Voice-based Fraud Detection System using Speech Emotion Recognition

Project Thesis

Fonseka H.F.N.K.

CS/2016/011

B.Sc. (Hons) in Computer Science specializing in Data Science

Faculty of Computing and Technology

University of Kelaniya

Sri Lanka

Declaration

I hereby declare that the work, which is being presented in this thesis, entitled "Voice-based Fraud Detection System using Speech Emotion Recognition", in partial fulfillment of the requirement for the award of the degree of B.Sc. (Hons) in Computer Science specializing in Data Science and submitted to the University of Kelaniya, Sri Lanka is an original work under the guidance of Dr. Rasika Rajapaksa, Senior Lecturer, University of Kelaniya.

I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree.

Abstract

In this research, the application of Speech Emotion Recognition, Neural Networks, and Fraud Detection is investigated to create a Voice-based Fraud Detection System. The idea in this research can be used to build an intelligence system and can be extended with many other features and any domain.

Some researchers recently had an interest in this SER concept. One is based on NLP which is capable of getting an audio file as an input and turning it into a text using the Google Speech Recognition API. Using stop words and Stemming words it will process into SVM classification and then from the Natural language toolkit, it will train data as Positive Negative, and Neutral [1]. But this system must train for many languages and the size of the data set will depend on the result and there is a language barrier.

Another research paper that was published by the Department of Computer Science Electrical Engineering, University of Missouri, USA proposed a novel solution for SER using deep learning techniques. They have used Convolutional Neural Network (CNN) technique to classify speech emotions. Also, their SER model is a live speech system that can predict real-time emotion [2].

Also, a research paper that was published in 2018 presented a novel treasure-hunting game. In each turn, a player chooses whether to lie or not about the location of the treasure and other players have to guess where the treasure is hidden. Here there is a system to detect whether the player is lying or not by analyzing the voice and gestures. In this study, they have learned that if the player is lying that person will speak slowly, pause longer, and speak in high pitched voice [3].

This system enables the user to avoid the shortcomings in the manual call center procedures. The system also can predict whether the customer or the caller is genuine or not, using the powerful Neural Network module. The response can be given in any visualization method like pie chart, bar chart, progress bars, and text format for a better experience. The speech Emotion recognition model is created according to the British accent, but it can be used for any other Language.

Acknowledgment

I would like to acknowledge and give my warmest thanks to my supervisor, Dr. Rasika Rajapaksha who guided me throughout the project and made this work possible. His advice and supervision carried me through all the stages of developing the application and Documentation.

I would also like to appreciate the guidance and the solid theoretical base given by the lectures of the Faculty of Computing and Technology, University of Kelaniya, Sri Lanka. Also, I want to thank my internship supervisor Mr. Uditha Bandara, Bluechip Technologies Asia for allowing me to work there as an Intern in Data Science and Machine Learning. During those six months, I learned a lot about the Practices of Computer Science, especially about Computer Vision, Machine Learning Algorithms, and Data Science which helped me to complete my research work.

Finally, I want to thank my parents who always supports me and guide me.

Table of Contents

List of figures

List of Abbreviations

AI – Artificial Intelligence

API – Application Programming Interface

NLP – Natural Language Processing

UI – User Interface

CSV – Comma- Separated Values

HTML – Hypertext Markup Language

CSS – Cascading Style Sheet

SaaS – Software as a Service

PaaS – Platform as a Service

VM – Virtual Machine

AWS – Amazon web services

Wav - Waveform Audio File

1. Project Charter

1.1. Description of the problem

According to Forbes, frauds are a common way for criminals to access people's personal and financial information. In 2020, the Federal Trade Commission received more than 2.1 million fraud reports from consumers. According to a 2021 report from Truecaller, money lost to fraud calls in the past 12 months equates to an estimated USD 29.8 billion. Scammers use various methods to trick people into giving up sensitive information like bank account numbers and passwords. Therefore, identifying scammers is incredibly important for public safety and financial sector stability.

In the modern world, the Insurance industry has become extremely popular and the main problem in this sector is Insurance Fraud. Insurance Fraud refers to deliberate acts with the intention of misleading lead the insurer to obtain financial benefits. Common frauds include providing false facts on the insurance application and submitting claims for injuries or damages that have never happened [4]. In Sri Lanka also these situations are happening which will directly affect the growth of those organizations. Because of this problem, the value of speech emotion recognition systems is increasing day by day. But there are very few researches in the domain of audio-based fraud detection that has been previously done. People who commit this kind of fraud include organized teams or criminals who steal a large number of sums through fraudulent businesses, ordinary people who are cheating by paying fewer subscriptions, and simulated disasters to make money[5].

There are algorithms for fraud detection. Yet most of those algorithms use Natural Language Processing (NLP) and primarily support only the English language. Hence the existing system is not particularly useful to a multilingual society. The expected outcome for the project is to develop an algorithm using deep learning to classify calls using speech emotion recognition. This novel approach will overcome the language barrier, and the classification happens from the vocal tone. As we know **Sinhala** is the native language of approximately 70 percent of Sri Lanka’s population, which equals about 13 million people and **Tamil** is the second official language, spoken by about five million people on the island, which is about 15 percent of the population[6]. Hence the existing systems are not very useful for our country because only a few people speak English. So, it will be very useful if there is a system that supports multiple languages to identify fraud calls.

1.2. Stakeholders

This system will help reduce the cost of analyzing audio manually, and it helps to avoid losses for the company due to fraud calls. This deep learning-based system will benefit the insurance industry, especially for call center agents and we can customize this to the banking sector, financial institutes and telecommunication industry, and any company with a call center.

The system can be used for criminals' vocal recording classification in law enforcement agencies, e.g., CID (Criminal Investigation Department), Intelligence, etc.

1.3. Business needs

Over the past two years, fraud has grown with an average of 6 frauds per company and the global cost of insurance fraud is about $80 billion [7]. It weakens the financial position of the businesses and causes grave consequences to the stakeholders. Currently, Sri Lankan Insurance companies detect fraud calls manually by listening and going through the documents repeatedly. It is not a reliable method, and they have to assign people for that task, and it’s very time-consuming. So, the insurance industry needs an automated solution for this.

1.4. Assumptions

As mentioned in the previous paragraph insurance companies may be assigning their staff to identify frauds manually by listening to the call recordings and it will cost them more money for their salaries. Also, it is not very realistic because it is extremely hard to identify whether the caller is genuine or suspicious by listening to the call. It will be easy if there is a system to identify the emotions of audio and it will save the cost of the salaries and the time. Also, some people might be fearful or stressed when faced with an accident and their voices can deviate from normal voices thus giving less accurate results. So, this may give the result a suspicious person since we do not have the relative datasets from the insurance company to train the model. In business aspects, we can get their data and train the model to increase accuracy.

1.5. Constraints

Constraints are lack of relative data and lack of computer power. Here lack of data means as mentioned before the relative dataset is not there. So, we must use the datasets which are already available on the internet. In machine learning, the accuracy directly depends on the dataset from which the model is being trained on. More data will increase the accuracy. But to manage a huge dataset on the local computer I have a storage limitation. So, the lack of computer compatibility matters.

Here I used the emotional audio dataset because I could not get access to the original lie and truth audio data which are in the insurance company databases. So, I had to detect whether the person is genuine or not using their emotions.

Another limitation is in the system we can get any audio and get the result. So, the insurer can use this for normal conversations when further investigating a problem by recording the conversation. But it may lead to privacy issues.

2. Literature Review

Despite the huge progress made in Artificial Intelligence (AI), we are still not able to naturally build interactions with machines because machines do not understand our emotional states. Recently the attention of the researchers has been increasing this Speech Emotion Recognition (SER) which tries to recognize emotional states by analyzing the speech signals[8]. Emotions are important in human communication and SER can be used in many applications like robots, mobile services, etc. Since 1920 researchers and scholars have taken this research area forward through numerous methods. Deep learning has improved the mechanism of recognizing human emotions from speech but still, there are some loopholes in the research of SER such as a shortage of data and insufficient accuracy of the trained models[9]. The first SER model based on deep learning was proposed in 2014 by Han et al[10].

There are diverse ways to represent emotions. For example, we can consider Ekman's six basic emotion model which is able to identify anger, disgust, sadness, fear , happiness, and surprise[11]. To train as we want, we need data. For that, several open-source databases can be found on the internet the first example we can consider RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) database. It includes emotional data of 24 different actors and those actors were asked to read two different sentences in many ways in North American English. The spoken styles they used are neutral, calm, happy, surprised, sad, fearful, angry and disgusted. It includes a total of 1440 files with these emotions[12]. The second database is CREMA-D (Crowd-sourced emotional multimodal actors' dataset) database which is similar to RAVDESS. In this, 91 different actors recorded their speech for 12 different sentences with 6 different emotions namely neutral, happy, surprised, sad and disgust it includes 7442 files in total[13]. IEMOCAP (Interactive Emotional Dyadic Motion Capture) database contain a large amount of emotional data and includes audio-visual recordings of five sessions between female and male. It has a recording of 10 speakers total in 12 hours of data[14].

A research paper that was published in 2018 by the Department of Electrical and Computer Engineering at Seoul National University proposed a system for Multimodal Speech Emotion Recognition Using Audio and Text[15]. They proposed a model that uses low-level audio signals as well as high-level text transcription to improve information included within low-resource datasets towards a greater degree. They also proposed a novel deep dual recurrent encoder model that simultaneously improves text and audio data emotion recognition from speech and their model accurately identifies emotion classes.

The recent technological advancements in signal processing, machine learning, and linear algebra have resulted in high-performing voice recognition systems[16]. The applications can be observed in the research[17], as well as in commercial[18]. Modern engines use various models, statistical speaker-dependent models, such as mixtures of Gaussian distributions[19] and speech signal representations using super vectors[20] or i-vectors[21] combined with powerful machine learning algorithms.

There is an enormous amount of literature that uses machine learning approaches for classification [22]are few among them. The SVM is the most commonly used classifier for global features[23]. Some other classifiers, such as decision trees[24] and K-nearest neighbor (KNN)[25], have also been used in speech emotion recognition. These approaches require very high-dimensional handcrafted features chosen empirically.

