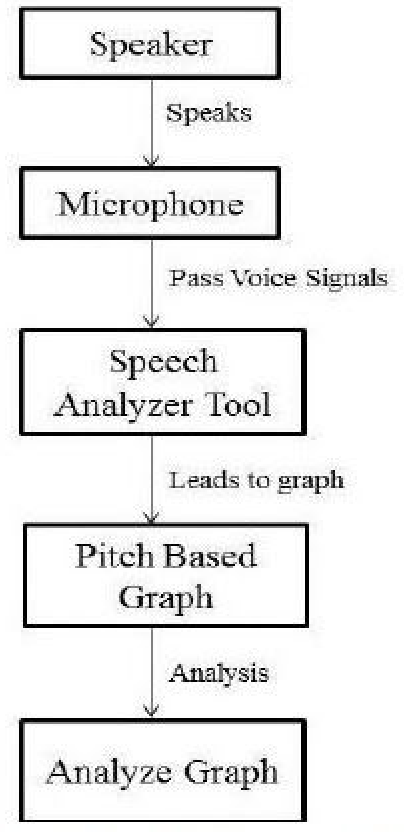


Figure 3.1 Flow chart for the complete process

A Speaker speaks or passes speech signals in form of words to the listener because of which communication holds. Listener decodes these signals and recognizes what speaker wants to say.

Whenever you speak, based on your pitch of voice, based on the speed of voice; listener can easily predict what type of felling an individual have when she speaks.

So our voice could also be used for sentiment detection. Present Paper proposed method for Sentiment detection based on the Speech Signals and tries to show some results which show how pitch and time varies during different emotions. Its voice signals produced by the speaker which will help in recognizing sentiment behind the speaker words.

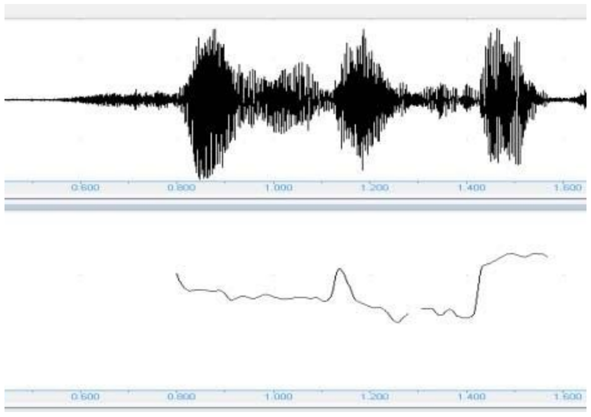


Figure 3.2 Speech signals when Angry

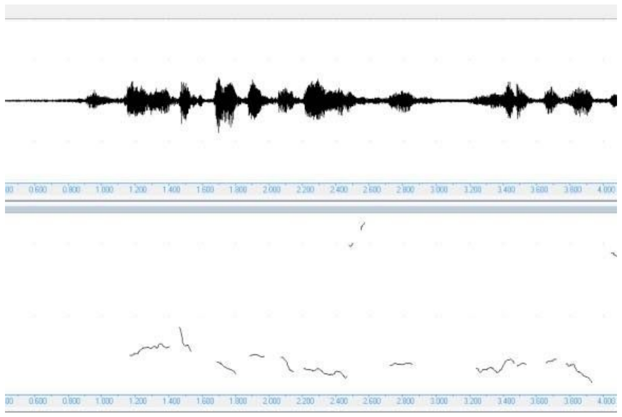


Figure 3.3 Speech signals when sad

The two parameters based on which signals will be analyzed are Pitch Frequency and Time Duration for completing a sentence. As somebody talks in various feeling, time and recurrence taken for a specific sentence comes to be diverse for. Along these lines, we chose to investigate signals dependent on Time and Pitch recurrence utilized by a person in various feelings. In the event that we dissect these above gave diagrams than we can see that in Figure 1 pitch recurrence is comes to be 225Hz approx. with time length as 1.58secs approx. while in Figure 2 pitch recurrence is 160HZ approx. what's more, time length is 5.4secs. From the investigation unmistakably there is variety in time for rivalry of sentence amid two unique conditions of feelings for example Furious and Sad

3.1.2 Survey from Detection and Analysis of Emotion From Speech Signals (Assel Davletcharova, Sherin Sugathan, Bibia Abraham, Alex Pappachen Jamesa)

For contemplating the fundamental idea of highlights in discourse under various passionate circumstances, we utilized information from three subjects. As a component of the information gathering, we recorded the voice of three distinctive female subjects. The subjects were approached to express certain feelings when the their discourse was recorded. The subjects were Russians and they communicated in Russian words under various enthusiastic states. A cell phone was utilized to record the discourse and was avoided at all costs about 15cms far from the mouth. The investigations were directed in a customary room having a zone of 25m2 . For extricating highlights from the recorded discourse sections, MATLAB capacities were utilized.

The analysis of the recorded speech signals were done in a MATLAB environment which provides several graphical visualizations for analysing a signal. In Figure.1(a), a power spectrum of the speech signal is shown which indicates the peak value using a dark red color. In Figure.1(b), a linear graphical representation of the power intensity is shown.

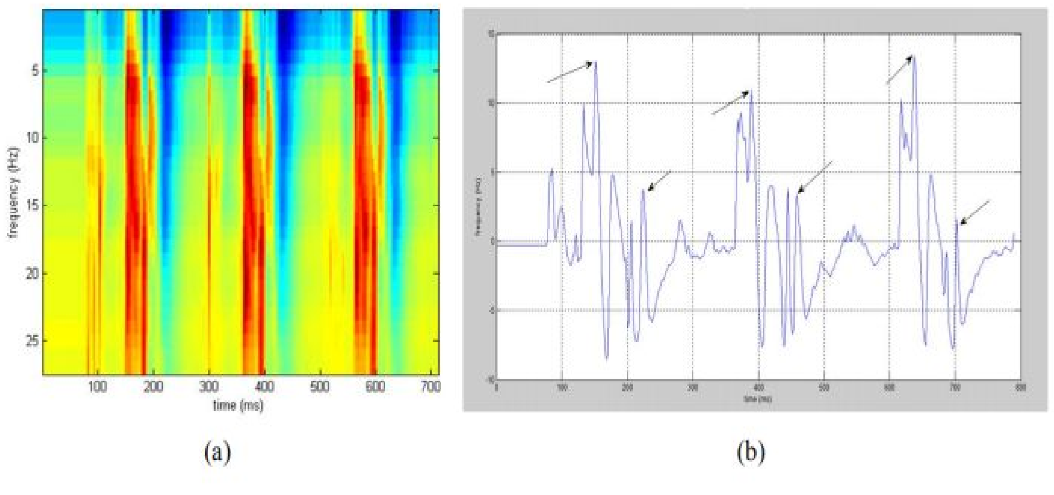


Figure 3.4 (a) The Power spectrum of a speech signal. (b) MFCC of a speech signal of a subject when one word was pronounced for three times.

The distance between the indicated peaks are used to form a feature vector.

The Mel-frequency cepstral coefficients (MFCC) are widely used in audio classification experiments due to its good performance. It removes and speaks to highlights of discourse flag. The Mel-cepstral takes brief time otherworldly shape with critical information about the nature of voice and generation impacts. To compute these coefficients the cosine change of genuine logarithm of the momentary range of vitality must be finished. At that point it is performed in mel-recurrence scale.

The readied dataset had three qualities including highlight remove, pulse and class. The class characteristic shows the passionate condition of the individual and is named as A, B and C for unbiased, outrage and satisfaction separately. The information was classifier utilizing weka programming. The information was run multiple times with the preparation information rate fluctuating from 10% to 90% with 10% advance. So as to check the viability of the model, the precision ought not diminish with changing the preparation set.

The paper investigated distinguishing the enthusiastic condition of an individual by discourse handling strategies. The investigation on words and letters under various passionate circumstances demonstrated that the enthusiastic state can change the discourse flag. It was seen that there are recognizable highlights in a discourse portion that portrays every feeling state.

After performing the classification tests on a dataset prepared from 30 subjects, it was observed that it is better considering data from an individual subject rather than a group of people. The development of a software based agent for emotion detection and heart rate analysis can greatly improve telemedicine based systems.

CHAPTER-4
SYSTEM ANALYSIS

4. SYSTEM ANALYSIS

4.1 System Requirements

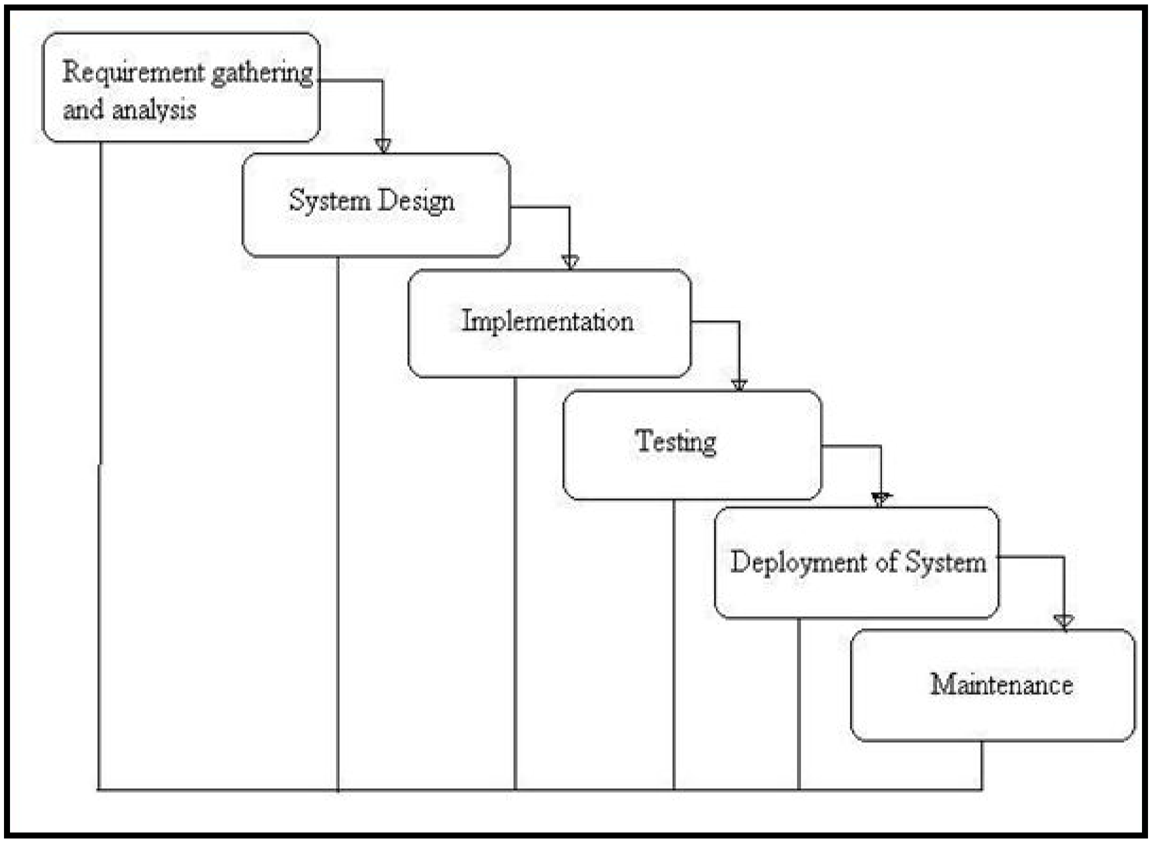


Figure 4.1 Waterfall Model

Project SDLC

- Project Requisites Accumulating and Analysis
- Application System Design
- Practical Implementation
- Manual Testing of My Application
- Application Deployment of System
- Maintenance of the Project

4.2 Requisites Accumulating and Analysis

It's the first and foremost stage of the any project as our is a an academic leave for requisites amassing we followed of IEEE Journals and Amassed so many IEEE Relegated papers and final culled a Paper designated by setting and substance importance input and for analysis stage we took referees from the paper and did literature survey of some papers and amassed all the Requisites of the project in this stage.

4.3 System Design

In System Design has divided into three types like GUI Designing, UML Designing with avails in development of project in facile way with different actor and its utilizer case by utilizer case diagram, flow of the project utilizing sequence, Class diagram gives information about different class in the project with methods that have to be utilized in the project if comes to our project our UML Will utilizable in this way The third and post import for the project in system design is Database design where we endeavor to design database predicated on the number of modules in our project.

4.4 Implementation

The Implementation is Phase where we endeavor to give the practical output of the work done in designing stage and most of Coding in Business logic lay coms into action in this stage its main and crucial part of the project.

4.5 Testing

4.5.1 Unit Testing

It is done by the developer itself in every stage of the project and fine-tuning the bug and module predicated additionally done by the developer only here we are going to solve all the runtime errors.

4.5.2 Manual Testing

As our Project is academic Leave we can do any automatic testing so we follow manual testing by endeavor and error methods.

4.6 Deployment of System

Once the project is total yare we will come to deployment of client system in genuinely world as its academic leave we did deployment i our college lab only with all need Software's with having Windows OS.

4.7 Maintenance

The Maintenance of our Project is one time process only.

4.8 Functional Requirements

- **Processor**: Intel i5 or higher.
- **Speed:** 2.1 Ghz or higher
- **RAM:** 8 GB or higher
- **Developer tools:** Anaconda Navigator,Command Prompt, Python 3.6

4.9 Application needs Non-Functional Requisites

- **Expanded System admin security**

  Overseer to eschew the abuse of the application by PC ought to be exceptionally secured and available.
- **Compactness**

  The Presentation of this application is facile to utilize so it is looks simple for the using client to comprehend and react to identically tantamount.
- **Unwavering quality**

  The functionalities accessible in the application this substructure has high probability to convey us the required inquiries.
- **Time take for Reaction**

  The time taken by the application to culminate an undertaking given by the client is very fast.
- **Multifariousness**

  Our application can be stretched out to incorporate the vicissitudes done by applications present now to enhance the performance of the item. This is implicatively insinuated for the future works that will be done on the application.
- **Vigor**

  The project is blame tolerant concerning illicit client/beneficiary sources of info. Blunder checking has been worked in the platforms to avert platforms disappointment.

4.10 Modules

4.10.1 Package Extraction Module:

This module involves the installing of different packages in both anaconda prompt and command prompt for training and testing modules respectively. Each package serve different purpose and these packages have to be installed in both anaconda prompt using conda install and in command prompt using pip install.Below are the list of packages to install.

- LibROSA is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.

- NumPy is the fundamental package for scientific computing with Python. It contains a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and Fortran code and useful linear algebra, Fourier transform, and random number capabilities

- TensorFlow is a Python library for high-performance numerical calculations that allows users to create sophisticated deep learning and machine learning applications.

- Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.

- Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc.

4.10.2 Training Module:

Pre-Processing For Emotion Recognition

Prior to feature extraction, some necessary steps are taken to manipulate speech signal. Pre-processing mainly includes sampling, normalization and segmentation. Speech signal is analog in form and it needs to be converted into digital form for processing. Simple flag is changed over into discrete time motion with the assistance of examining. Inspecting guarantees that the first attributes of the flag are held. As indicated by examining hypothesis, when the testing recurrence is more prominent than or equivalent to double the greatest simple flag recurrence, the discrete time flag can remake the first simple flag.

Volume is a vital reality while computing discourse vitality and different highlights. Standardization process utilizes the flag succession to guarantee each sentence has a tantamount volume level. Discourse is an arbitrary flag and its attributes change with time, yet this change isn't immediate. Division process separates the flag succession into various edges with cover. Covering is done to keep away from loss of information due to associating. The signal s(n) becomes si (n) once framed, where i indicates the number of frames. After pre-processing, characteristics of the whole speech signal can be studied from statistical values.

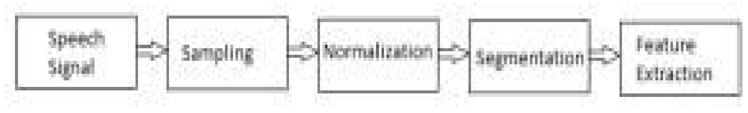


Figure 4.2 Preprocessing for emotion recognition

Features For Emotion Classification and Recognition

Energy

Energy is the most basic feature in speech signal processing. It plays a vital role in emotion recognition, e.g. speech signals corresponding to happiness and anger have much higher energy than those of sadness.

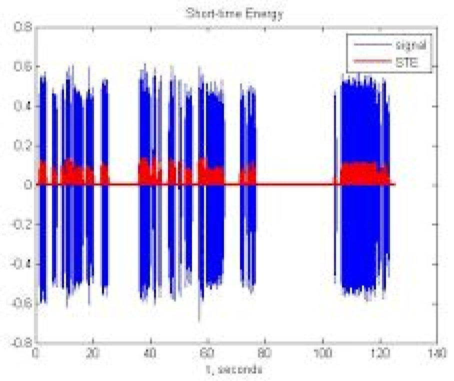


Figure 4.3 Short term energy

Pitch

Pitch is known as the perceived rising and falling of voice tone. It is the perceptual form of fundamental frequency because it sets the periodic baseline for all higher frequency harmonics contributed by oral resonance cavities. It represents the vibration frequency of vocal folds during speaking. There are many ways to estimate pitch from a speech signal. Autocorrelation method is used because it is a commonly used method and is easy to practice. This method uses short term analysis technique to maintain characteristic for each frame, which means pre-processing should be fully applied before pitch extraction. Since autocorrelation can decide the period of a periodic signal, autocorrelation is applied for each frame.