Another system was proposed using Dempster-Shafer theory and Bayesian inferencing for superimposed fraud detection in mobile communication networks[26]. They used a supervised neural network to detect fraud in mobile telecommunication networks[27]. The notion of voice call graphs to represent voice calls from domestic callers to foreign recipients and used a Markov Clustering-based method for isolating dominant fraud activities from these international calls[28]. Although many approaches have been proposed to detect telecom fraud, some issues still deserve further consideration.

3. Requirements Specification

3.1. Purpose

Let us consider how this system will address the problems faced by the insurance sector. As mentioned before, the global cost of insurance fraud is nearly 80 billion dollars over the past two years, and it is a huge amount. Here the proposed system can analyze the calls in real-time and give a prediction as to whether the caller is genuine or not. When a call comes to the call center agent that person can start recording the call at the right moment and if it turns out suspicious, an agent can ask questions accordingly. Also, there needs to be a technique to identify fraud call recordings from the previous calls (history). In this system, the agent can input call recordings from the history and analyze them too.

So, the purpose is to help companies to avoid the wastage of money and time. When it comes to the financial state first thing is to avoid fraud so that they can save money. Second thing is to analyze these calls manually they need staff and some resources. This system will help to solve the problem with less workforce. Next is time and the company's service efficiency will be increased by the system and the problems related to these fraud activities will be solved in less time. It will also support the staff to make their decisions quickly.

3.2. Scope

The primary objective of the research is to create a web application based on Speech Emotion Recognition to detect fraud. The proposed system's SER ability will cater to a wide range of people including people with low computer literacy. Since the voice is the basic method of communication between insurer and customer it will be easy to use the system. The proposed system will be capable of analyzing the audio and outputting the emotions as percentages.

For the creation of a speech emotion recognition model, one can use two approaches. One approach is to use an existing dataset, or the other is to use a dataset provided by a specific area or a sector and then build the domain-specific speech recognizer. Explanation of both methods is mentioned with their respective objectives.

If an existing dataset is used objectives would be as follows,

RAVDESS dataset is one the best datasets we can use to create the model it includes audio clips of twenty-four actors and includes more spoken style compared to other datasets. According to those objectives are as follows.

- Preprocessing of the dataset.
    - Since we are analyzing the waves of the audio it should be in .wav format. If it is not, we need to convert it to the wav format.
    - Every audio needs to be in the same sampling rate. (Ex: 16000Hz)
- Build an accurate model.
- Create a simple user-friendly Web Application.
- Provide some analytics if needed. For example, progress bars or charts to visualize the data

If the dataset is from the Insurance sector it means Insurance companies normally have a database that includes the call recordings that are already proved as genuine and not and objectives will be followed.

- Collection and preprocessing of the audio dataset.
    - If the insurance company provides their data, they have for the past few years first we must convert those audios into the Wav format.
    - Must adjust the sample rate.

Other objectives will be the same as when used as an existing dataset.

3.3. Functional and non-function requirements

Background and a literature survey were done in the SER and Dataset to find out about the certain similar implementation that has been done by the scholars worldwide. As it is evident in the literature survey that there has not been done an SER-based Fraud detection system in the insurance domain and the following functional requirements were identified for the project.

- The proposed model should be able to identify the emotions of the caller.
- The proposed model should be able to identify the high-pitched voices and silences to detect audios that are suspicious.
- Must be able to work in different languages other than English.
- Caller agent should be able to record the call and get the result in real-time.
- UI should be simple and easy to manage

As non-functional requirements, these things can be identified

- Realtime result has to be given in less time which means systems has to be efficient.
- Since the application is related to customers and in this software, we save all those call recordings. So, privacy should be ensured. If we are using database security must be ensured.

3.4. Software and hardware requirements

First, for the coding requirement, it is better to use an IDE like PyCharm or also Jupyter notebook, or Google collab can be used. Since the programming language used here is python there should be some basic libraries installed on the computer. For the UI development HTML, and CSS can be used, and for the designing part we can even use Adobe tool. When it comes to the enterprise level it is better to use a database to store needed information and retrieve those in a dashboard.

Computer capability matters considering the analysis and model training. So, it is better to have a powerful computer so that we can do the training quickly and adjust needed hyperparameters to get an optimized model.

3.5. Security requirements

Since the application is using call recordings of the customer they must be stored in a database for further analysis. But the problem is storing personal information must be in a secure location because it can cause privacy issues. When the caller is contacting the company as per the normal procedure, they record the personal information of the caller and record the call. If this information is not stored in a secure way those details can be revealed by other parties and this will affect the privacy agreements that are signed between the customers and the company.

So, it is a major requirement to ensure security. We can go for a secure database so that we don't have to worry about security issues. Also, when purchasing the product security companies can request a license key that is customized for them so that people from outside cannot access the system.

4. Analysis and Design

4.1. Analysis

4.1.1. Data Collection

Using RAVDESS emotional dataset we can train and test the model. But few things need to follow up before using that model. The original size of the RAVDESS dataset is around 24GB but the idea is to use a smaller portion of it due to the lack of computational power and resources. Data preprocessing is very important because the whole system will be dependent on the training data set. So first to preprocess the data we need to identify what are things that need to change before feeding those data into the system. Since we are analyzing the waves of the audio all the audios must be in .wav format. Since the system will be implemented using the PyAudio library the sample rate of all the audio files must be the same. So, the next thing is to implement a script to handle this task.

4.1.2. Data Preprocessing

Before we begin to extract features, we must understand and analyze the data. First, let’s consider these audio files and how to analyze them step by step.

The first step is to understand the data we have already downloaded from Kaggle and GitHub. The emotional dataset is a predefined dataset that already has the labels to

Sound waves have a sampling rate that indicates the discrete intervals and quality of the audios. Each sample is the amplitude of the wave at some point of time interval or duration, where the bit depth decides how detailed the sample will be.

When we consider signal processing, sampling indicates the reduction of a continuous signal into a series of discrete values. The number of samples taken in a fixed amount of time is called the sampling frequency or rate. If the audio has a high sampling frequency, then it provides less information loss, but the higher computational expense and low sampling frequencies have higher information loss, but they are fast and less expensive for computation.

Sound can be represented as an audio signal having parameters like bandwidth, frequency, etc. Normally, the audio signal can be described as a function of Time and Amplitude. Some devices are built to help you get these sounds and represent those in computer-readable formats. For example, wav (Waveform Audio File) format, WMA (Windows Media Audio) format, mp3 (MPEG-1Audio Layer 3) format, etc.[29]

Generally, audio processing involves the acoustics feature extraction relevant to the task and some decision-making tasks such as detection, classification, etc. Fortunately, we have many useful python libraries which make this task reliable and easier.

I have mainly used two libraries for audio acquisition and playback:

- Librosa

A Python module to analyze audio signals which include the basic functions or methods to build a MIR (Music information retrieval) system. Installation is simple with the pip command which we normally use for any other library[30].

Using Librosa, an audio signal can be represented as a one-dimensional NumPy array, as “y”. This y is describing the sampling rate denoted by “sr”, the frequency (in Hz) at which values of y are sampled. The duration of a signal can be calculated by dividing the number of samples by the sampling rate,

duration_seconds = float(len(y)) / sr

By default, the librosa.load() function downmixes to mono by averaging right and left channels when loading audio files, and then resample the signal to the default sample rate(sr) 22050 Hz. Since I wanted to use a 16000-sample rate, I changed it to that value when loading the dataset.

The librosa. core submodule includes commonly used functions. Core functionality can be divided into four categories, and they are time-series operations, frequency conversion, spectrogram calculation, and pitch changeable operations.

The librosa. feature module includes a variety of spectral representations and most of them are based on Short-time Fourier Transform. The Mel frequency scale is used to describe audio signals since it provides a similar model of human frequency. Mel-scale spectrogram (librosa.feature.Mel spectrogram) and Mel-frequency Cepstral Coefficients or in other words MFCC (librosa.feature.mfcc) are provided as features. While Mel-scale is used to capture timbral aspects of music, they produce less resolution of pitches and pitch classes. Pitch class or chroma can be used to encode harmony while suppressing variations in loudness, octave height, or timbre. Another representation of harmony and also pitch can be produced by the tonnetz function, which decides tonal centroids as coordinates in a six-dimensional space.

- SoundFile

An audio library based on libsndfile, CFFI, and NumPy. SoundFile can read and write sound files. Data can be written into the file using a soundfile. write () method, or read from the file using soundfile.read() method[31].

Here librosa library has more functionalities and predefined methods that can be used for the feature extraction of the dataset. So, I selected the librosa library over Soundfile.

Any sound that is generated by a human is determined by the shape of their vocal tract. If this can be determined accurately, any sound produced can be represented. There are many features of audios such as MFCC (Mel Frequency Cepstral Coefficient), Chroma, Mel, Tonnetz, contrast, etc. When training the model, we can analyze how each audio feature will impact.

After Loading and splitting the data we can use the MLP classifier which is a multi-layer perceptron classifier. It uses a neural network model to optimize the log-loss function using stochastic gradient descent or limited memory BFGS (Broyden–Fletcher–Goldfarb–Shanno) algorithm[32].

4.2.Design

Here the call center agent can record the call coming from the customer or can use audios from their history as input and analyze those. Output is given by a text and other visualization methods such as pie charts, bar charts, etc. All the emotions and their percentages are displayed on the result page in UI.

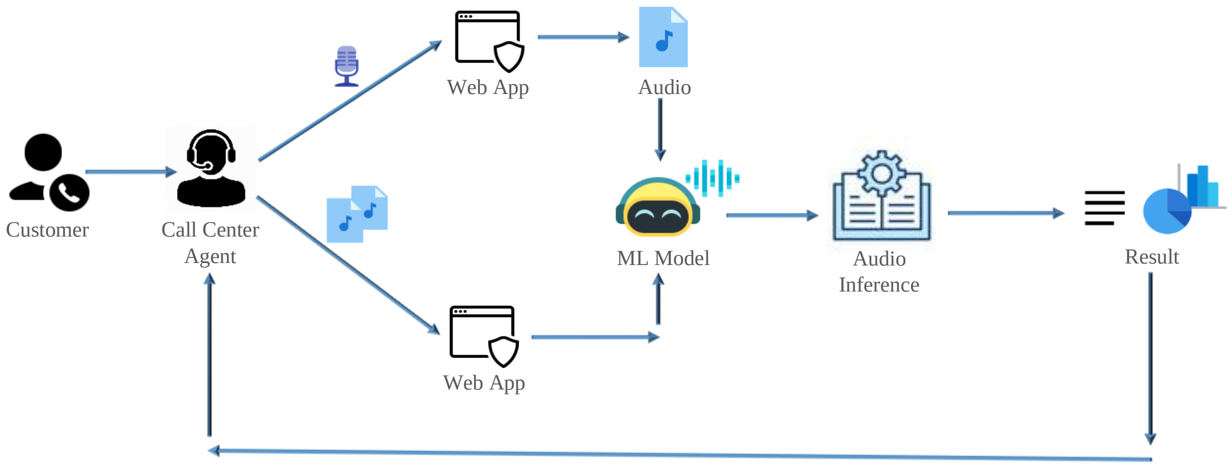
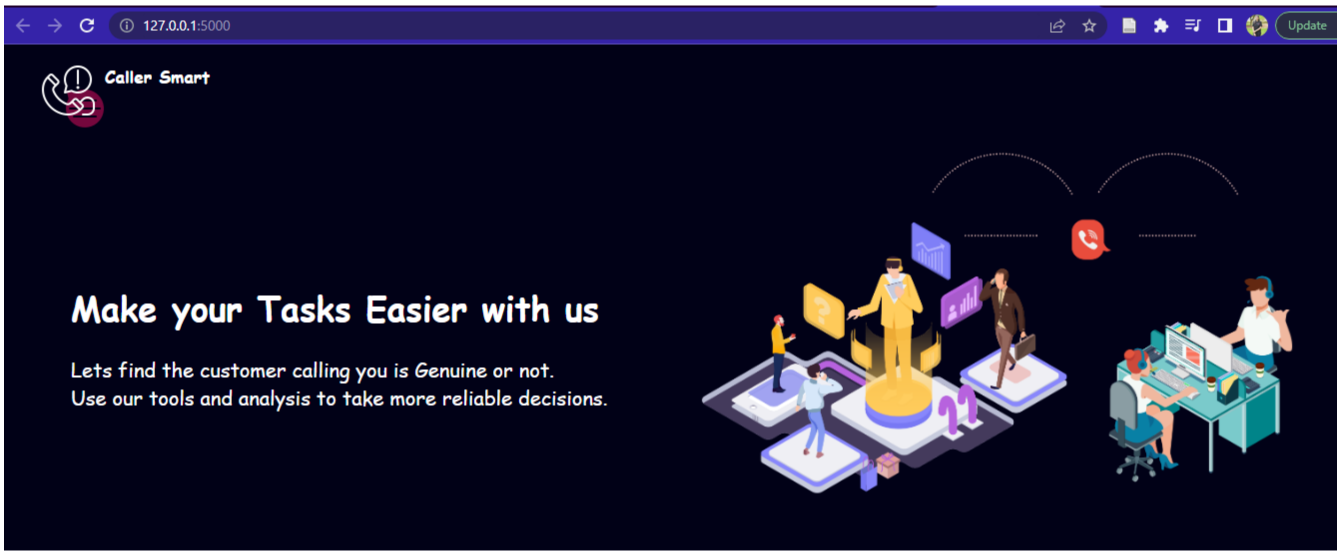


Figure 4.2.1 High-Level System Architecture

Let's move on to the actual design of the UI which is done by HTML and CSS.

First, the users have to enter the home page which will be navigating them to the recording page and bulk upload page. Index page is having a small brief about the application and what it does.

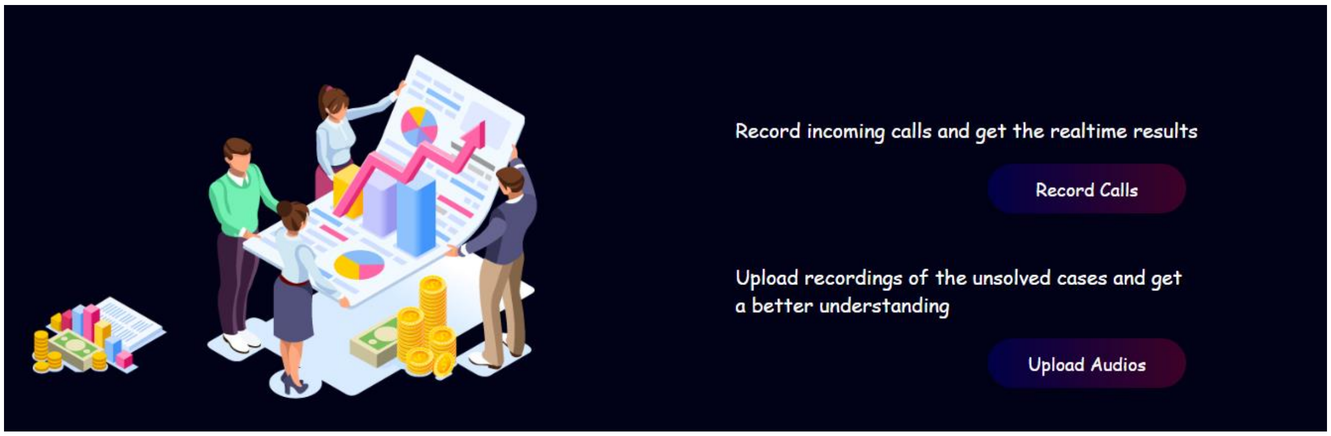


Make your Tasks Easier with us

Lets find the customer calling you is Genuine or not.
Use our tools and analysis to take more reliable decisions.

Figure 4.2.2 Home page

Names of the two buttons on the home page are "Record Calls" and "Upload Audios" which user will be redirected to the recording screen and upload the audios they have stored. There is a small brief for each button so that user can understand and move on to the next pages without any guidance.


Record incoming calls and get the realtime results

Upload recordings of the unsolved cases and get a better understanding

Figure 4.2.3 Home page including the buttons

When the user selects the first button which is “Record Calls” it will be redirected to the call recording interface. It shows user a drop-down list which he/she can select the duration that the audio will be recorded and then press “Start Recording” button. Afterwards user will see a loading animation in the tab which indicates the audio is being recorded and user have to wait until the selected duration will end and load the next page.

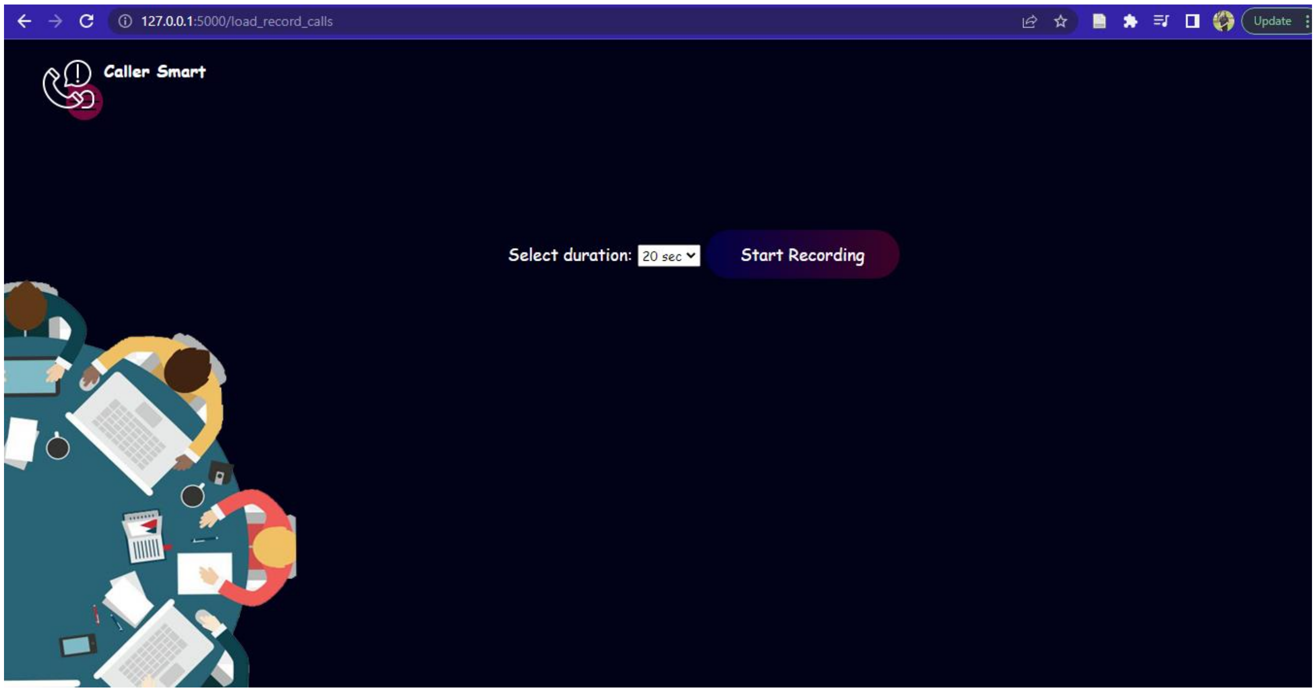
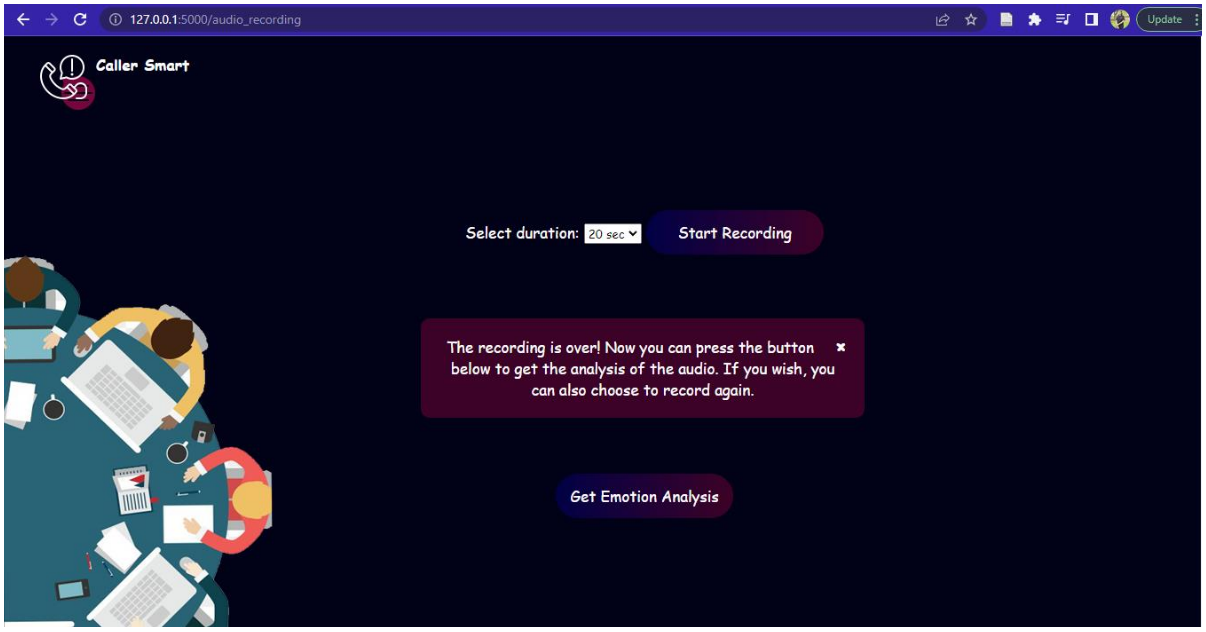