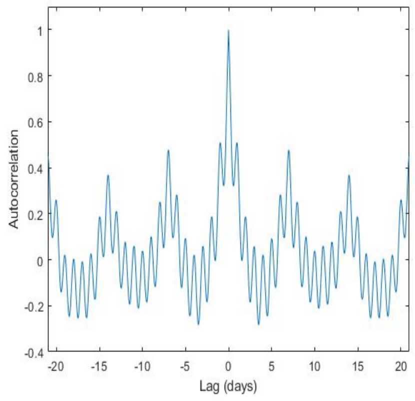


Figure 4.4 AutoCorrelation

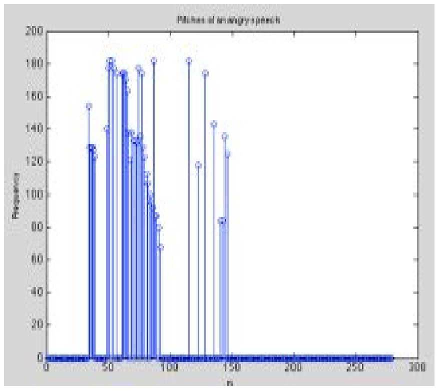


Figure 4.5 Pitch for angry Signal

Mel Frequency Cepstral Coefficients (MFCC)

The Mel-Frequency Cepstrum Coefficients (MFCC) is an accurate representation of short time power spectrum of a sound. The advantage of MFCC is that it imitates the reaction of human ear to sounds using a mel scale instead of linearly spaced frequency bands.

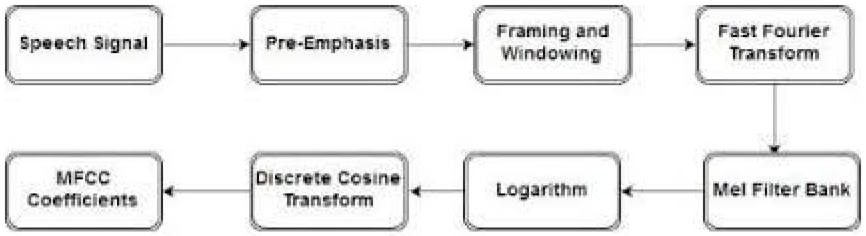


Figure 4.6 MFCC Block Diagram

Classification with Neural Network

The general working flow is shown in figure. Input data and target data are loaded. Input data here is a matrix of the features extracted from the speech inputs. Target data indicates the emotional states of these inputs.

Next, the percentage of input data into 3 categories namely training, validation and testing is chosen randomly.

The training set fits the parameters of the classifier i.e. finds the optimal weights for each feature. Validation set tunes the parameters of a classifier that is it determines a stop point for training set.

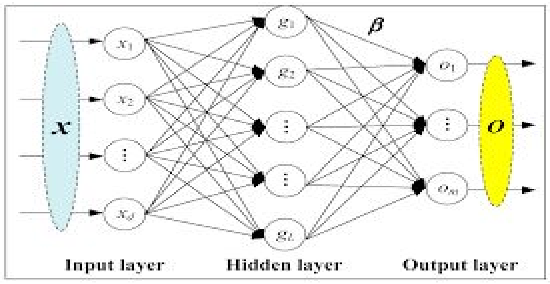


Figure 4.7 Artificial Neural Network

Finally test set tests the final model and estimates the error rate. The default value sets training in 70 percent and 15 percent each for the rest. Initially the default values are used. Next, the number of hidden layers is chosen. As discussed previously, more the number of hidden layers, more complicated the system, better the result. Lastly the network is trained several times. The mean square together with error rate will indicate how good the results are.

4.10.3 Testing Module:

The data owner (Owner) defines the access policy about who can get access to each file, and encrypts the file under the defined policy. First of all, each owner encrypts his/her data with a symmetric encryption algorithm. Then, the owner formulates access policy over an attribute set and encrypts the symmetric key under the policy according to public keys obtained from CA. After that, the owner sends the whole encrypted data and the encrypted symmetric key (denoted as ciphertext CT) to the cloud server to be stored in the cloud.

4.10.4 Recording Module:

In this Module record.py is executed which enables a platform for the user to record the speech data through external sources or his own voice. The recorded voice can be further analysed to know the emotional state of the voice data and thus achieving the feature of real time emotion analysis by predicting the emotion.

4.11 Architecture:

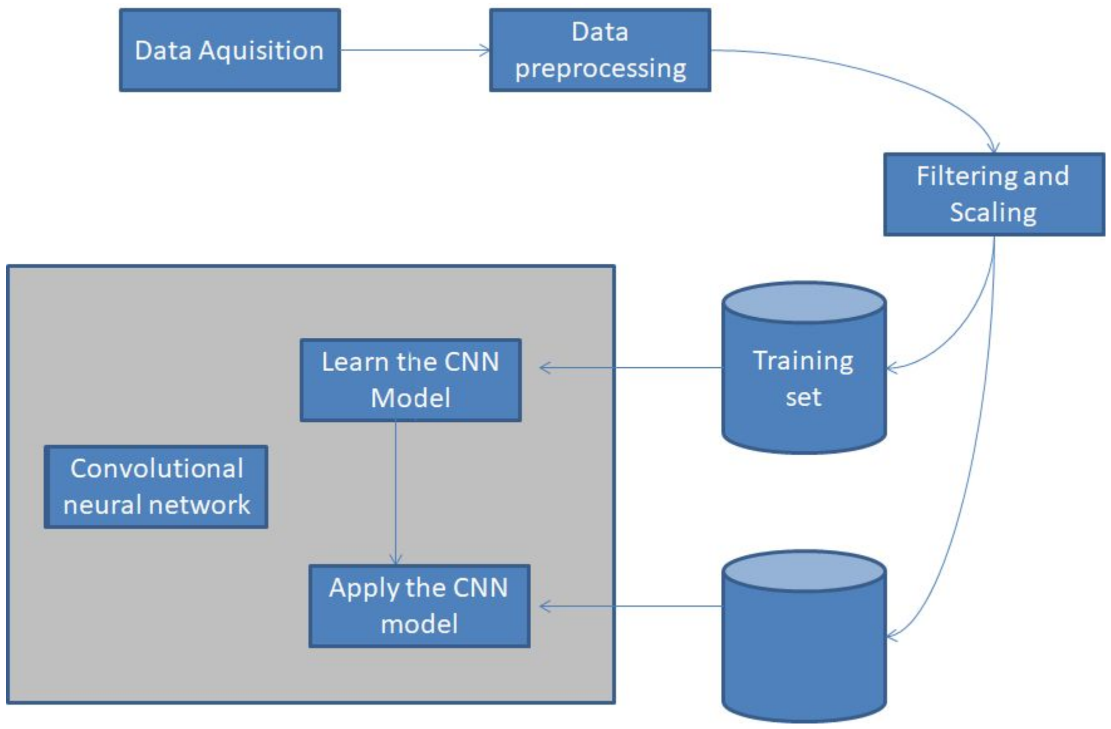


Figure 4.8 Architecture of Emotion Analysis through Voice

4.12 Algorithm and Concepts used in Project

4.12.1 MFCC (Mel Frequency Cepstral Coefficient)

The initial phase in any programmed discourse acknowledgment framework is to separate highlights for example distinguish the parts of the sound flag that are useful for recognizing the phonetic substance and disposing of the various stuff which conveys data like foundation commotion, feeling and so forth.

The central matter to comprehend about discourse is that the sounds created by a human are sifted by the state of the vocal tract including tongue, teeth and so on. This shape figures out what sound turns out. On the off chance that we can decide the shape precisely, this should give us an exact portrayal of the phoneme being delivered.

The state of the vocal tract shows itself in the envelope of the brief timeframe control range, and the activity of MFCCs is to precisely speak to this envelope. This page will give a short instructional exercise on MFCCs.

Mel Frequency Cepstral Coefficients (MFCCs) are an element broadly utilized in programmed discourse and speaker acknowledgment.

- Casing the flag into short edges.
- For each edge figure the periodogram gauge of the power range.
- Apply the mel filterbank to the power spectra, total the vitality in each channel.
- Take the logarithm of all filterbank energies.
- Take the DCT of the log filterbank energies.
- Keep DCT coefficients 2-13, dispose of the rest.

There are a couple of more things regularly done, once in a while the edge vitality is affixed to each component vector. Delta and Delta-Delta highlights are generally likewise annexed. Liftering is likewise regularly connected to the last highlights.

A sound flag is continually changing, so to improve things we expect that on brief time scales the sound flag doesn't change much (when we state it doesn't transform, we mean factually for example measurably stationary, clearly the examples are always showing signs of change on even brief time scales). This is the reason we outline the flag into 20-40ms casings. On the off chance that the edge is a lot shorter we don't have enough examples to get a solid ghostly gauge, in the event that it is longer the flag changes a lot all through the edge.

The subsequent stage is to figure the power range of each edge. This is roused by the human cochlea (an organ in the ear) which vibrates at various spots relying upon the recurrence of the approaching sounds. Contingent upon the area in the cochlea that vibrates (which wobbles little hairs), distinctive nerves fire advising the cerebrum that specific frequencies are available.Our periodogram gauge plays out a comparable employment for us, distinguishing which frequencies are available in the edge.

The periodogram phantom gauge still contains a great deal of data not required for Automatic Speech Recognition (ASR). Specifically the cochlea can not recognize the contrast between two firmly separated frequencies. This impact turns out to be progressively articulated as the frequencies increment. Hence we take bunches of periodogram canisters and total them up to get a thought of how much vitality exists in different recurrence locales. This is performed by our Mel filterbank: the main channel is thin and gives a sign of how much vitality exists close to 0 Hertz. As the frequencies get higher our channels get more extensive as we become less worried about varieties. We are just keen on generally how much vitality happens at each spot. The Mel scale lets us know precisely how to space our filterbanks and how wide to make them. See underneath for how to figure the separating.

When we have the filterbank energies, we take the logarithm of them. This is likewise spurred by human hearing: we don't hear uproar on a direct scale. For the most part to twofold the percieved volume of a sound we have to put 8 fold the amount of vitality into it. This implies vast varieties in vitality may not sound all that extraordinary if the sound is noisy in the first place. This pressure activity makes our highlights coordinate all the more intently what people really hear. Why the logarithm and not a shape root? The logarithm enables us to utilize cepstral mean subtraction, which is a channel standardization method.

The last advance is to process the DCT of the log filterbank energies. There are 2 fundamental reasons this is performed. Since our filterbanks are for the most part covering, the filterbank energies are very connected with one another. The DCT decorrelates the energies which implies inclining covariance frameworks can be utilized to demonstrate the highlights in for example a HMM classifier. In any case, see that just 12 of the 26 DCT coefficients are kept. This is on the grounds that the higher DCT coefficients speak to quick changes in the filterbank energies and incidentally, these quick changes really corrupt ASR execution, so we get a little improvement by dropping them.

4.12.2 Convolutional neural network

CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

Convolutional Neural Networks (CNN) is one of the variants of neural networks used heavily in the field of Computer Vision. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers. Here it simply means that instead of using the normal activation functions defined above, convolution and pooling functions are used as activation functions.

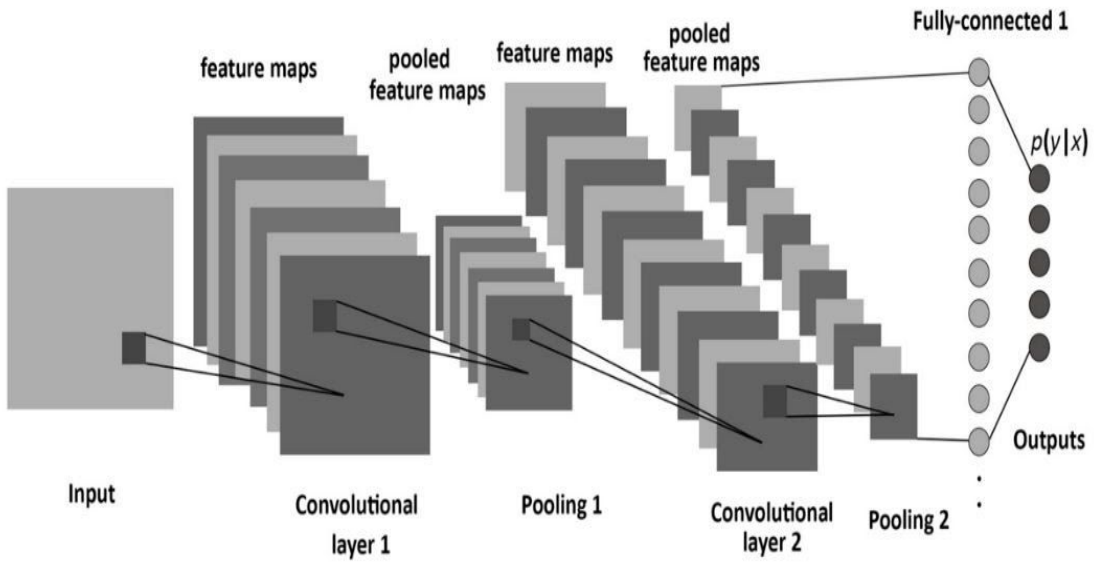


Figure 4.9 Architecture of CNN

A sound flag is continually changing, so to improve things we expect that on brief time scales the sound flag doesn't change much (when we state it doesn't transform, we mean factually for example measurably stationary, clearly the examples are always showing signs of change on even brief time scales). This is the reason we outline the flag into 20-40ms casings. On the off chance that the edge is a lot shorter we don't have enough examples to get a solid ghostly gauge, in the event that it is longer the flag changes a lot all through the edge.

The subsequent stage is to figure the power range of each edge. This is roused by the human cochlea (an organ in the ear) which vibrates at various spots relying upon the recurrence of the approaching sounds. Contingent upon the area in the cochlea that vibrates (which wobbles little hairs), distinctive nerves fire advising the cerebrum that specific frequencies are available. Our periodogram gauge plays out a comparable employment for us, distinguishing which frequencies are available in the edge.

The periodogram phantom gauge still contains a great deal of data not required for Automatic Speech Recognition (ASR). Specifically the cochlea can not recognize the contrast between two firmly separated frequencies. This impact turns out to be progressively articulated as the frequencies increment. Hence we take bunches of periodogram canisters and total them up to get a thought of how much vitality exists in different recurrence locales. This is performed by our Mel filterbank: the main channel is thin and gives a sign of how much vitality exists close to 0 Hertz. As the frequencies get higher our channels get more extensive as we become less worried about varieties. We are just keen on generally how much vitality happens at each spot The Mel scale lets us know precisely how to space our filterbanks and how wide to make them. See underneath for how to figure the separating.

When we have the filterbank energies, we take the logarithm of them. This is likewise spurred by human hearing: we don't hear uproar on a direct scale. For the most part to twofold the perceived volume of a sound we have to put 8 fold the amount of vitality into it. This implies vast varieties in vitality may not sound all that extraordinary if the sound is noisy in the first place. This pressure activity makes our highlights coordinate all the more intently what people really hear. Why the logarithm and not a shape root. The logarithm enables us to utilize cepstral mean subtraction, which is a channel standardization method.

The last advance is to process the DCT of the log filterbank energies. There are 2 fundamental reasons this is performed. Since our filterbanks are for the most part covering, the filterbank energies are very connected with one another. The DCT decorrelates the energies which implies inclining covariance frameworks can be utilized to demonstrate the highlights in for example a HMM classifier. In any case, see that just 12 of the 26 DCT coefficients are kept. This is on the grounds that the higher DCT coefficients speak to quick changes in the filterbank energies and incidentally, these quick changes really corrupt ASR execution, so we get a little improvement by dropping them.

Convolution neural network algorithm is a multilayer perceptron that is the special design for identification of two-dimensional image information . Always has more layers: input layer, convolution layer, sample layer and output layer. In addition, in a deep network architecture the convolution layer and sample layer can have multiple.

CNN is not as restricted boltzmann machine, need to be before and after the layer of neurons in the adjacent layer for all connections, convolution neural network algorithms, each neuron don't need to do feel global image, just feel the local area of the image. In addition, each neuron parameter is set to the same, namely, the sharing of weights , namely each neuron with the same convolution kernels to deconvolution image.

CHAPTER-5
SYSTEM DESIGN

5. SYSTEM DESIGN

5.1 UML Diagrams Introduction:

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML can be described as a general purpose visual modeling language to visualize, specify, construct and document software system.

Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non software systems as well like process flow in a manufacturing unit etc…,

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. The goal of UML can be defined as a simple modeling mechanism to model all possible practical systems in today's complex environment.

5.2 Use Case Diagram:

5.2.1 Definition:

A **use case diagram** is a representation of a user's interaction with the system, that shows the relationship between the user and the different use cases, in which the user is involved.

A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

An utilization case chart is a dynamic or conduct graph in UML. Use case outlines show the usefulness of a framework utilizing performing artists and use cases. Use cases are a lot of activities, administrations, and capacities that the framework needs to perform. In this unique situation, a framework is something being created or worked, for example, a site. The on-screen characters are individuals or substances working under characterized jobs inside the framework.

Use case graphs are profitable for envisioning the practical necessities of a framework that will convert into plan decisions and advancement needs.

They additionally help distinguish any inside or outside elements that may impact the framework and ought to be mulled over.

They give a decent abnormal state examination from outside the framework. Use case charts indicate how the framework associates with on-screen characters without stressing over the subtleties of how that usefulness is executed.

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either uses or extends.

A uses relationship indicates that one use case is needed by another in order to perform a task. An extends relationship indicates alternative options under a certain use case.
In brief, the purposes of use case diagrams can be said to be as follows:

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

When we are planning to draw a use case diagram, we should have the following items identified.

- Functionalities to be represented as use case
- Actors
- Relationships among the use cases and actors.

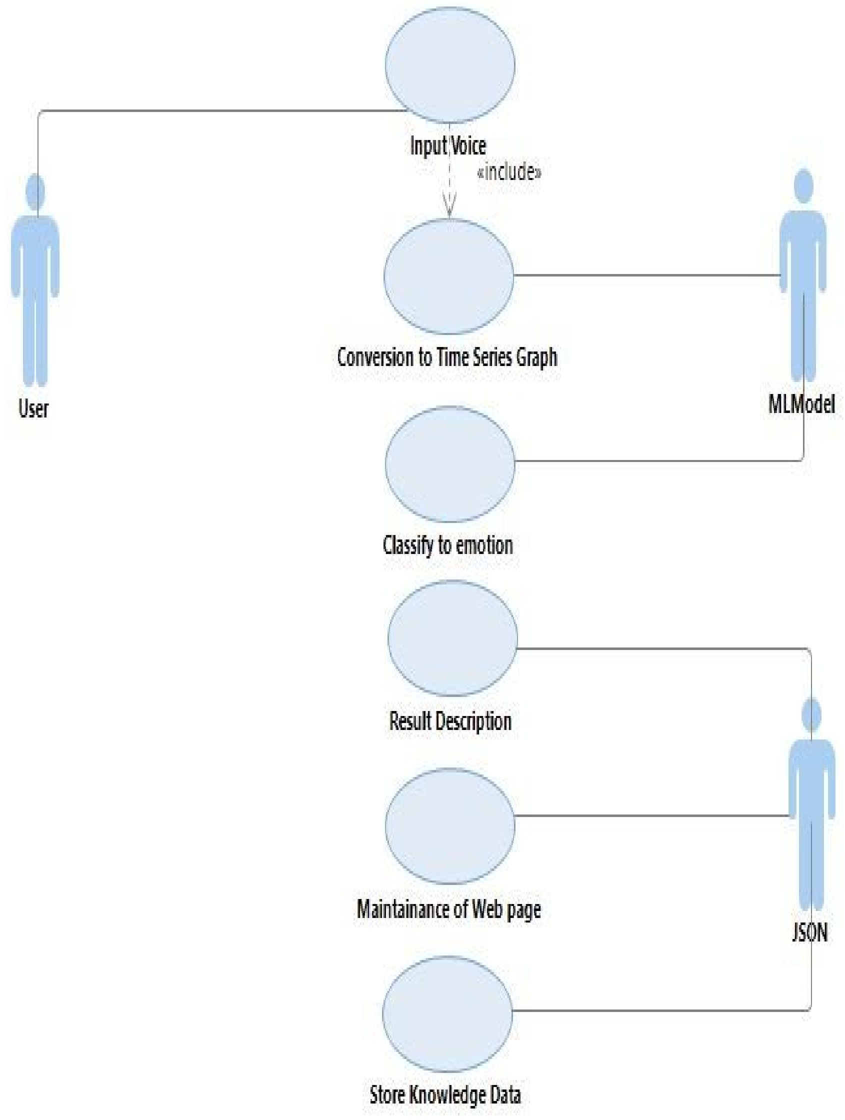


Figure 5.1 Use Case Diagram

5.2.2 Description:

The actors involved in this system are user, CNN model and JSON. The respective actions performed by the actors are represented in the form of use cases with which the actors are associated.

A usage case outline inside the Unified Modeling Language we used in our Project Development is Star (UML) could be a sort of behavioral chart portrayed out by and produced using a Use-case examination.

Its inspiration is to gift a graphical layout of the presence of mind gave by a system to the extent performing specialists, their targets (addressed as use cases), and any conditions between those use cases.

The most explanation behind a use case diagram is to show what structure limits are played out that on-screen character. Parts of the entertainers inside the system will be diagram.

5.3 Activity Diagram:

5.3.1 Definition:

Activity diagrams

These are graphical portrayals of work processes of stepwise exercises and activities with help for decision, cycle and simultaneousness.

In the Unified Modeling Language, movement graphs are planned to display both computational and authoritative procedures. Movement graphs demonstrate the general stream of control.

Activity diagram is an important diagram in UML to describe the dynamic aspects of the system. It is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

The purpose of an activity diagram can be described as follows:

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

Activity diagrams are mainly used as a flowchart that consists of activities performed by the system. Activity diagrams are not exactly flowcharts as they have some additional capabilities. These additional capabilities include branching, parallel flow, swimlane, etc.

Before drawing an activity diagram, we should identify the following elements:

- Activities
- Association
- Conditions
- Constraints

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

5.3.2 Description:

Movement graph is fundamentally a flowchart to speak to the stream starting with one action then onto the next action. The action can be portrayed as a task of the framework. The control stream is attracted starting with one task then onto the next. This stream can be consecutive, stretched, or simultaneous. Action graphs manage all kind of stream control by utilizing distinctive components, for example, fork, join, and so on

Activity outlines are graphical delineations of work systems of stepwise exercises and activities with assistance for choice, cycle and synchronization. Inside the Unified Modeling Language we utilized in our Project Development is Star , advancements designs will be standard portray the business and operational in little stages work methodology of parts amidst a structure. Relate in nursing movement chart demonstrates the surge of association.

In movement graph depicts the control stream from a begin point to a completion point appearing different choice ways that exist while the action is being executed.

We can delineate both successive handling and simultaneous preparing of exercises utilizing a movement chart. They are utilized in business and procedure displaying where their essential use is to portray the dynamic parts of a framework. An action graph is fundamentally the same as a flowchart.

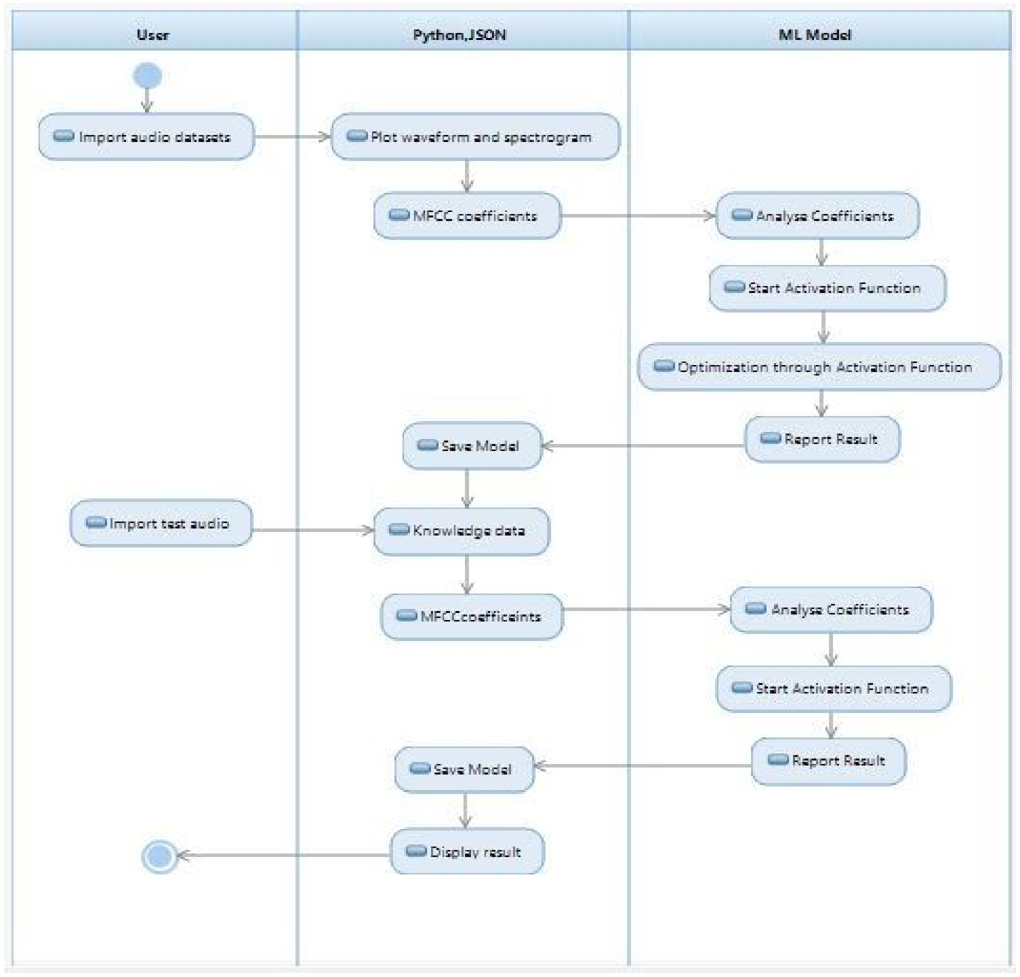


Figure 5.2 Activity Diagram

5.4 Sequence Diagram

5.4.1 Definition:

A Sequence diagram is a communication graph that shows how forms work with each other and the request in which the moves make it. The Sequence diagram depicts the time sequence among various objects in an application.

It depicts the sequence of messages with which objects communicate with each other so that they carry out the required functionality.

It consists of the lifelines which are usually parallel vertical lines. It consists of horizontal arrows which indicates the direction of the messages that are exchanged in a proper order which makes the user easy to understand.

The lifeline for a given object represents a role. The synchronous calls are represented with the help of a solid arrow head whereas the asynchronous messages are represented with the help of open arrowheads.

All objects are represented according to their time ordering. Timing of messages plays a major role in sequence diagrams. An object is killed immediately after its use in sequence diagrams.

1. **Common Properties:** An arrangement graph is much the same as unique sort of diagram and offers some indistinguishable properties from other diagrams. In any case, it varies from every single other diagram in its content.

2. **Contents Objects:** They are normally named or unknown instances of class, however may likewise speak to occurrences of different things, for example components, collaboration and nodes. Graphically, object is represented as a rectangle by underlying its name.

3. **Links:** A link is a semantic association among objects i.e., an object of an affiliation is called as a connection. It is represented as a line.

4. **Messages:** A message is a determination of a correspondence between objects that passes on the data with the desire that the action will follow.

There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

Object:

An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance. The object icon is similar to a class icon except that the name is underlined: An object's concurrency is defined by the concurrency of its class.

Message:

A message is the communication carried between two objects that trigger an event. A message carries information from the source focus of control to the destination focus of control. The synchronization of a message can be modified through the message specification. Synchronization means a message where the sending object pauses to wait for results.

Link:

A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes.

The existence of a relationship between two classes symbolizes a path of communication between instances of the classes: one object may send messages to another. The link is depicted as a straight line between objects or objects and class instances in a collaboration diagram. If an object links to itself, use the loop version of the icon.

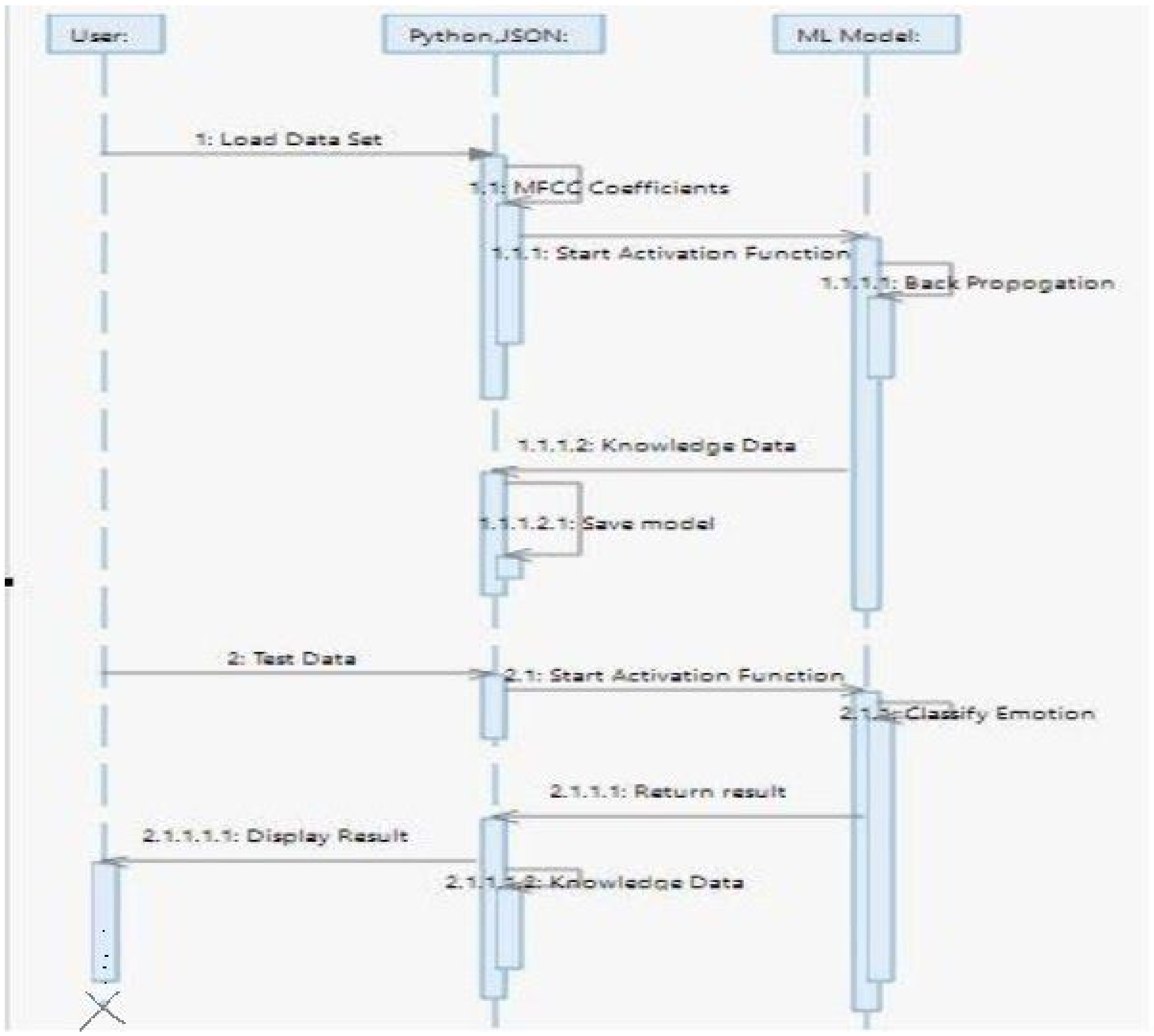


Figure 5.3 Sequence Diagram

5.4.2 Description:

A gathering diagram in Unified Modeling Language we used in our Project Development is Star (UML) could be a sensibly association graph that shows however frames work with each other and in what mastermind. It’s a create of a Message Sequence Chart. Progression diagrams are regularly known as event plots, event conditions, and short lived approach outlines.

5.5 Class Diagram

5.5.1 Definition:

Class diagrams are the main building blocks of every object oriented methods. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in its context. It describes various kinds of objects and the static relationship in between them.

The main purpose to use class diagrams are:

- This is the only UML which can appropriately depict various aspects of OOPs concept.
- Proper design and analysis of application can be faster and efficient.
- It is base for deployment and component diagram.

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

The purpose of the class diagram can be summarized as −

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

In a class diagram, each object is modelled as a class. Each class consists of section or compartments:

1. Class name
2. Attributes of a class or operations
3. Methods or functions
4. Documentation (optional section)

There are 4 approaches for identifying classes:

1. Noun phrase approach:
2. Common class pattern approach.
3. Use case Driven Sequence or Collaboration approach.
4. Classes, Responsibilities and collaborators Approach.

1. **Noun Phrase Approach:**

The guidelines for identifying the classes:

a. Look for nouns and noun phrases in the use-cases.
b. Some classes are implicit or taken from general knowledge.
c. All classes must make sense in the application domain; Avoid computer implementation classes – defer them to the design stage.
d. Carefully choose and define the class names After identifying the classes we have to eliminate the following types of classes: Adjective classes.

2. **Common class pattern approach:**

The following are the patterns for finding the candidate classes:

a. Concept class.
b. Events class.
c. Organization class
d. Peoples class
e. Places class
f. Tangible things and devices class.

3. **Use case driven approach:**

We have to draw the sequence diagram or collaboration diagram. If there is need for some classes to represent some functionality then add new classes which perform those functionalities.

4. **CRC approach:**

The process consists of the following steps:

a. Identify classes’ responsibilities (and identify the classes)
b. Assign the responsibilities
c. Identify the collaborators.