Figure 4.2.4 Call recording screen

After recording duration is over the below interface will be loaded and in that the user can see the same button and the drop-down list which was in the previous page to record if needed. It will avoid going back to the previous page unnecessarily.

There is another button in the screen with a flash message to guide the user. If the user wants to see the results of the audio that has been already recorded, he/she has to press the button “Get Emotion Analysis” and it will go to the next page after loading for some amount of time and shows the results using a better visualization effect that user can understand it well. The reason it will take some time to load that page is it has to analyze the audio and get the results by going through the model we created. The time will depend on the size of the recording which means the duration of the audio. If the audio is recorded in a small duration the results will be given in a less time.



The recording is over! Now you can press the button below to get the analysis of the audio. If you wish, you can also choose to record again.

Figure 4.2.5 Interface after complete recording

The next result page will be further discussed in the Implementation chapter.

Let's move on to the home page again and look at the second button which is "Upload Audios".

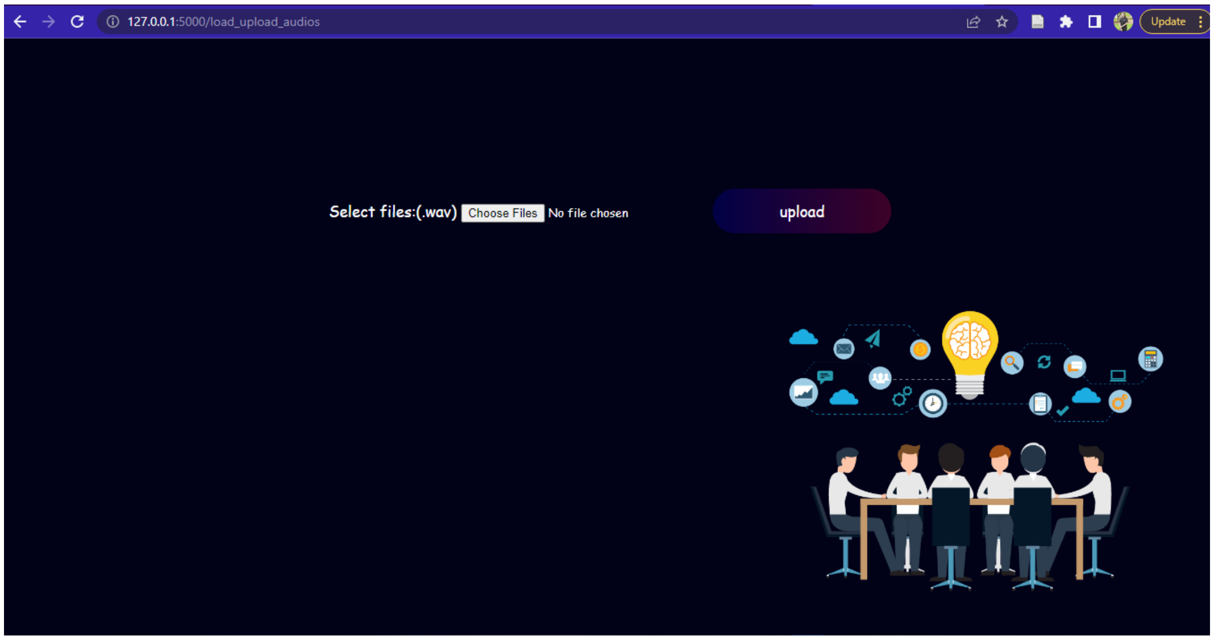


Figure 4.2.6 Interface for uploading audio file

When user click on that button it will be redirecting into the bulk upload page which user can upload audios from their databases specially the audios from unsolved cases or the pending cases. Also, they can upload any other conversations recorded with the insurance holder.

From “choose files” option, the user can browse the folder location and select the files and press “Enter”.

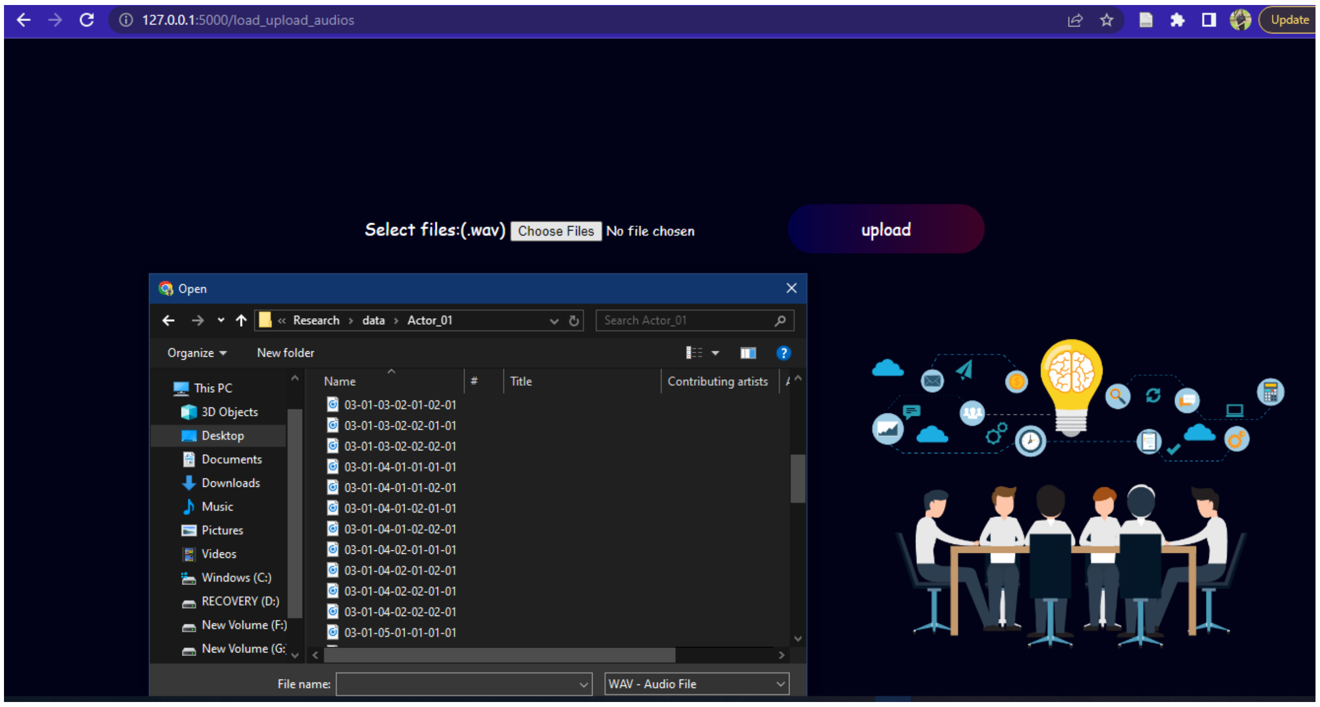


Figure 4.2.7 Browse files

Then it will indicate the number of files user selected and then he/she can click on “Upload”.

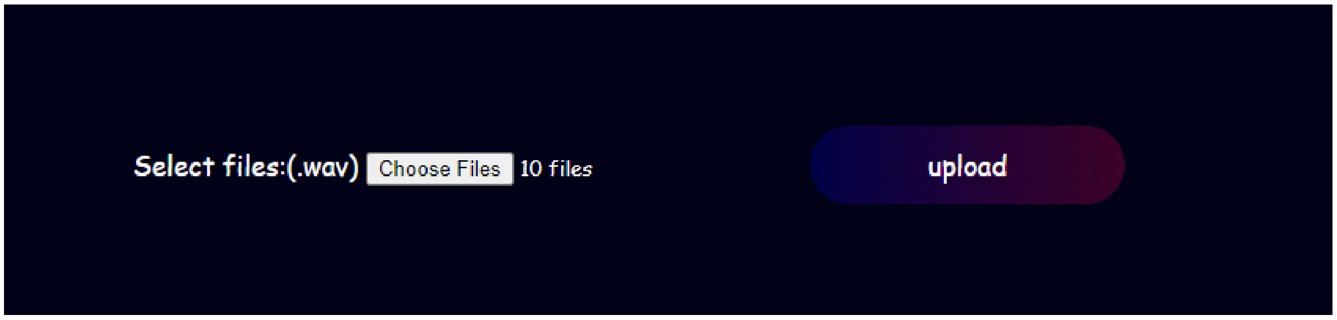


Figure 4.2.8 Indicating the number files chosen

Then it will take some time to load the page because each file has to be analyzed and generate the results for every file separately. The result page will further be discussed in the Implementation Chapter.

5. Implementation

5.1. Build a Machine Learning model

When the files are loaded, they will be decompressed and converted into a NumPy array. This array will look the same with every file type. In memory, these audios are represented as a time series, showing the amplitude at each timestep.