The following points ought to be recollected while drawing a class diagram:

A. The name of the class diagram must be meaningful to portray the aspect of the framework.
B. Each component and their connections must be distinguished ahead of time.
C. Each class has a responsibility (attributes and methods) that must be identified clearly.
D. Number of properties for each class must be minimum. Since pointless properties will make the diagram convoluted.
E. At whatever point required to depict some part of the diagram use notes Since toward the finish of the diagram it must be justifiable to the designer/coder.
F. Before finalizing the last version, the diagram must be drawn on plain paper and revise whatever number circumstances as would be prudent to make it redress.

1. **Scopes:**

The UML diagrams have two different types of scopes for class members:

i. instance members scope

ii. ii. classifier members scope

2. **Classifier members** are "static" members of a class in many programming languages. The scope is the class itself.

i. Static attributes are common to all other objects that invoke the class.

ii. Static methods are not instantiated.

3. **Instance members** are nothing but the members that are local to an object.

i. The main purpose of instance members is to allow the objects to store their states.

ii. Declarations outside the methods are usually known as instance members.

In software engineering, a class diagram in the Unified Modeling Language is a sort of static structure chart that portrays the structure of a framework by demonstrating the framework's classes, their qualities, tasks, and the connections among articles.

Class diagram is used for visualizing, describing, and documenting different aspects of a system. It gives an overview of the functions and modules that are being used in the system.

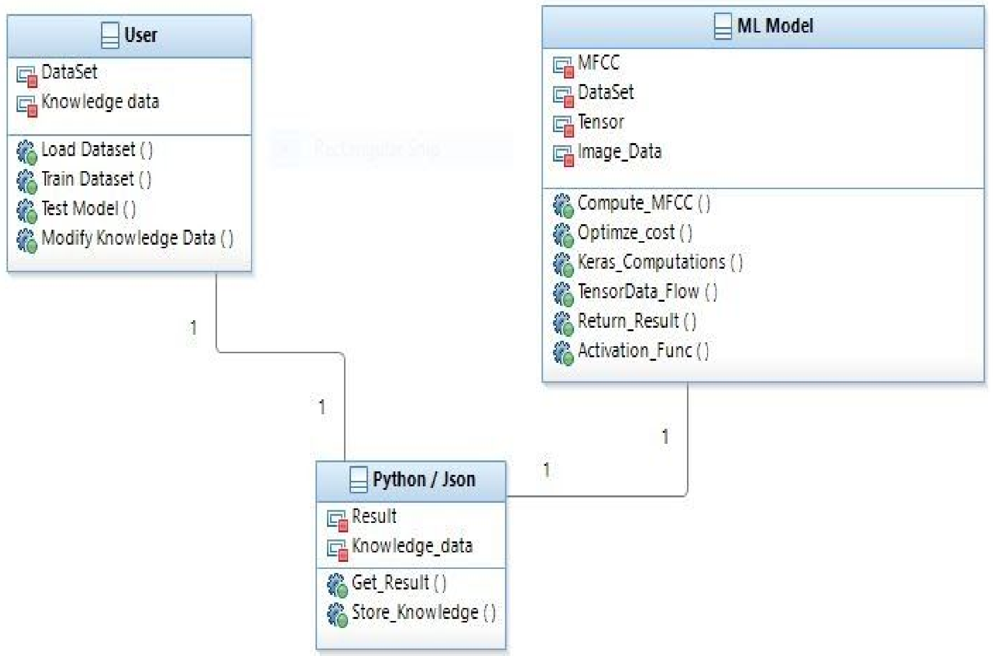


Figure 5.4 Class Diagram

5.5.2 Description:

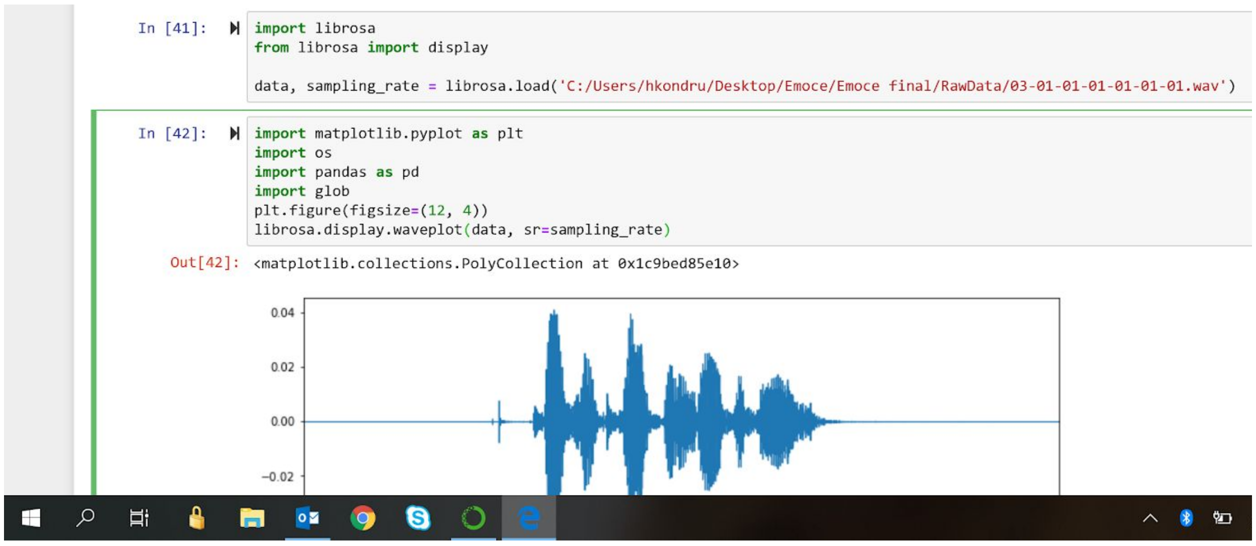
The figure Above shows the various classes involved along with their relationships and functionality. In PC code planning, a class plot inside the Unified Modeling Language we used in our Project Development is Star (UML) could be a sort of static structure outline that delineates the structure of a system by showing the system's characterizations, their qualities, operations (or methodologies), and moreover the associations among the groupings. It elucidates that class contains data.

CHAPTER-6

IMPLEMENTATION

6. IMPLEMENTATION

6.1 Importing required packages and plotting waveform

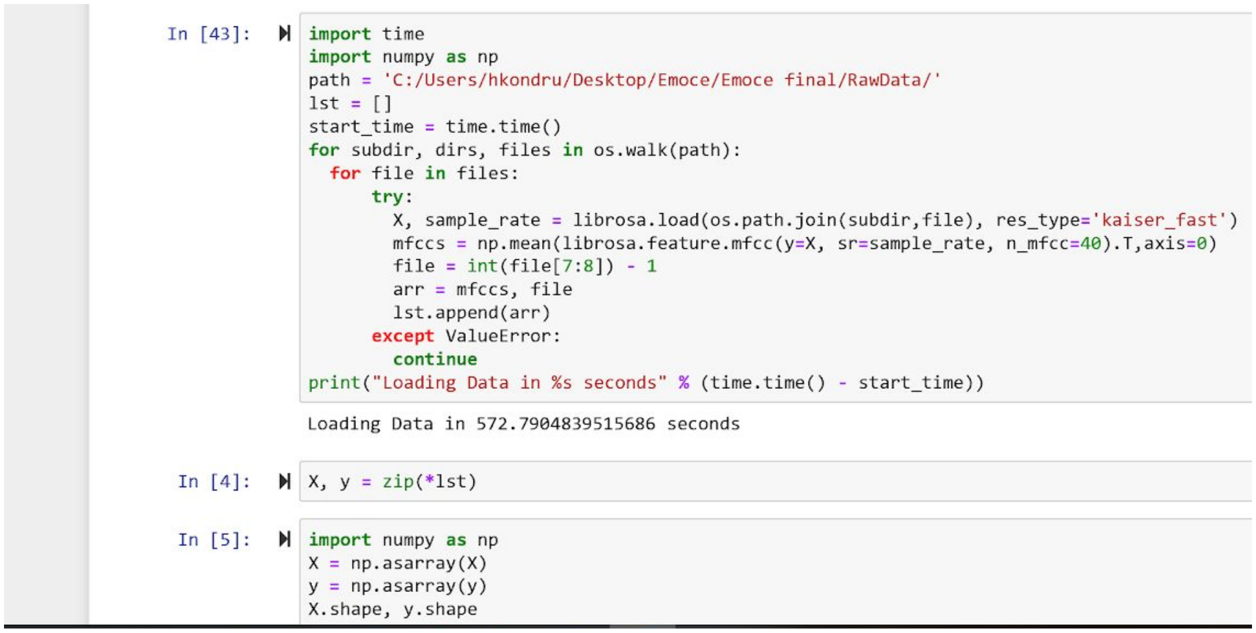


```
In [41]:  import librosa
          from librosa import display

          data, sampling_rate = librosa.load('C:/Users/hkondru/Desktop/Emoce/Emoce final/RawData/03-01-01-01-01-01-01.wav')
```

```
In [42]:  import matplotlib.pyplot as plt
          import os
          import pandas as pd
          import glob
          plt.figure(figsize=(12, 4))
          librosa.display.waveplot(data, sr=sampling_rate)

Out[42]:  <matplotlib.collections.PolyCollection at 0x1c9bed85e10>
```

Figure 6.1 Importing required packages and plotting waveform

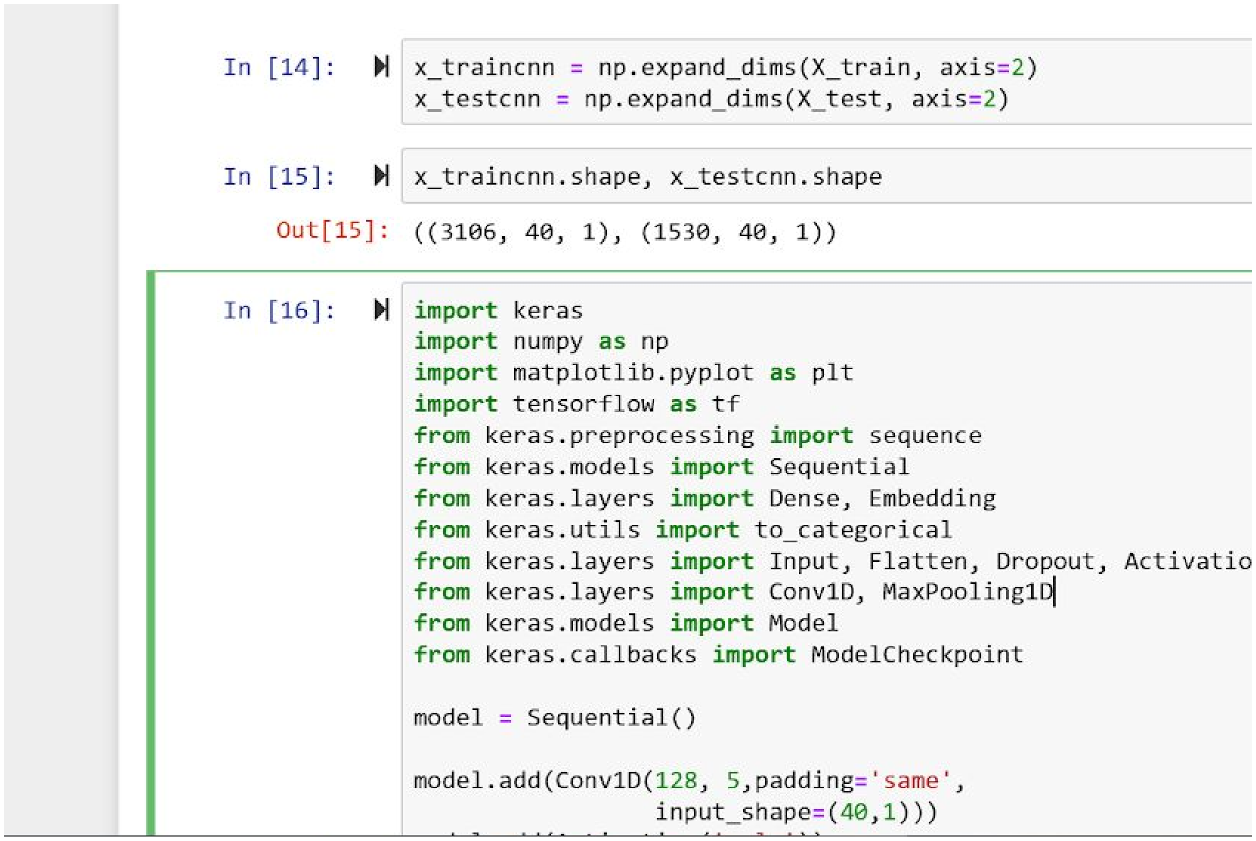
6.2 Loading all data files and obtaining mfcc coefficients



```
In [43]:  import time
          import numpy as np
          path = 'C:/Users/hkondru/Desktop/Emoce/Emoce final/RawData/'
          lst = []
          start_time = time.time()
          for subdir, dirs, files in os.walk(path):
            for file in files:
                try:
                  X, sample_rate = librosa.load(os.path.join(subdir,file), res_type='kaiser_fast')
                  mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T,axis=0)
                  file = int(file[7:8]) - 1
                  arr = mfccs, file
                  lst.append(arr)
                except ValueError:
                  continue
          print("Loading Data in %s seconds" % (time.time() - start_time))

          Loading Data in 572.7904839515686 seconds

In [4]:   X, y = zip(*lst)

In [5]:   import numpy as np
          X = np.asarray(X)
          y = np.asarray(y)
          X.shape, y.shape
```

Figure 6.2 Loading all data files and obtaining mfcc coefficients

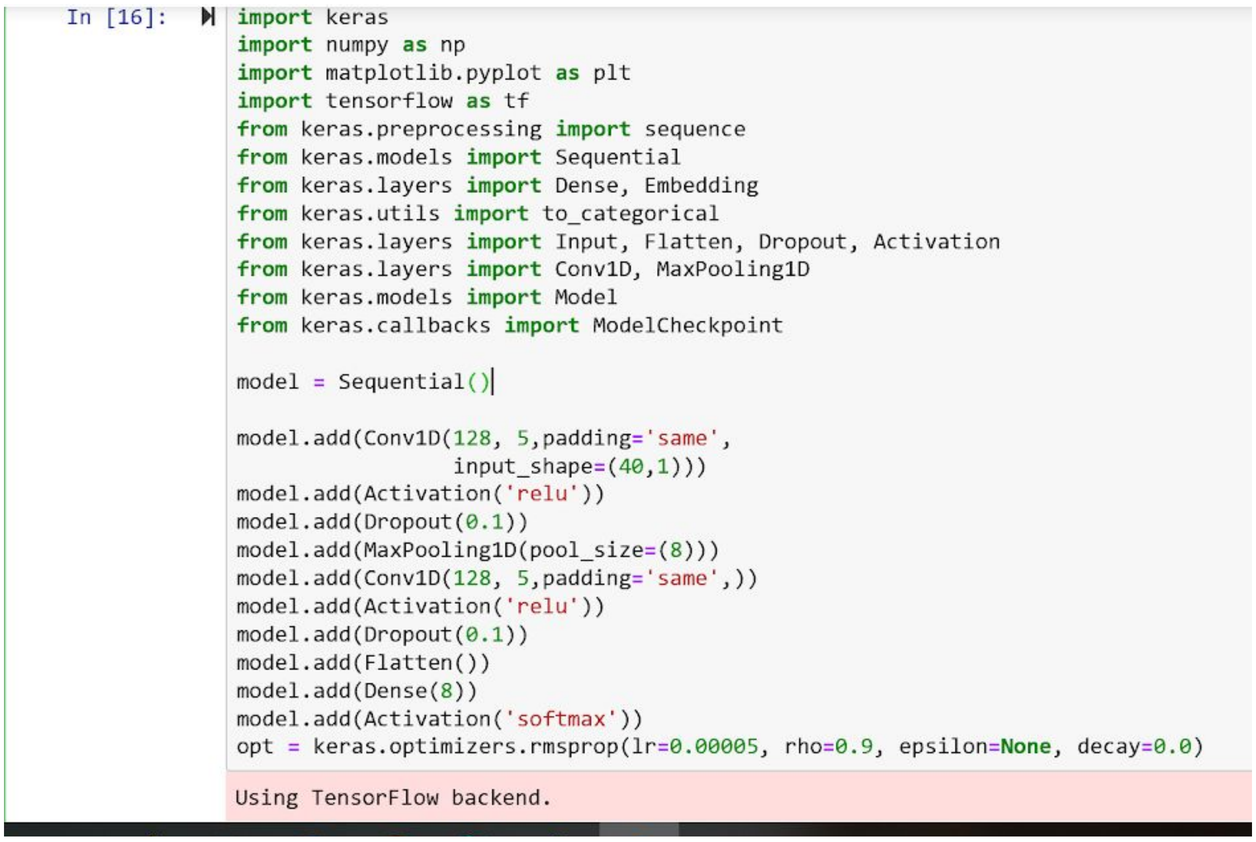
6.3 Building CNN Model



```
In [14]:  x_traincnn = np.expand_dims(X_train, axis=2)
           x_testcnn = np.expand_dims(X_test, axis=2)

In [15]:  x_traincnn.shape, x_testcnn.shape

Out[15]:  ((3106, 40, 1), (1530, 40, 1))

In [16]:  import keras
           import numpy as np
           import matplotlib.pyplot as plt
           import tensorflow as tf
           from keras.preprocessing import sequence
           from keras.models import Sequential
           from keras.layers import Dense, Embedding
           from keras.utils import to_categorical
           from keras.layers import Input, Flatten, Dropout, Activatio
           from keras.layers import Conv1D, MaxPooling1D
           from keras.models import Model
           from keras.callbacks import ModelCheckpoint

           model = Sequential()

           model.add(Conv1D(128, 5,padding='same',
                            input_shape=(40,1)))
```