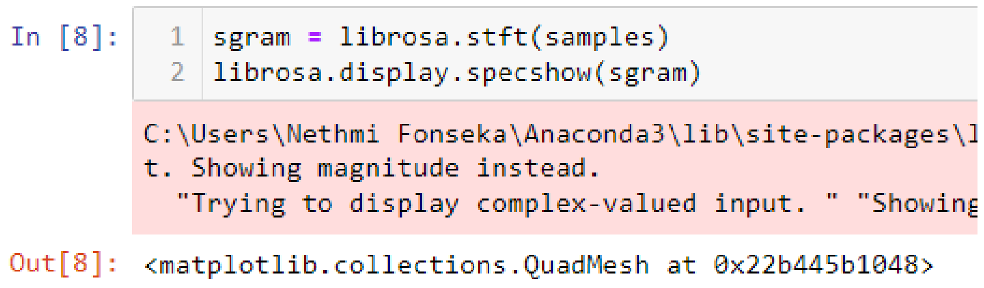
Spectrograms

It is very rare to take raw audios directly as inputs in deep learning models. The common practice is to convert those audios into a spectrogram. The spectrogram is simply a snapshot of an audio wave and since it is an image, we can input it to a CNN-based architecture which is developed for image handling.

Using Fourier Transforms, Spectrograms can be generated from sound signals. It decomposes the signal into its component frequencies and then the amplitude of every frequency can be displayed.

A Spectrogram divides the sound signal duration into smaller time components and then applies the Fourier Transform to those segments, to find the frequencies in that segment. Then it combines the Fourier Transforms for every component into a single plot.

This plot is showing Frequency (y-axis) vs Time (x-axis) and different colors for the amplitude of each frequency. Bright colors indicate the higher energy of the signal.



```
In [8]:  sgram = librosa.stft(samples)
         librosa.display.specshow(sgram)

C:\Users\Nethmi Fonseka\Anaconda3\lib\site-packages\l
t. Showing magnitude instead.
  "Trying to display complex-valued input. " "Showing

Out[8]: <matplotlib.collections.QuadMesh at 0x22b445b1048>
```

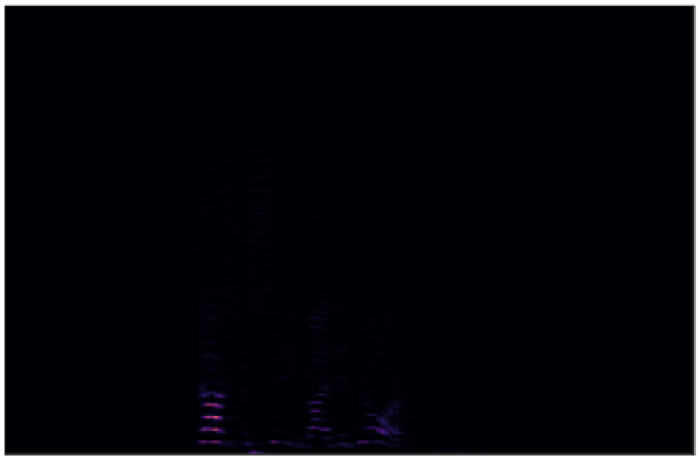


Figure 5.1.9 After applying spectrogram

Unfortunately, this spectrogram doesn't have much information to observe. This happens because of the way humans perceive sound. Normally humans can hear a narrow range of frequencies and amplitudes and these frequencies are known as 'pitch'. Humans do not perceive frequencies linearly and humans are more sensitive to differences between low frequencies and high ones.

Mel Spectrograms

A Mel Spectrogram is a bit different from a regular Spectrogram that plots Frequency vs Time. It uses the Mel Scale without using Frequency on the y-axis and Decibel Scale instead of Amplitude which indicates colors.

In [10]:
```python
import numpy as np
import matplotlib.pyplot as plt

# use the decibel scale to get the final Mel Spectrogram
mel_sgram = librosa.amplitude_to_db(mel_scale_sgram, ref=np.min)
librosa.display.specshow(mel_sgram, sr=sample_rate, x_axis='time', y_axis='mel')
plt.colorbar(format='%+2.0f dB')
```

Out[10]:
```
<matplotlib.colorbar.Colorbar at 0x2232f53a348>
```

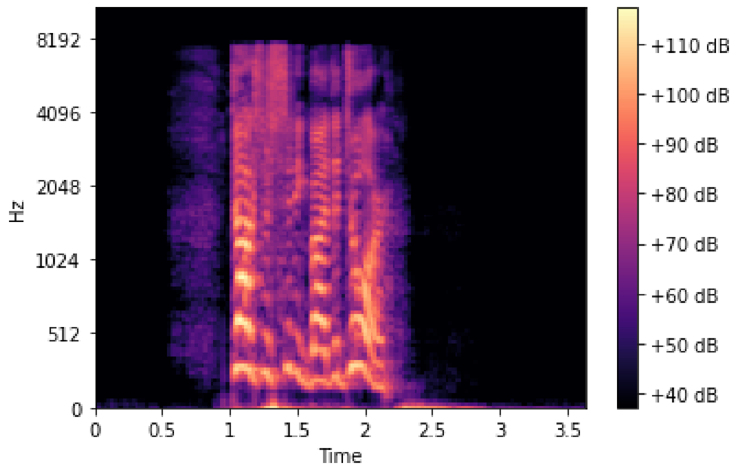
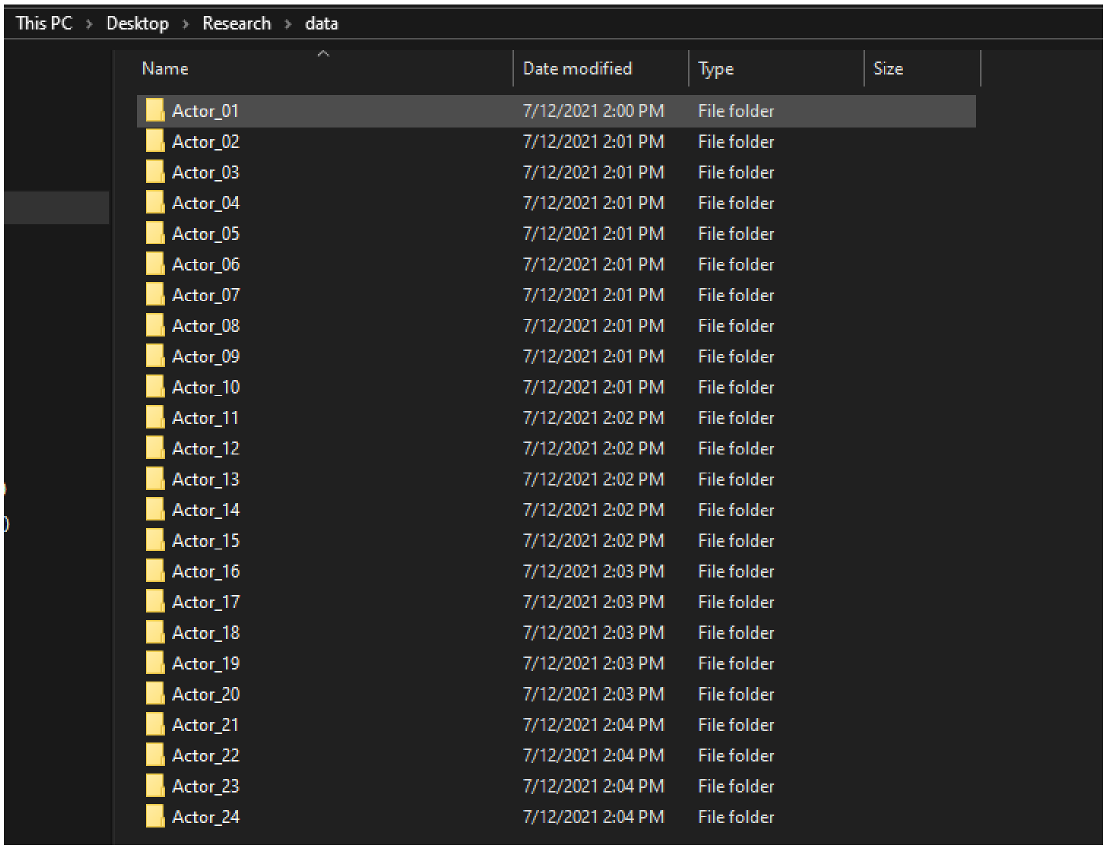


Figure 5.1.10 After modifying with Mel-spectrogram

For deep learning models, it is better to use this than using a simple Spectrogram. By modifying it with the Decibel Scale instead of Amplitude we can get a clear result. So, it’s better to use the Mel spectrogram than a simple spectrogram.

Now let’s consider the backend implementation of the program. As mentioned before the first thing is data preprocessing. Following figure shows the data I have collected from Kaggle. It has 24 actors saying some sentences with emotions.