Figure 6.3 Building CNN Model

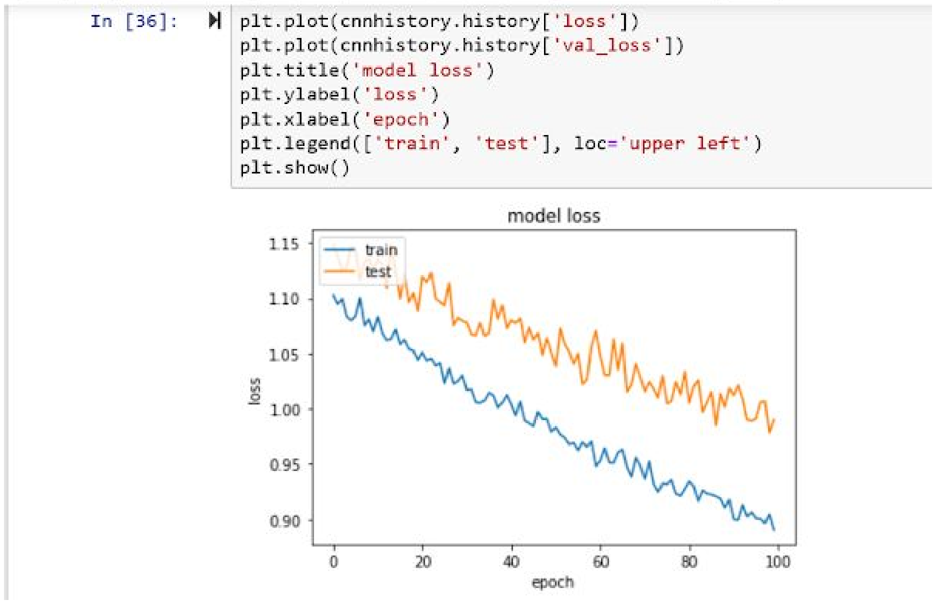
6.3.1 Importing required packages



```
In [16]:  import keras
          import numpy as np
          import matplotlib.pyplot as plt
          import tensorflow as tf
          from keras.preprocessing import sequence
          from keras.models import Sequential
          from keras.layers import Dense, Embedding
          from keras.utils import to_categorical
          from keras.layers import Input, Flatten, Dropout, Activation
          from keras.layers import Conv1D, MaxPooling1D
          from keras.models import Model
          from keras.callbacks import ModelCheckpoint

          model = Sequential()

          model.add(Conv1D(128, 5,padding='same',
                           input_shape=(40,1)))
          model.add(Activation('relu'))
          model.add(Dropout(0.1))
          model.add(MaxPooling1D(pool_size=(8)))
          model.add(Conv1D(128, 5,padding='same',))
          model.add(Activation('relu'))
          model.add(Dropout(0.1))
          model.add(Flatten())
          model.add(Dense(8))
          model.add(Activation('softmax'))
          opt = keras.optimizers.rmsprop(lr=0.00005, rho=0.9, epsilon=None, decay=0.0)

          Using TensorFlow backend.
```

Figure 6.4 Importing required packages

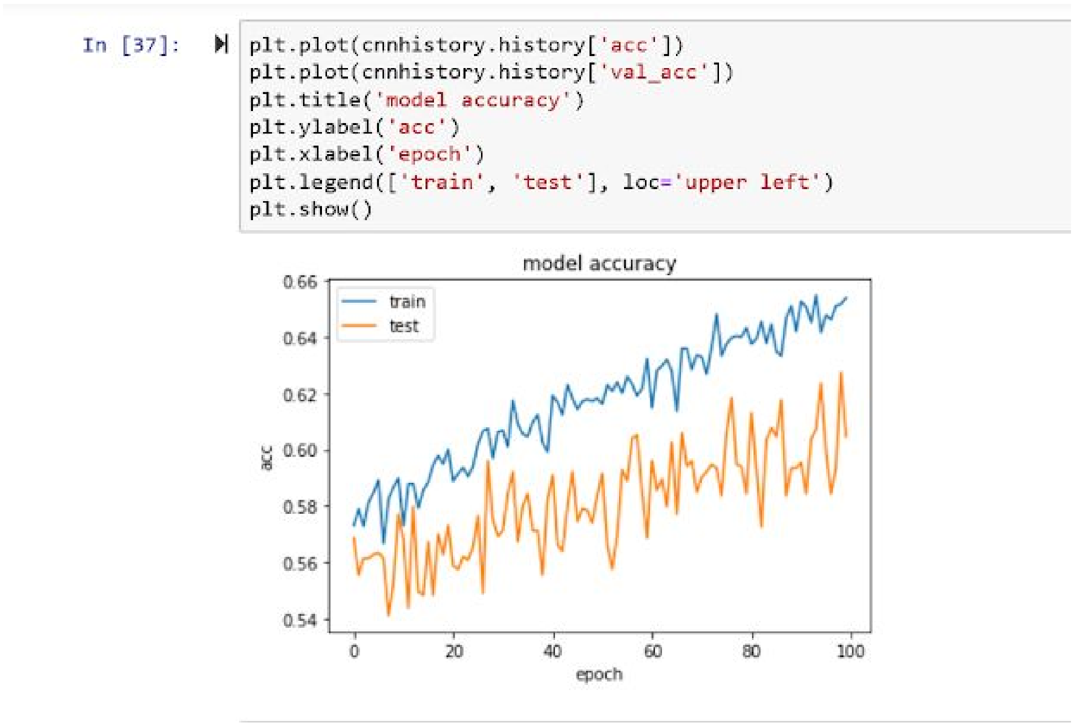
6.4 Illustrating Model loss compared to training vs testing



```
In [36]:  plt.plot(cnnhistory.history['loss'])
          plt.plot(cnnhistory.history['val_loss'])
          plt.title('model loss')
          plt.ylabel('loss')
          plt.xlabel('epoch')
          plt.legend(['train', 'test'], loc='upper left')
          plt.show()
```

Figure 6.5 Model loss

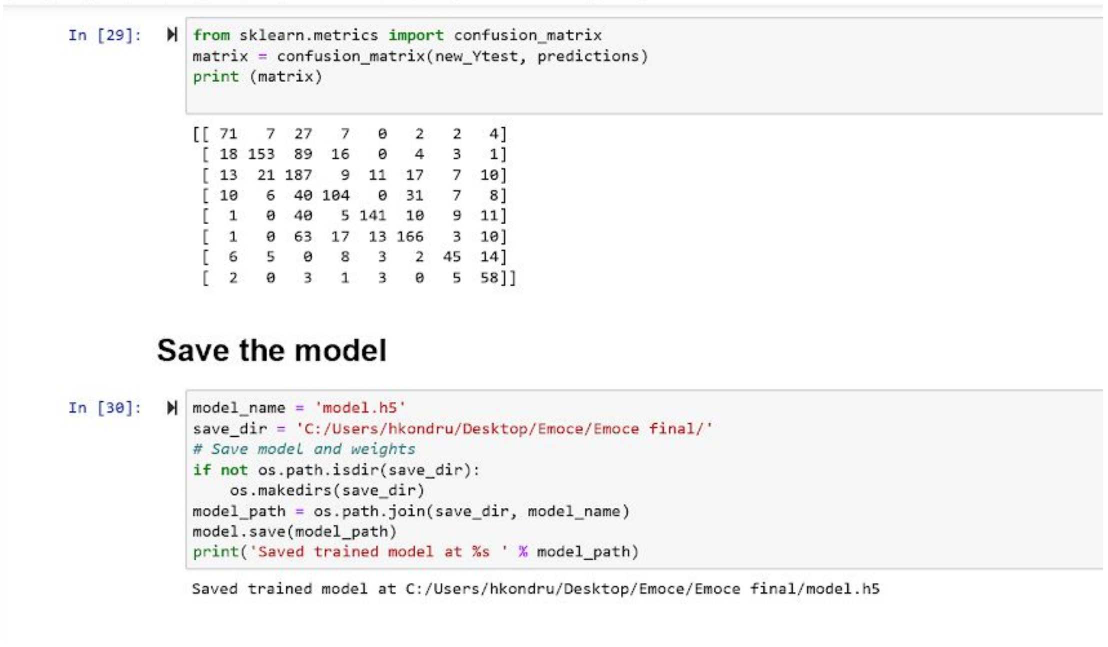
6.5 Model Accuracy compared to training vs testing



```
In [37]:  plt.plot(cnnhistory.history['acc'])
          plt.plot(cnnhistory.history['val_acc'])
          plt.title('model accuracy')
          plt.ylabel('acc')
          plt.xlabel('epoch')
          plt.legend(['train', 'test'], loc='upper left')
          plt.show()
```

Figure 6.6 Model Accuracy

6.6 Saving the Model



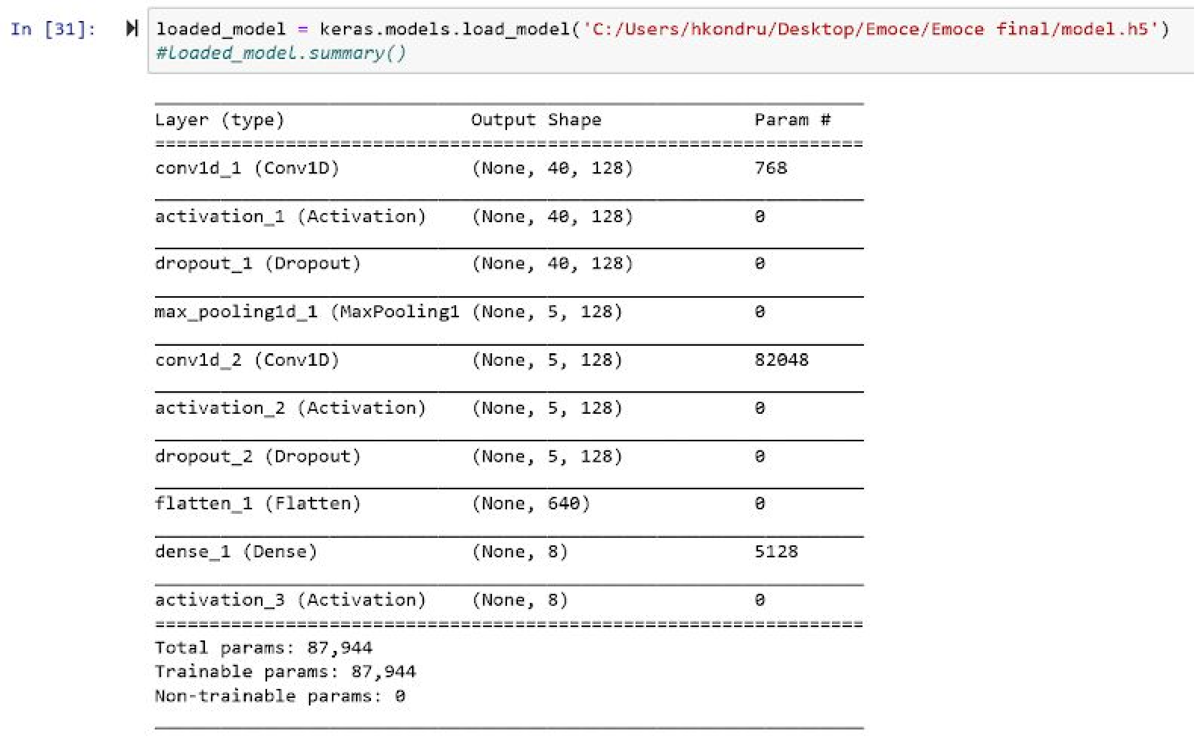
```
In [29]:  from sklearn.metrics import confusion_matrix
          matrix = confusion_matrix(new_Ytest, predictions)
          print (matrix)

[[ 71   7  27   7   0   2   2   4]
 [ 18 153  89  16   0   4   3   1]
 [ 13  21 187   9  11  17   7  10]
 [ 10   6  40 104   0  31   7   8]
 [  1   0  40   5 141  10   9  11]
 [  1   0  63  17  13 166   3  10]
 [  6   5   0   8   3   2  45  14]
 [  2   0   3   1   3   0   5  58]]
```

Save the model

```
In [30]:  model_name = 'model.h5'
          save_dir = 'C:/Users/hkondru/Desktop/Emoce/Emoce final/'
          # Save model and weights
          if not os.path.isdir(save_dir):
              os.makedirs(save_dir)
          model_path = os.path.join(save_dir, model_name)
          model.save(model_path)
          print('Saved trained model at %s ' % model_path)

Saved trained model at C:/Users/hkondru/Desktop/Emoce/Emoce final/model.h5
```

Figure 6.7 Saving Model

6.7 Reloading the model to test it

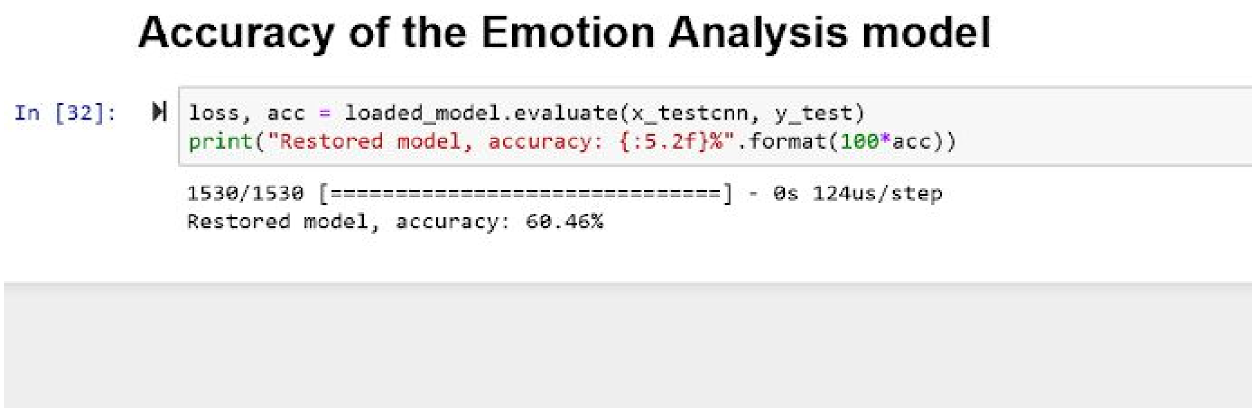


```
In [31]:  loaded_model = keras.models.load_model('C:/Users/hkondru/Desktop/Emoce/Emoce final/model.h5')
          #loaded_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 40, 128)           768
_________________________________________________________________
activation_1 (Activation)    (None, 40, 128)           0
_________________________________________________________________
dropout_1 (Dropout)          (None, 40, 128)           0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 5, 128)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 5, 128)            82048
_________________________________________________________________
activation_2 (Activation)    (None, 5, 128)            0
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 128)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 640)               0
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 5128
_________________________________________________________________
activation_3 (Activation)    (None, 8)                 0
=================================================================
Total params: 87,944
Trainable params: 87,944
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.8 Reloading the model to test it

6.8 Accuracy of the Emotion Analysis Model
Gi



Accuracy of the Emotion Analysis model

```
In [32]:  loss, acc = loaded_model.evaluate(x_testcnn, y_test)
          print("Restored model, accuracy: {:5.2f}%".format(100*acc))

1530/1530 [==============================] - 0s 124us/step
Restored model, accuracy: 60.46%
```

Figure 6.9 Accuracy of the Emotion Analysis Model

6.9 Code for predicting the emotion of given audio clip as input

➢ **Predict.py**

```
import keras

import numpy as np

import librosa

import PySimpleGUI as sg

class livePredictions:

    def __init__(self, path, file):

        self.path = path

        self.file = file

    def load_model(self):

        self.loaded_model = keras.models.load_model(self.path)

        return self.loaded_model.summary()

    def makepredictions(self):

        data, sampling_rate = librosa.load(self.file)

        mfccs = np.mean(librosa.feature.mfcc(y=data, sr=sampling_rate, n_mfcc=40).T, axis=0)

        x = np.expand_dims(mfccs, axis=2)
        x = np.expand_dims(x, axis=0)




 predictions = self.loaded_model.predict_classes(x)

        #print( "Prediction is", " ", self.convertclasstoemotion(predictions))

        return self.convertclasstoemotion(predictions)

    def convertclasstoemotion(self, pred):

        self.pred  = pred

        if pred == 0:

            pred = "neutral , The emotion experienced by the subject is neutral to emotional context
surrounding the subject"

            return pred

        elif pred == 1:

            pred = "calm , The subject is in a resonant state of clamness in the emotional context
surrounding the subject"

            return pred

        elif pred == 2:

            pred = "happy , The subject is cheerfull and experiencing a state of bliss in the emotional
context surrounding the user"

            return pred
        elif pred == 3:

     pred = "sad , The subject is quite gloomy given the situvation due to
disapointment/disatisfaction in the context surrounding the user"

           return pred

        elif pred == 4:

              pred = "angry , The subject is in a state of distress and frustration and is expected to act
on an  impluse due to the context surrounding the subject"

            return pred

        elif pred == 5:

               pred = "fearful ,The subject is experiencing a moment of clouded judgement and in a
inferior state of mind in the context surrounding the subject"

            return pred

        elif pred == 6:

              pred = "disgust ,The subject is a deep sound of haste for the sutivation surrounding the
user and is expected to react in a unpleasant way"

            return pred

        elif pred == 7:
            pred = "surprised ,The subject is very excited also experiencing a state of anxiety and is
expected to react in a pleasant way."
          return pred



  layout=[
    [sg.Text('Input the Wav File for Analysis')],
    [sg.Input(key='fin'),sg.FileBrowse()],
    [sg.Text('Emotion is')],
    [sg.Output(size=(70, 15),font=('Century Gothic',10),key='fout')],
    [sg.Submit(button_color=('white', '#508CA4')),sg.Cancel(button_color=('white', '#508CA4'))]
]
window=sg.Window('Emotion Analysis Through Speech').Layout(layout)
button,finp=window.Read()
while True:
    event, values = window.Read()
    if event is None or event == 'Cancel':
        break
                pred = livePredictions(path='C:/Users/hkondru/Desktop/Emoce/Emoce
final/Ravdess_model/model.h5',file=finp['fin'])
    pred.load_model()
    a=pred.makepredictions()

    window.FindElement('fout').Update(a)

window.Close()
```

6.10 Code block for live recording module

Recording.py

```
  import pyaudio
import wave
CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 2
RATE = 44100
RECORD_SECONDS = 4
WAVE_OUTPUT_FILENAME =    “import pyaudio”
  import wave
CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 2
RATE = 44100
RECORD_SECONDS = 4
WAVE_OUTPUT_FILENAME  =                              "C:/Users/hkondru/Desktop/Emoce/Emoce
final/output2.wav"
  p = pyaudio.PyAudio()
stream=
p.open(format=FORMAT,channels=CHANNELS,rate=RATE,input=True,frames_per_buffer=C
HUNK)
print("Start talking!! your voice is being recorded..!!")
```

```
frames = []
for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
   frames.append(data) # 2 bytes(16 bits) per channel
print("*Thank You!! done recording")
 stream.stop_stream()
stream.close()
p.terminate()
wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b.join(frames))
wf.close()
```

CHAPTER-7
SOFTWARE TESTING

7. Software testing

7.1 Definition

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs.The sample data are used for testing. It is not the quantity, but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately and efficiently before live operation commands.

Software testing is one of the main stages of project development life cycle to provide our cessation utilizer with information about the quality of the application and ours, in our Project we have undergone some stages of testing like unit testing where it's done in development stage of the project when we are in implementation of the application after the Project is yare we have done manual testing with different Case of all the different modules in the application we have even done browser compatibility testing in different web browsers in market, even we have done Client side validation testing on our application.

7.2 Testing Objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

- A successful test is one that uncovers an error that hasn't been discovered yet.
- A good test case is one that has probability of finding an error, if it exists.
- The test is inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

7.3 Levels of Testing

In order to uncover present in different phases we have the concept of levels of testing. The basic levels of testing are:

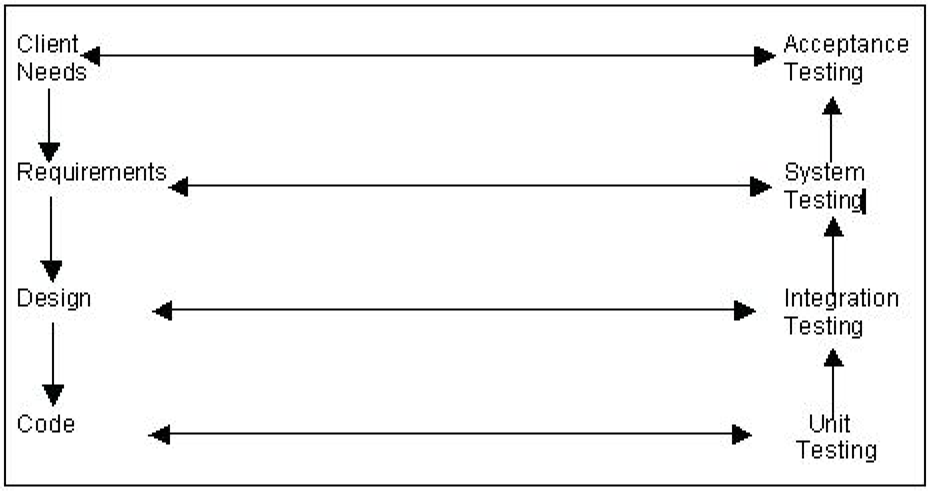


Figure 7.1 Levels of Testing

Code testing:

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

Specification Testing:

Executing this specification starting what the program should do and how it should perform under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

Unit testing:

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module.

There are some validation checks for fields also. For example, the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system. The unit testing is done in the stage of implementation of the project, only the errors solved in development stage and some of the errors we come across in development are given below.

Each module can be tested using the following two strategies:

1. Black Box Testing
2. White Box Testing

7.4 Black Box Testing

Black box testing is a software testing technique in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications. In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

Input
Black Box
Output

Figure 7.2 Black Box Testing

The above Black Box can be any software system you want to test. For example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes
- them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify
- that the SUT will be able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

- Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
- Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing – Regression testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

7.5 White Box Testing

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability. White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the **box testing** approach of software testing. Its counter-part, black box testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term **whitebox** was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell(or box) into its inner workings. Likewise, the **black box** in **black box testing** symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into two basic steps. This is what testers do while testing an application using the white box testing technique:

Step 1) Understand the Source Code

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software.The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) Create test cases and Execute

The second basic step to white box testing involves testing the application's source code for proper flow and structure.One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

System Testing:

Once the individual module testing is completed, modules are assembled and integrated to perform as a system. The top down testing, which began from upper level to lower level module, was carried out to check whether the entire system is performing satisfactorily.

There are three main kinds of System Testing

i. Alpha Testing

ii. Beta Testing

iii. Acceptance Testing

Alpha Testing:

This refers to the system testing that is carried out by the test team with the Organization.

Beta Testing:

This refers to the system testing that is performed by a selected group of friendly customers.

Acceptance Testing:

This refers to the system testing that is performed by the customer to determine whether to accept the delivery of the system or not.

Integration Testing:

Data can be lost across an interface, one module can have an adverse effort on the other sub functions, when combined, may not produce the desired major functions.Integrated testing is the systematic testing for constructing the uncover errors within the interface. The testing was done with sample data. The developed system has run successfully for this sample data. The need for integrated test is to find the overall system performance.

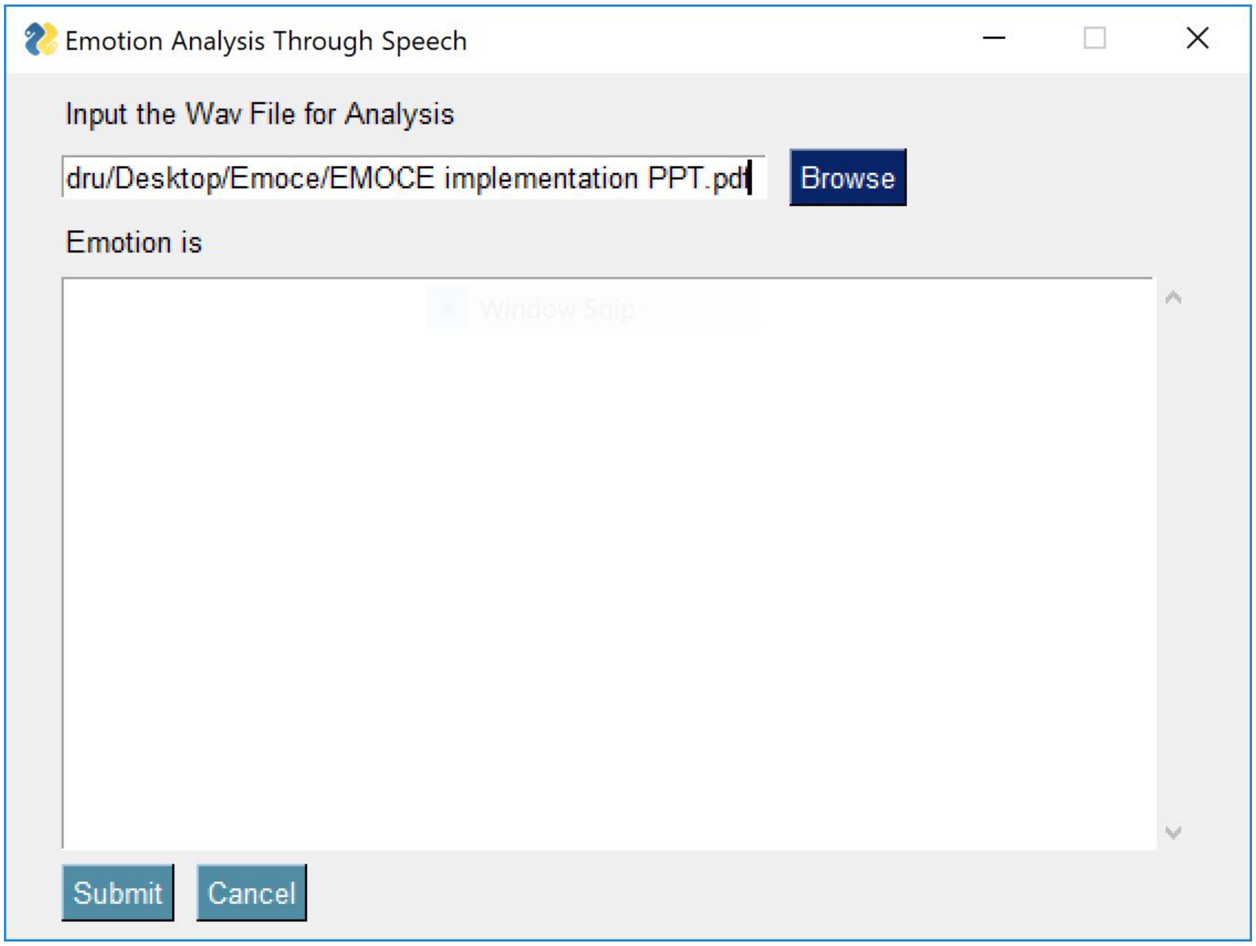
Output testing:

After performance of the validation testing, the next step is output testing. The output displayed or generated by the system under consideration is tested by asking the user about the format required by system.

7.6 Testing done when application is in development stage

7.6.1 Input data Error

This project is designed for the prediction of .wav files that contains the voice data which are used for predicting the emotions of the voices based on the Mel Frequency Cepstral Coefficient values . By taking the input data file of any other format other than .wav makes this GUI to get closed abruptly.

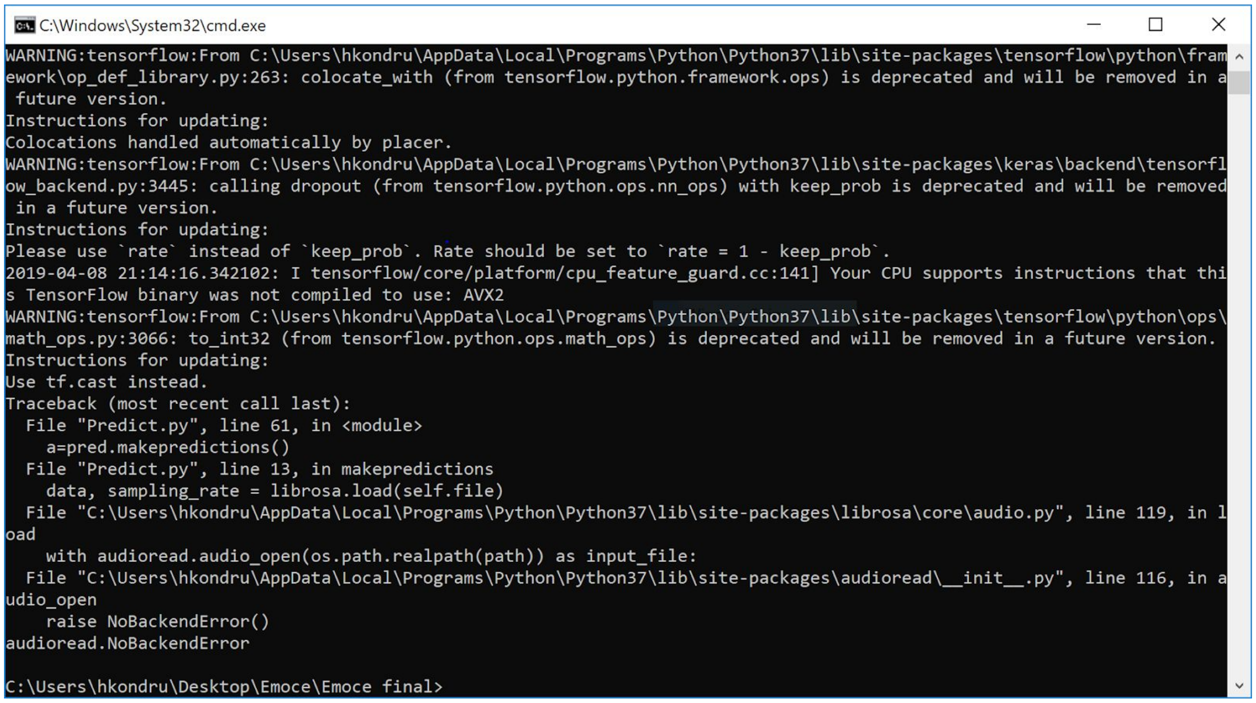


Emotion Analysis Through Speech

Input the Wav File for Analysis

dru/Desktop/Emoce/EMOCE implementation PPT.pdf

Figure 7.3 Input data Error



```
WARNING:tensorflow:From C:\Users\hkondru\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\fram
ework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a
 future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\hkondru\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorfl
ow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
 in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
2019-04-08 21:14:16.342102: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX2
WARNING:tensorflow:From C:\Users\hkondru\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\
math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Traceback (most recent call last):
  File "Predict.py", line 61, in <module>
    a=pred.makepredictions()
  File "Predict.py", line 13, in makepredictions
    data, sampling_rate = librosa.load(self.file)
  File "C:\Users\hkondru\AppData\Local\Programs\Python\Python37\lib\site-packages\librosa\core\audio.py", line 119, in l
oad
    with audioread.audio_open(os.path.realpath(path)) as input_file:
  File "C:\Users\hkondru\AppData\Local\Programs\Python\Python37\lib\site-packages\audioread\__init__.py", line 116, in a
udio_open
    raise NoBackendError()
audioread.NoBackendError

C:\Users\hkondru\Desktop\Emoce\Emoce final>
```

Figure 7.4 Error of Input DataFile

This is resolved in the later stages using exceptional handling which can give a notification to the user that only .wav data files can be taken as input.

7.7 Manual testing on project application

7.7.1 TEST CASES

S#	Prerequisites	S#	Test Data Requirement
1	PreProcessed data	1	The audio file should be with .wav extension.

Test Condition
Emotion of the voice data file.

Test Case ID	Test Case File	Expected Results	Actual Results	Pass/Fail/Not Executed/Suspended
1	01-01-01-01-02-02-05.wav	neutral	neutral	Pass
2	03-02-03-01-02-02-06.wav	happy	Happy	Pass
3	output1.wav	calm	sad	Fail
4	output2.wav	happy	happy	Pass
5	03-02-06-02-02-02-14.wav	Angry	Fearful	Fail
6	02-02-02-02-01-02-06.wav	Calm	Calm	Pass

7	01-01-01-01-01-02-03.wav	Neutral	Calm	Fail
8	02-02-06-02-01-02-19.wav	Clam	Clam	Pass
9	03-02-01-02-02-02-12.wav	Disgust	Disgust	Pass

Table 1 Results

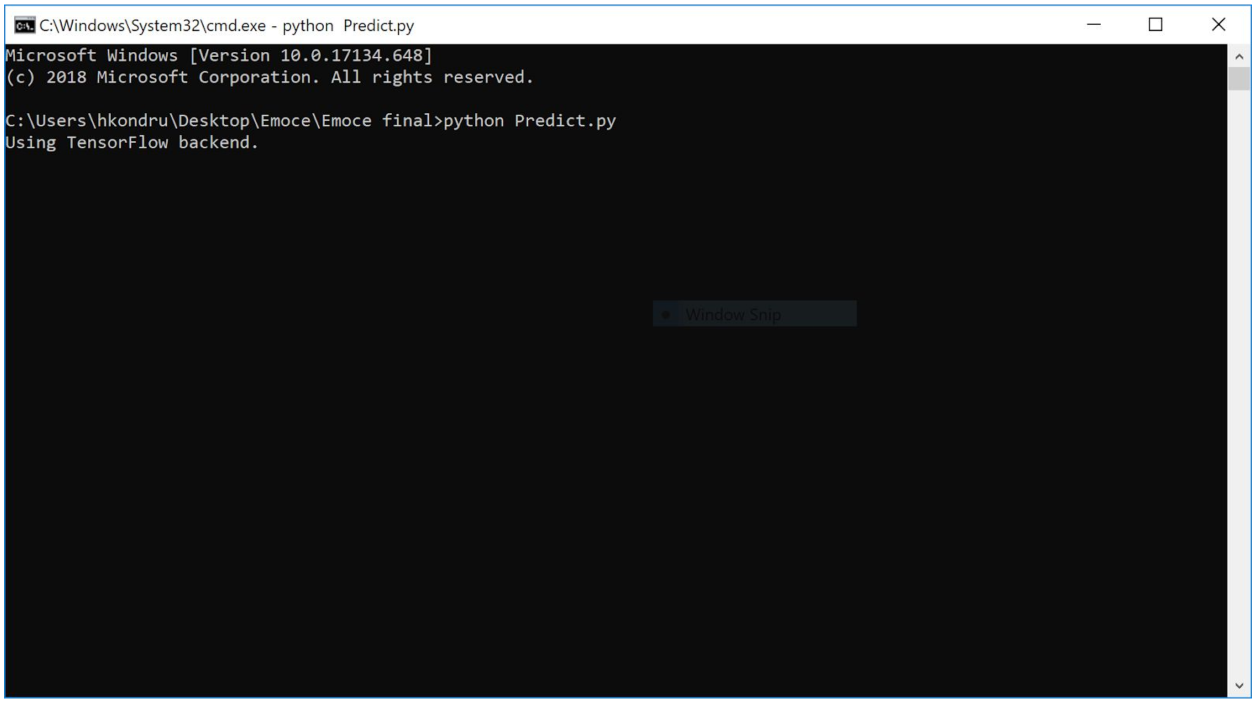
CHAPTER-8

RESULTS

8. RESULTS

8.1 Project Training Module

The below figure shows the execution of prediction.py code with would enable the user to use a GUI where the user can give a data File as input to the constructed CNN model which can give a most approximate emotion as output to the loaded data file as per the MFCC Coefficients extracted from it using different packages.