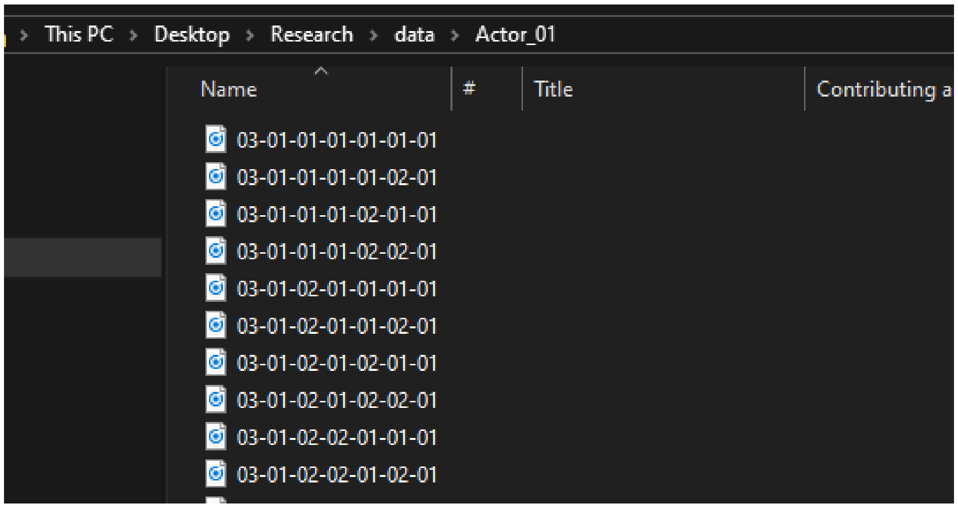


This PC › Desktop › Research › data

Name	Date modified	Type	Size
Actor_01	7/12/2021 2:00 PM	File folder	
Actor_02	7/12/2021 2:01 PM	File folder	
Actor_03	7/12/2021 2:01 PM	File folder	
Actor_04	7/12/2021 2:01 PM	File folder	
Actor_05	7/12/2021 2:01 PM	File folder	
Actor_06	7/12/2021 2:01 PM	File folder	
Actor_07	7/12/2021 2:01 PM	File folder	
Actor_08	7/12/2021 2:01 PM	File folder	
Actor_09	7/12/2021 2:01 PM	File folder	
Actor_10	7/12/2021 2:01 PM	File folder	
Actor_11	7/12/2021 2:02 PM	File folder	
Actor_12	7/12/2021 2:02 PM	File folder	
Actor_13	7/12/2021 2:02 PM	File folder	
Actor_14	7/12/2021 2:02 PM	File folder	
Actor_15	7/12/2021 2:02 PM	File folder	
Actor_16	7/12/2021 2:03 PM	File folder	
Actor_17	7/12/2021 2:03 PM	File folder	
Actor_18	7/12/2021 2:03 PM	File folder	
Actor_19	7/12/2021 2:03 PM	File folder	
Actor_20	7/12/2021 2:03 PM	File folder	
Actor_21	7/12/2021 2:04 PM	File folder	
Actor_22	7/12/2021 2:04 PM	File folder	
Actor_23	7/12/2021 2:04 PM	File folder	
Actor_24	7/12/2021 2:04 PM	File folder	

Figure 5.2.11 RAVEDESS dataset

It was preprocessed to the same sample rate of 16000 and the file name indicates the emotion including in the audio.



This PC › Desktop › Research › data › Actor_01

Name	#	Title	Contributing a
03-01-01-01-01-01-01			
03-01-01-01-01-02-01			
03-01-01-01-02-01-01			
03-01-01-01-02-02-01			
03-01-02-01-01-01-01			
03-01-02-01-01-02-01			
03-01-02-01-02-01-01			
03-01-02-01-02-02-01			
03-01-02-02-01-01-01			
03-01-02-02-01-02-01			

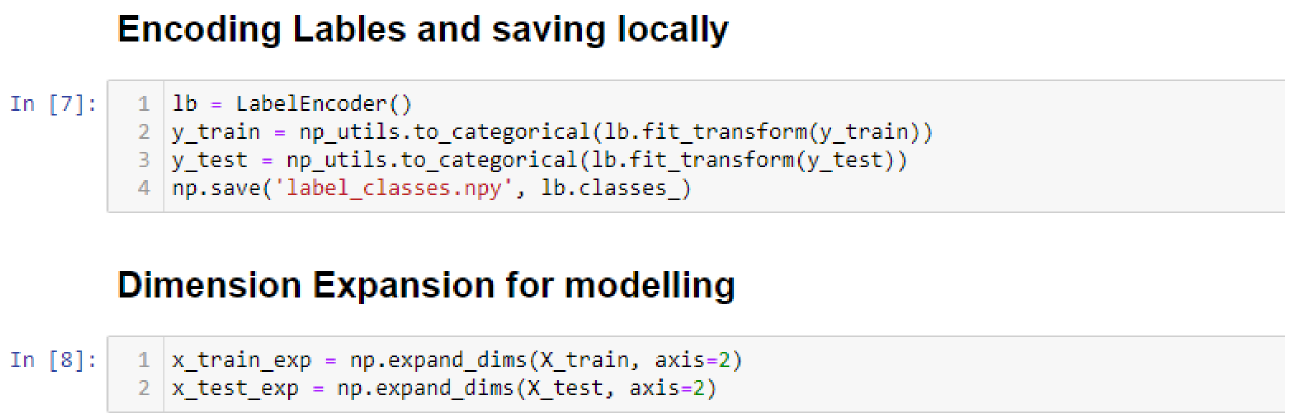
Figure 5.1.12 Audio files include in Actor files

These are some python libraries we can use for preprocessing. We can use Librosa to extract speech features, soundfile to read audio files and os, NumPy, glob, and utils packages for other functionalities like loading data, Calculations, etc.

Then to train and test the model we can use sklearn and here are the libraries we can import,

- sklearn.model_selection.train_test_splitsplit – This will split matrices or arrays into random train and test subsets. Allowed inputs are pandas data frames, lists, NumPy arrays, or SciPy-sparse matrices and we can decide the size of the test data size and train data size.
- sklearn.neural_network.MLPClassifier - This model optimizes the log-loss function using stochastic gradient descent or Limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm.
- sklearn.metrics.accuracy_score

Then I started training the model. First, I loaded the dataset with labels and then declared the training sets and testing set.



Encoding Lables and saving locally

```
In [7]:
lb = LabelEncoder()
y_train = np_utils.to_categorical(lb.fit_transform(y_train))
y_test = np_utils.to_categorical(lb.fit_transform(y_test))
np.save('label_classes.npy', lb.classes_)
```

Dimension Expansion for modelling

```
In [8]:
x_train_exp = np.expand_dims(X_train, axis=2)
x_test_exp = np.expand_dims(X_test, axis=2)
```

Figure 5.1.13 Code for encoding labels

Following figure shows the Model Architecture and let’s consider the arguments of CNN. Here I have used Conv1D because its mostly used for the time series data specially for the audio classification.

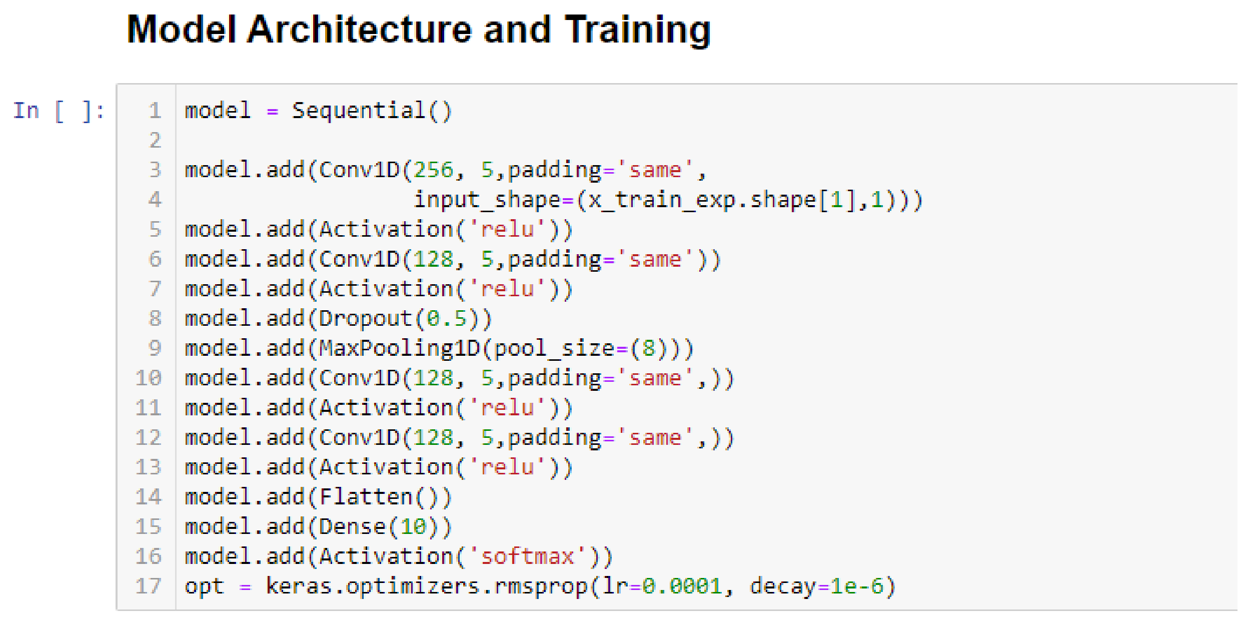
I didn’t chose Conv2D or Conv3D because in Conv2D kernel moves in two directions and it is suitable for image data. In Conv3D kernal moves in three directions and it is mostly used in 3D image data like MRIs and CT scans.

Padding has three statuses. “valid”, “same” and “casual”. First one which is “valid” indicated no padding. “same” is padding with zeros either up or down, or left or right of the input.in that case output has the same dimentions as the input. “casual” is such that current output does not depend on the next input.

Rectified linear unit or Relu is the most commonly used activation funtion in deep learning because it is easy to understand. If the funtion recieves negative input it gives zero and it the positive inputs are there, then the output returned is the value which was given as the input itself.

Dropout is regularization method that can be used to avoid overfitting from the nueral network. It randomly disables nuerons including their connections correspondingly. This forces all the neurons to learn without depending on a single neuron and generalize better.

Root mean squared propagetion ot rmsprop is used to accelarate the optimization process.It is considered as a gradient descent optimization algorithm. To improve the capability of the algorithm it decreases the number of evalutions in order to reach the optima.