```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\hkondru\Desktop\Emoce\Emoce final>python Predict.py
Using TensorFlow backend.
```

Figure 8.1 Prediction Module

8.1.1 User GUI to Load selected DataFile

This GUI provides the User with an interface to browse and load any data file with extension .wav that is already existing in the user's system.

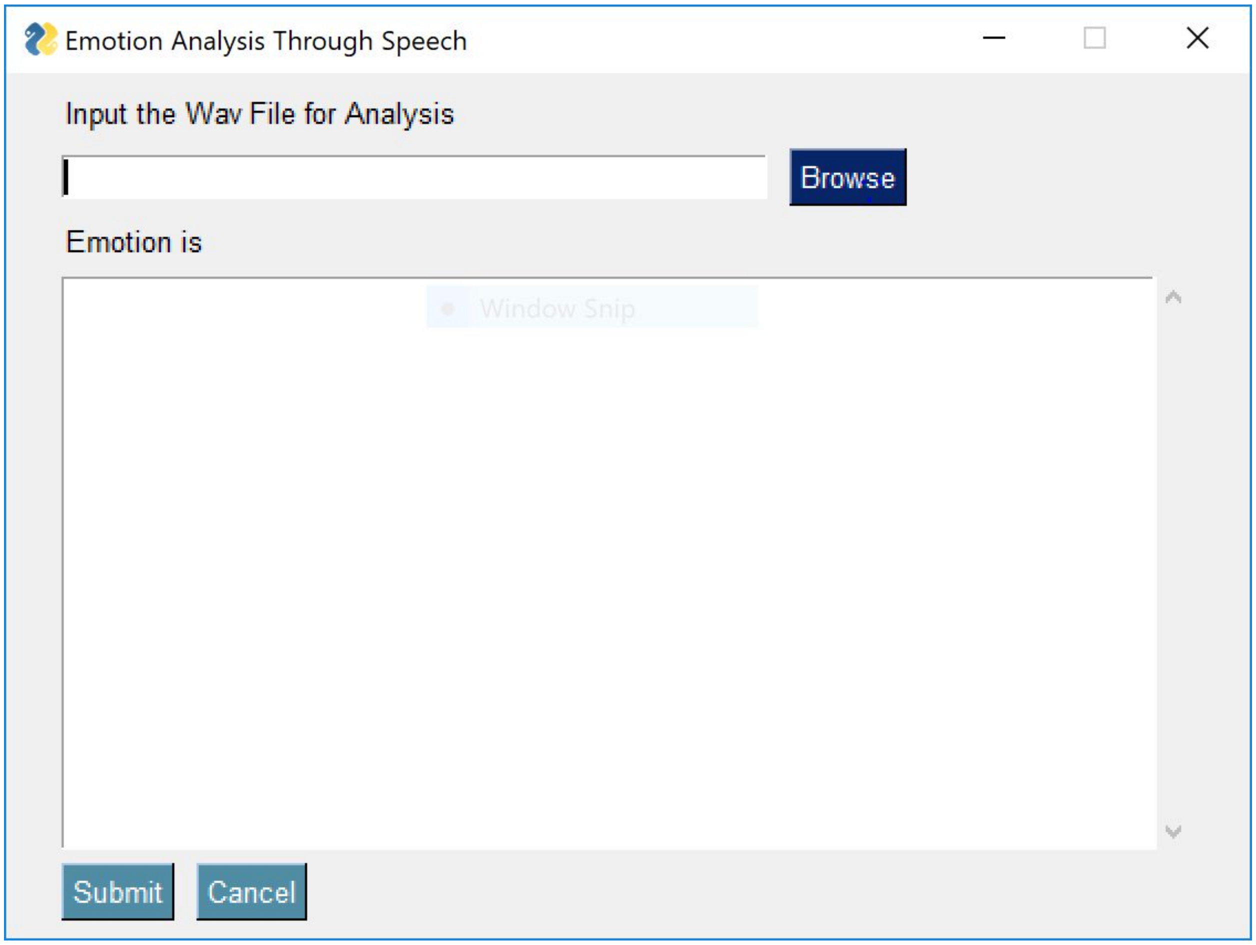

Emotion Analysis Through Speech
Input the Wav File for Analysis
Browse
Emotion is
Window Snip
Submit
Cancel

Figure 8.2 User GUI to browse data file

8.1.2 Loaded Sample Data File

This loaded data File is sent as an input argument to the Convolution neural Network Model build by training the datasets.

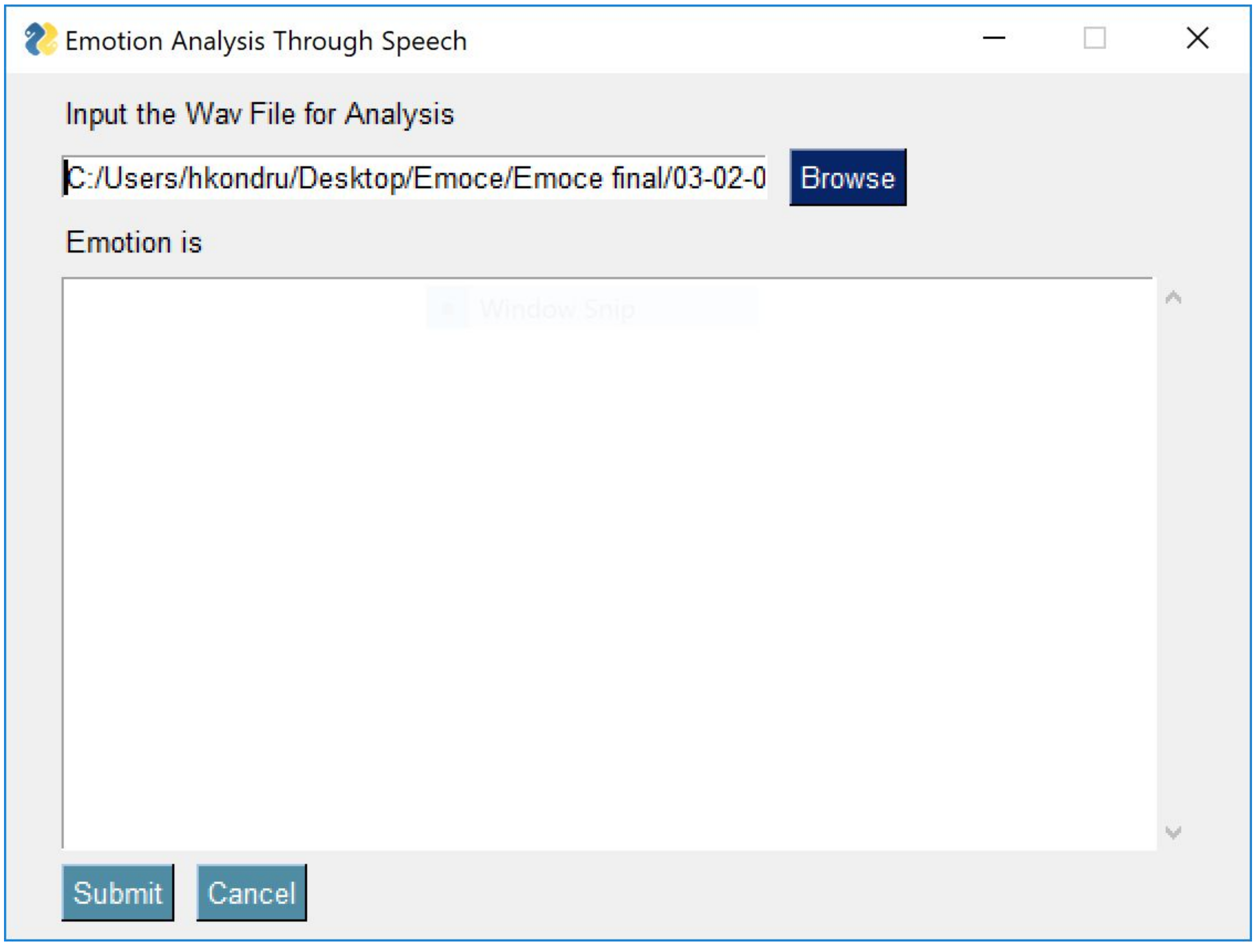
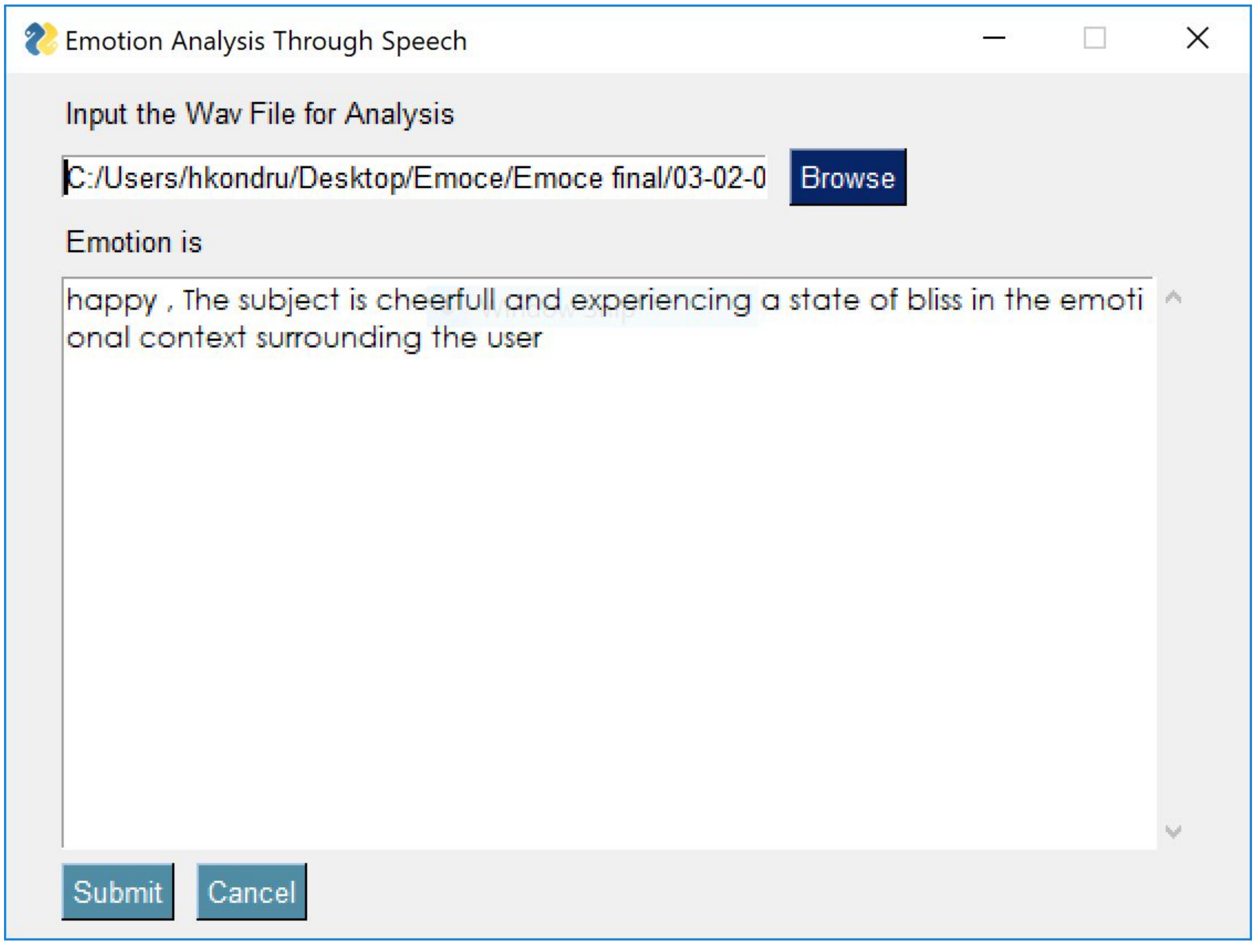

Emotion Analysis Through Speech
Input the Wav File for Analysis
C:/Users/hkondru/Desktop/Emoce/Emoce final/03-02-0
Browse
Emotion is
Submit
Cancel

Figure 8.3 Loaded Sample DataFile

8.1.3 Predicting emotion of dataset

The below interface displays the predicted emotion for the loaded data file by the user.This emotion is usually predicted by using the CNN model that is built by using the MFCC Coefficients extracted from the data sets using different packages.



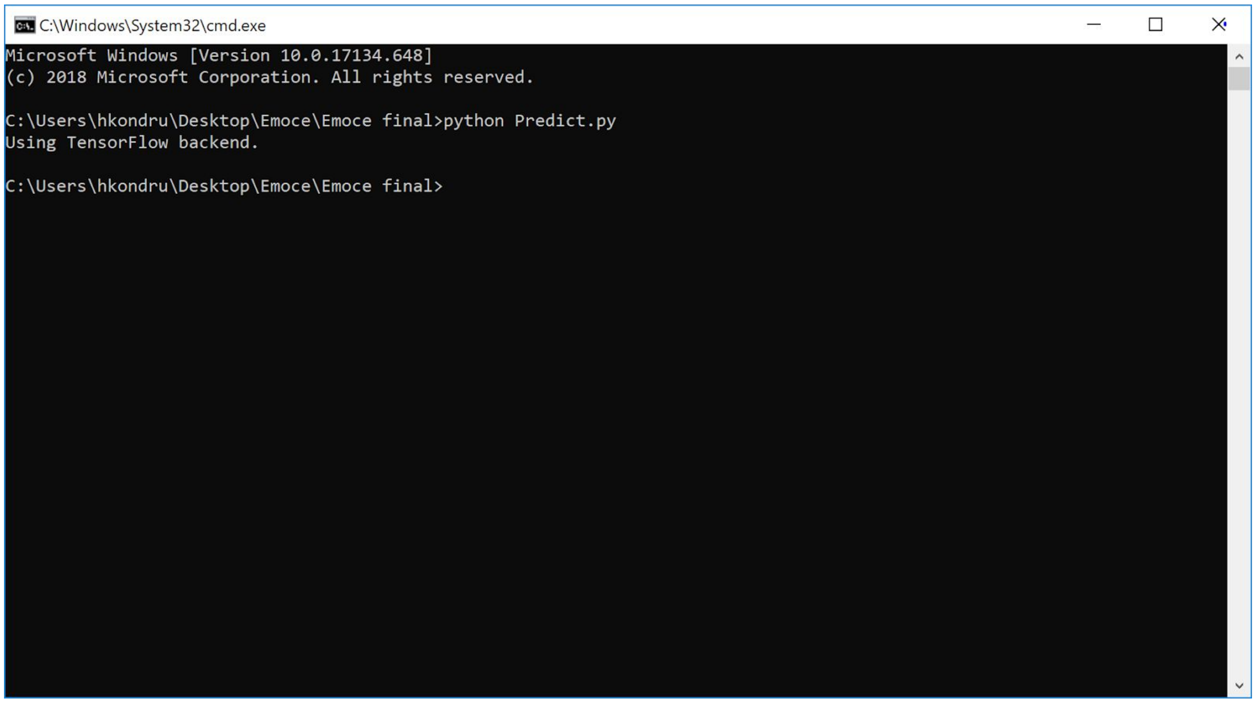
Emotion Analysis Through Speech
Input the Wav File for Analysis
Browse
Emotion is
Submit
Cancel

C:/Users/hkondru/Desktop/Emoce/Emoce final/03-02-0

happy , The subject is cheerfull and experiencing a state of bliss in the emoti
onal context surrounding the user

Figure 8.4 Predicting emotion of dataset

8.1.4 Predict.py GUI end Module

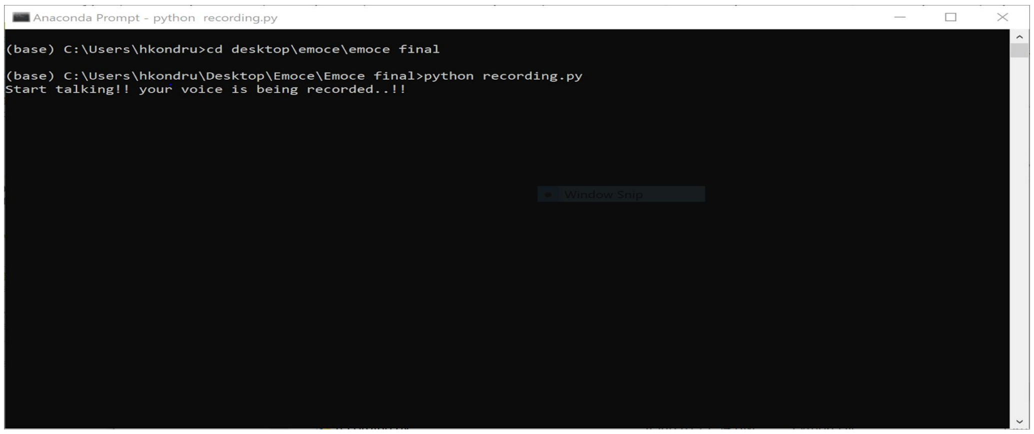


```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\hkondru\Desktop\Emoce\Emoce final>python Predict.py
Using TensorFlow backend.

C:\Users\hkondru\Desktop\Emoce\Emoce final>
```

Figure 8.5 Predict.py End Module

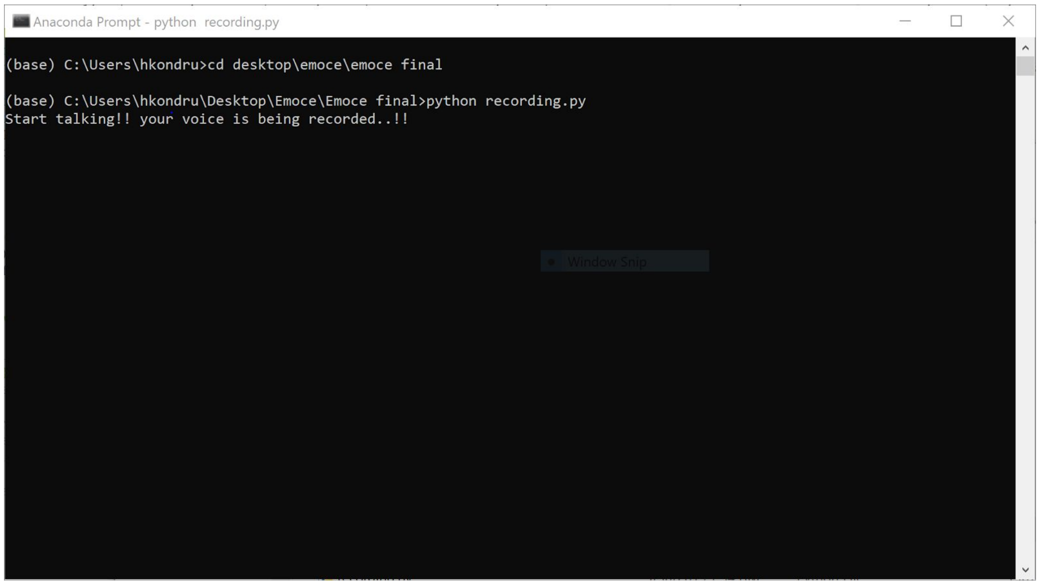
8.2 Recording Module

In this module we can record the voice data of the user and save it in .wav form which can be useful for predicting the emotion of that data file.



```
(base) C:\Users\hkondru>cd desktop\emoce\emoce final

(base) C:\Users\hkondru\Desktop\Emoce\Emoce final>python recording.py
Start talking!! your voice is being recorded..!!
```

Figure 8.6 Voice Recording



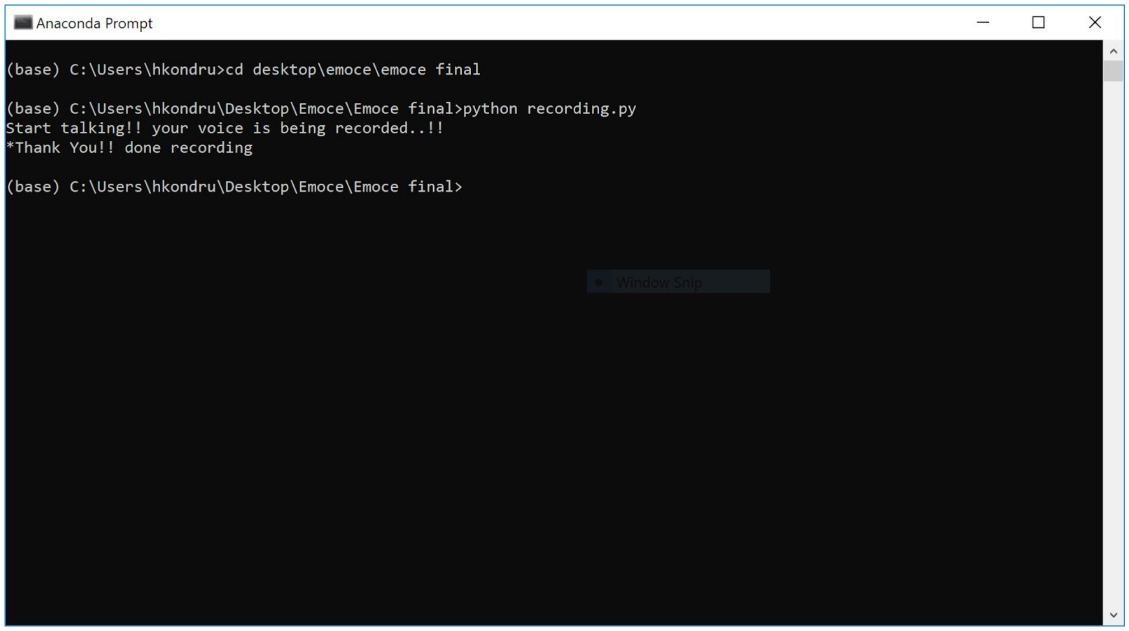
Anaconda Prompt - python recording.py

```
(base) C:\Users\hkondru>cd desktop\emoce\emoce final

(base) C:\Users\hkondru\Desktop\Emoce\Emoce final>python recording.py
Start talking!! your voice is being recorded..!!
```

Figure 8.7 Record.py Module

8.2.1 Recording Voice Data

There is a notification given to the user that specifies the user to start talking and his voice is being recorded and it later gives another notification that it stopped recording the data.



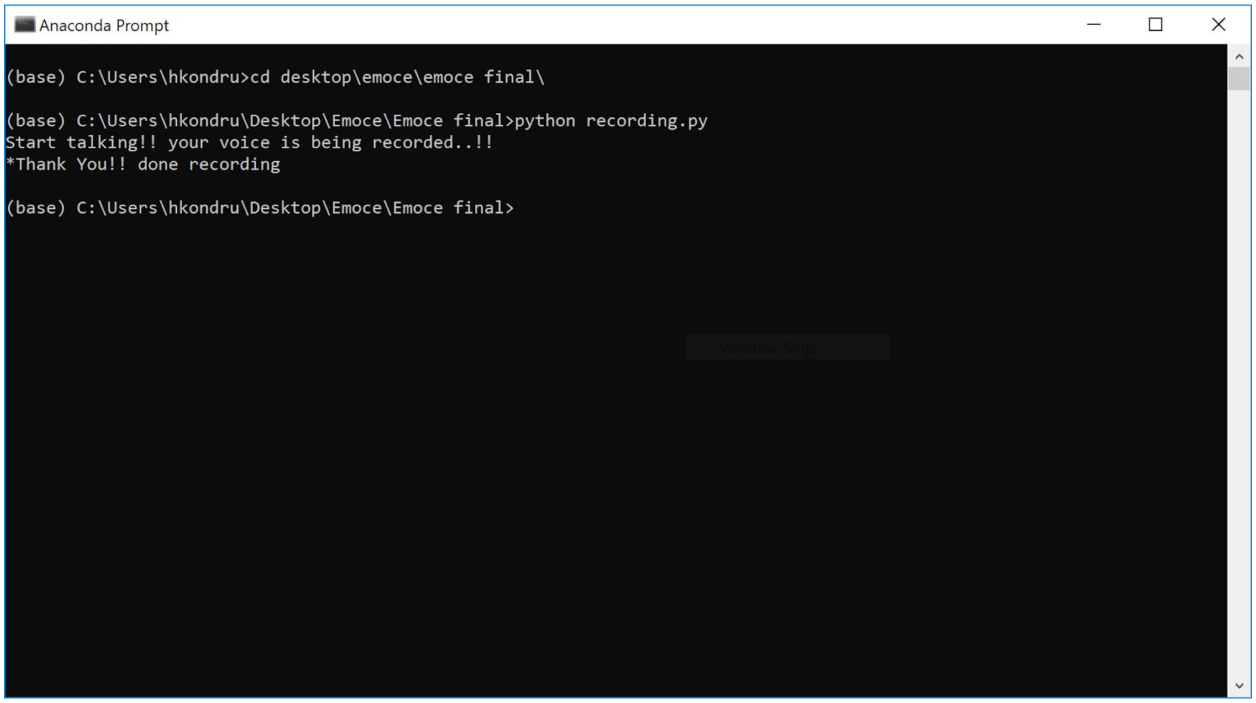
Anaconda Prompt

```
(base) C:\Users\hkondru>cd desktop\emoce\emoce final

(base) C:\Users\hkondru\Desktop\Emoce\Emoce final>python recording.py
Start talking!! your voice is being recorded..!!
*Thank You!! done recording

(base) C:\Users\hkondru\Desktop\Emoce\Emoce final>
```

Figure 8.8 Recording Voice Data

8.2.2 Saving Recorded data

This specifies the user that the voice data is stopped recording and the recorded data is saved with .wav extension format which can be useful for the prediction of emotion.

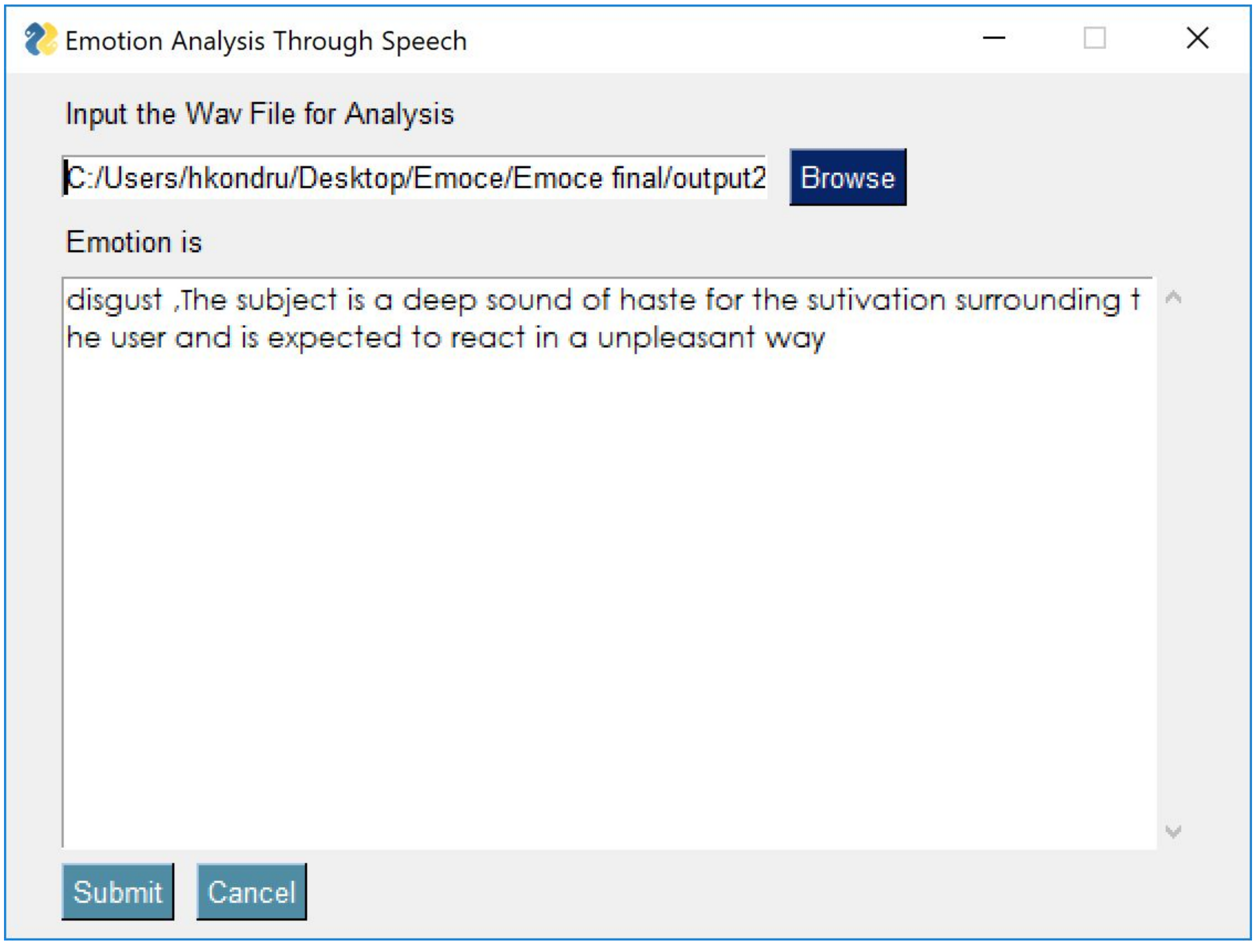


Anaconda Prompt

```
(base) C:\Users\hkondru>cd desktop\emoce\emoce final\

(base) C:\Users\hkondru\Desktop\Emoce\Emoce final>python recording.py
Start talking!! your voice is being recorded..!!
*Thank You!! done recording

(base) C:\Users\hkondru\Desktop\Emoce\Emoce final>
```

Figure 8.9 Saving Recorded Data

8.2.3 Predicting the saved Data Record


Emotion Analysis Through Speech
Input the Wav File for Analysis
Browse
Emotion is
Submit
Cancel

C:/Users/hkondru/Desktop/Emoce/Emoce final/output2

disgust ,The subject is a deep sound of haste for the sutivation surrounding t
he user and is expected to react in a unpleasant way

Figure 8.10 Predicting the saved Data Record

CHAPTER-9 CONCLUSION AND FUTURE SCOPE

9. CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

- Additional functionalities such as better graphic representation and live audio recording have been developed to elevate user experience.
- Understanding and analysing the emotions of a firm's employees and customers enables the firm to tailor its product to perfection also resulting in contentment of society.
- The problem statement enhances the research component by gathering data from multi-culturally diversified speakers.

9.2 Future scope

- Handling lengthier audio files: Being able to train the model and test it with longer audio datafiles yielding better accuracy.
- Efficient extraction of MFCC coefficients: MFCC values are not very robust in the presence of additive noise, and so it is common to normalise their values in speech recognition systems to lessen the influence of noise. Some researchers propose modifications to the basic MFCC algorithm to improve robustness, such as by raising the log-mel-amplitudes to a suitable power (around 2 or 3), which reduces the influence of low-energy component.

BIBLIOGRAPHY

References:

[1]Assel Davletcharova, “Detection and Analysis of Emotion From Speech Signals”, Jan. 20, 2015. [online]. Available: https://arxiv.org/ftp/arxiv/papers/1506/1506.06832.pdf [Accessed: Dec. 25, 2018]

[2] Xingjie Wei,” Emotion Analysis for Personality Inference from EEG Signals”, Dec. 29, 2017. [online]. Available: https://ieeexplore.ieee.org/document/8241761 [Accessed: Jan. 2, 2019]

[3]Nidhi Chandra, ”A Proposed Approach with Analysis of Speech Signals for Sentiment Detection”,October,01,2015[Online].Available:https://ieeexplore.ieee.org/document/7279936

[4]Poorna Banerjee Dasgupta, “Detection and Analysis of Human Emotions through Voice and Speech Pattern Processing”, 1 October 2017[Online].Available: https://arxiv.org/ftp/arxiv/papers/1710/1710.10198.pdf

SHOW AND TELL
PANDA PUNNAIAH SQUARE