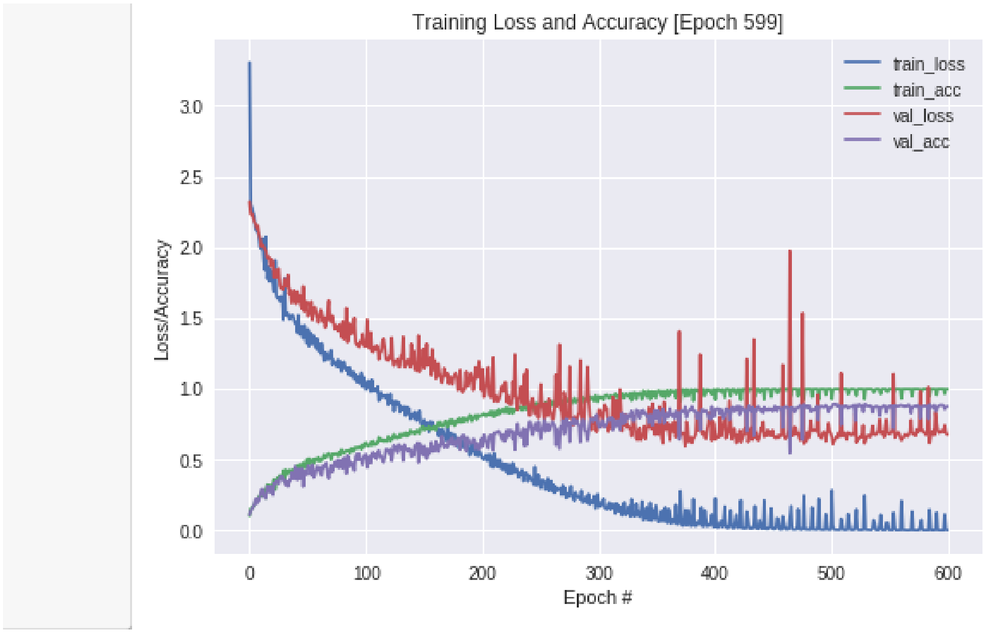


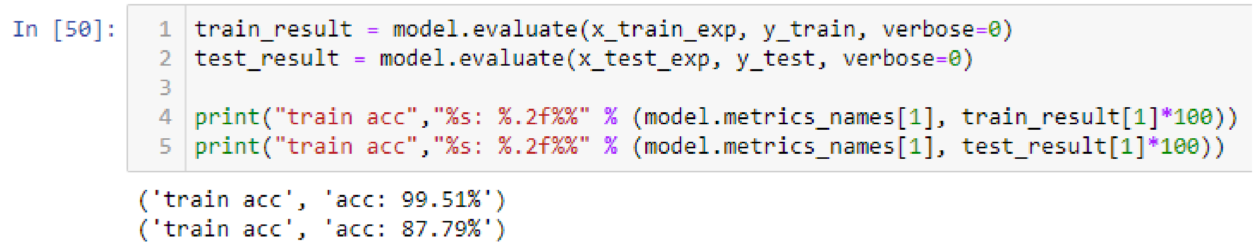
Model Architecture and Training

```
In [ ]:
model = Sequential()

model.add(Conv1D(256, 5,padding='same',
                 input_shape=(x_train_exp.shape[1],1)))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same'))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Conv1D(128, 5,padding='same',))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(10))
model.add(Activation('softmax'))
opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)
```

Figure 5.1.14 Model Architecture

After training the model with different parameters for more than 50 times, following figure shows the highest accuracies I got for the model.





```
In [50]:  train_result = model.evaluate(x_train_exp, y_train, verbose=0)
          test_result = model.evaluate(x_test_exp, y_test, verbose=0)

          print("train acc","%s: %.2f%%" % (model.metrics_names[1], train_result[1]*100))
          print("train acc","%s: %.2f%%" % (model.metrics_names[1], test_result[1]*100))

          ('train acc', 'acc: 99.51%')
          ('train acc', 'acc: 87.79%')
```

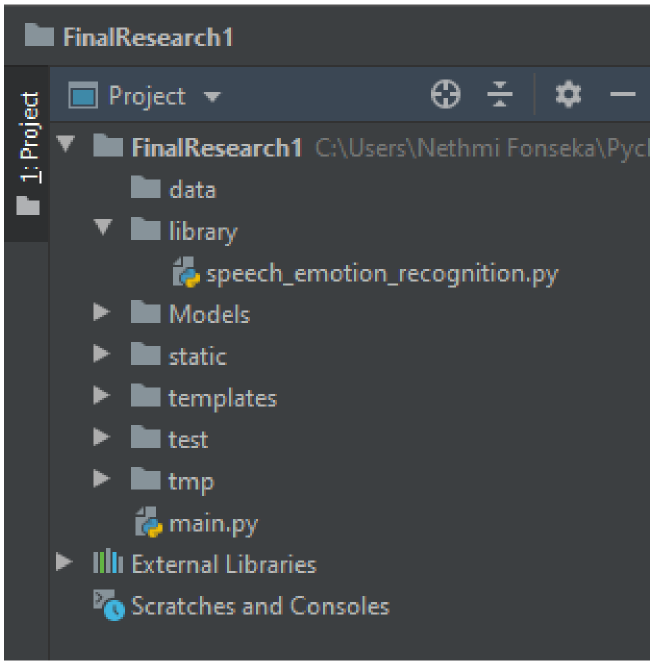
Figure 5.1.15 Accuracy of the ML model

5.2.Backend Implementation with python

Let's consider the next steps, the implementation of the python flask web application. As mentioned in the design chapter, the user interface was designed beforehand and then I started working on the python program. I chose PyCharm as an IDE, but this can be implemented using any other python supported IDEs like Visual studio code, Brackets etc. PyCharm IDE gives the privilege to work on an Anaconda environment itself.

In brief I used PyAudio library to record audios because it supports on a variety of platforms and Wave library supports wav sound format and allows to read and write wav files. Another important library is sys which is for system-specific parameters and functions. This module provides access to variables that are maintained or used by the interpreter. To create the web application I used flask, HTML, and CSS and to provide the analytics such as pie charts or bar charts, matplotlib, and mpld3 can be used. Flask was used because it allows to connect python backend with HTML and CSS.

This is the file format I used for the application. I used a “library” directory to store speech_emotion_recognition.py file which includes all the analysis functions of the program. Then the “models” folder includes the trained ML model and “static” folder includes CSS and Js files. The “templates” directory is holding the HTML files namely index.html, record.html, analysis.html, upload.html, upload_analysis.html and summary.html. “test” directory includes the test audios which I found from a GitHub project. It includes truth and lie audios of various characters. “tmp” has the temporary audio files that are recorded through the application. “main.py” includes all the basic functionalities of the application.



```
FinalResearch1 C:\Users\Nethmi Fonseka\Pyc
    data
    library
        speech_emotion_recognition.py
    Models
    static
    templates
    test
    tmp
    main.py
External Libraries
Scratches and Consoles
```

Figure 5.2.16 File Directory

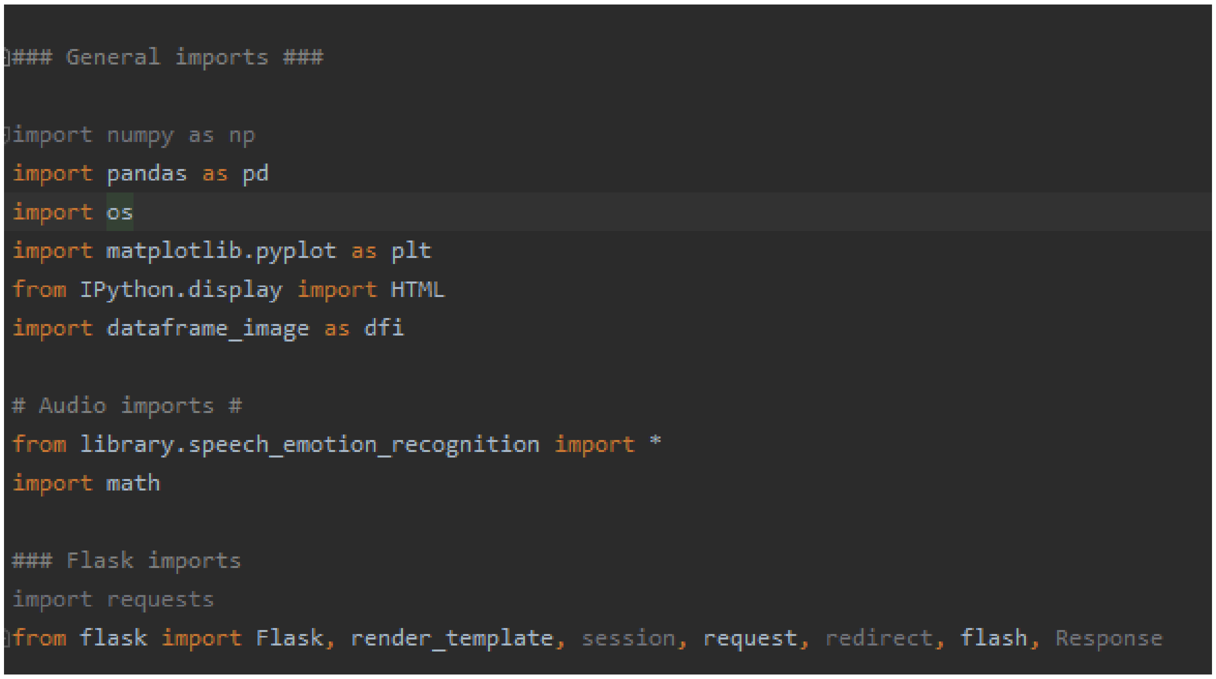
5.2.1. main.py

Now let us consider the general imports of main.py. First one is NumPy which is used to create arrays. Normally in Python we have list data type which works as a array but it is not very accurate. NumPy arrays works 50X faster than lists.

Pandas module is used to analyze data. It has functions to clean, explore, analyze and manipulate data. Here I have used pandas to create data frames. The python library os allows to interact with the operating system. This mainly helps to interact with file system.

Matplotlib.pyplot is a library that makes it as MATLAB. It includes collection of functions that will help us to create figures, plots, add labels to the plots etc. The next import is dataframe_image which is used to convert dataframes made using pandas into image formats.

Audio imports are the speech_emotion_recognition library which I created and math library. Speech_emotion_recognition library will be discussed later this chapter. So, let’s consider the math module. It is a built-in module can be used for mathematical function or tasks.



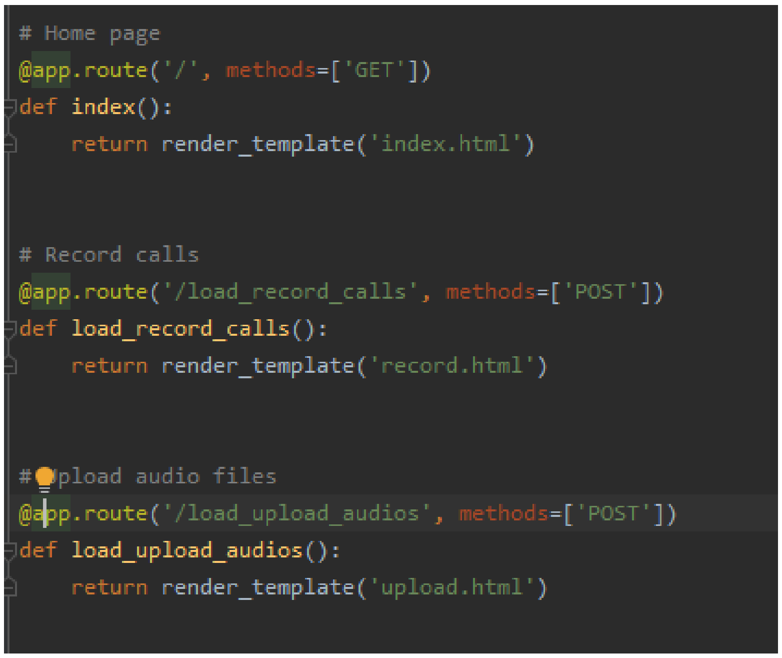
```python
### General imports ###

import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
from IPython.display import HTML
import dataframe_image as dfi

# Audio imports #
from library.speech_emotion_recognition import *
import math

### Flask imports
import requests
from flask import Flask, render_template, session, request, redirect, flash, Response
```

Figure 5.2.1.17 General imports

Next module is a very important one for the web application development. Flask has other imports including render_template, session, request, flash, redirect and response. Flask is mainly a web application framework based on Jinja2 template engine and Werkzeug WSGI toolkit. WSGI or web server gateway interface is considered as a common interface between web application and severs. Werkzeug is a toolkit that implements response, requests and utility functions which allows a web framework to be built on that toolkit. Jinja2 is a template engine which combines web template system with the data source to render a web page. Flask is also referred as microframework which helps to keep the application scalable and simple.

Following figure shows the way I managed to connect python script with the html files. Here for the Home page, I have used a GET request because the server manages to return data from that.

For Record calls page I have used POST request because it helps to send html data to the server. When the user clicks on the button "Record Calls" that html request will going to the server and act accordingly.



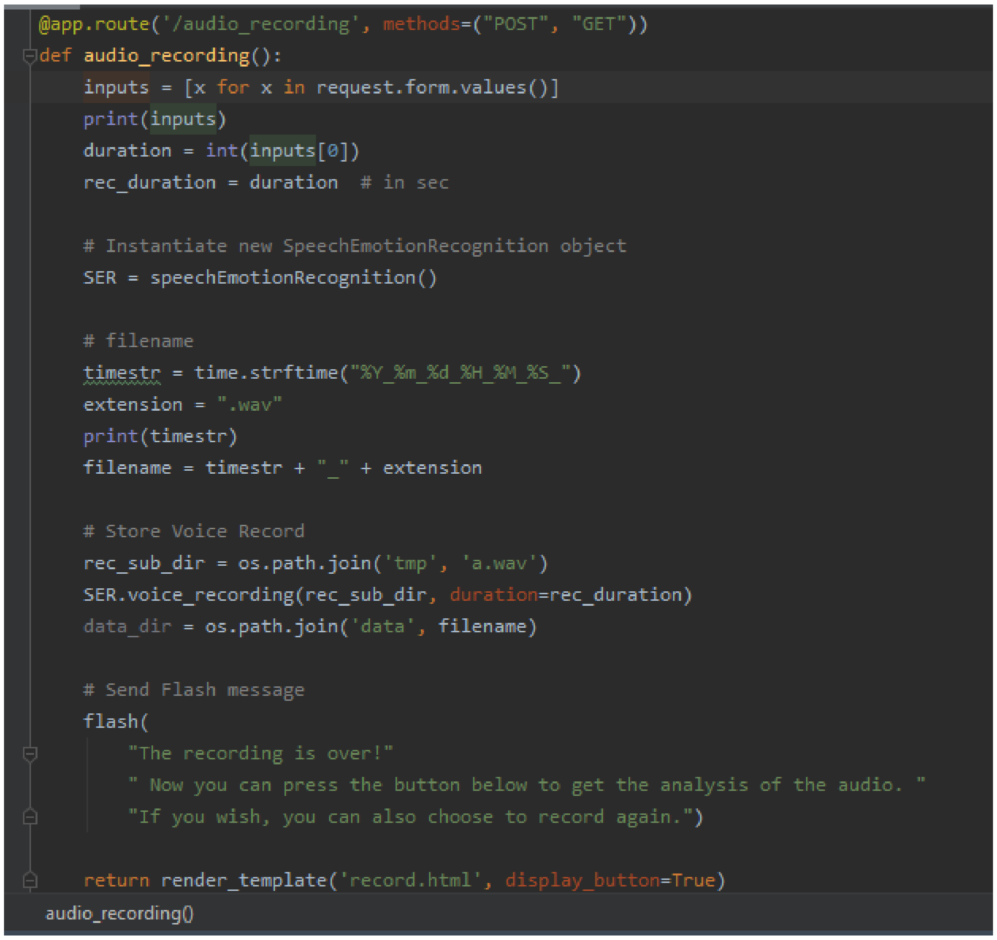
```
# Home page
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')


# Record calls
@app.route('/load_record_calls', methods=['POST'])
def load_record_calls():
    return render_template('record.html')


#Upload audio files
@app.route('/load_upload_audios', methods=['POST'])
def load_upload_audios():
    return render_template('upload.html')
```

Figure 5.2.1.18 Loading HTML pages

Next, I wrote a method to record calls. When the user going into the call recording interface, he/she can select the duration of the recording. That value has to be sent to the backed so that program runs for that particular time. To do that I have written a POST request along with the following code.

Here “inputs” variable gets the values from html form and that value is assigned to the “duration” variable. Then it is assigned to the “rec_duration” variable. Next, I have initiated a speech emotion recognition object in order to connect the speech_emotion_recognition.py library.

Then I have created a filename with the real-time information like date with year, month, day, and time, so that users can use that for further reference. There is a temporary audio file that will be stored in the tmp folder, and that audio file will be analyzed throughout the program, and it will be replaced in the next recording. When the recording is completed a flash message and another button will be displayed.

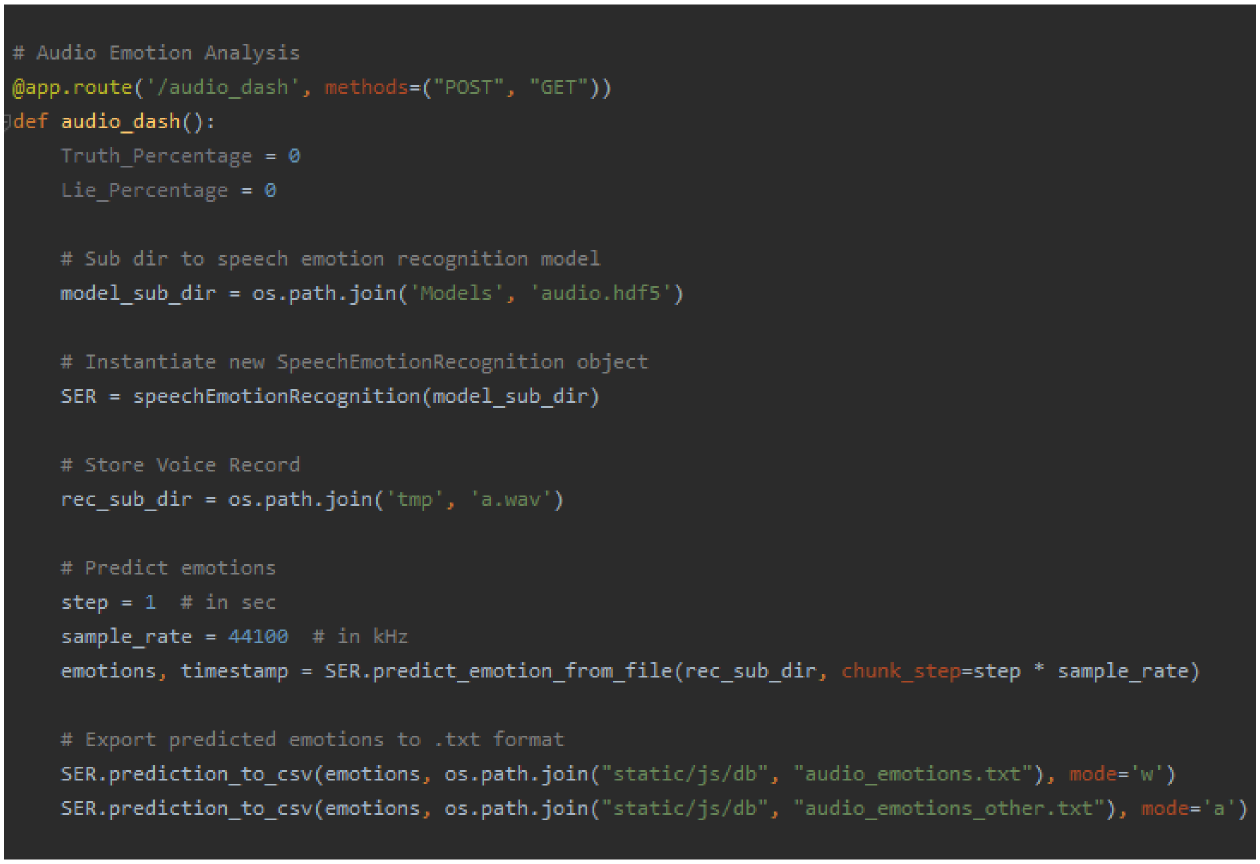


```
@app.route('/audio_recording', methods=("POST", "GET"))
def audio_recording():
    inputs = [x for x in request.form.values()]
    print(inputs)
    duration = int(inputs[0])
    rec_duration = duration  # in sec

    # Instantiate new SpeechEmotionRecognition object
    SER = speechEmotionRecognition()

    # filename
    timestr = time.strftime("%Y_%m_%d_%H_%M_%S_")
    extension = ".wav"
    print(timestr)
    filename = timestr + "_" + extension

    # Store Voice Record
    rec_sub_dir = os.path.join('tmp', 'a.wav')
    SER.voice_recording(rec_sub_dir, duration=rec_duration)
    data_dir = os.path.join('data', filename)

    # Send Flash message
    flash(
        "The recording is over!"
        " Now you can press the button below to get the analysis of the audio. "
        "If you wish, you can also choose to record again.")

    return render_template('record.html', display_button=True)
```

Figure 5.2.1.19 Audio recording

This code is to do the analysis and get the result. When the user clicks on the button Get Emotion Analysis backend of the program works like this. It will load the ML model into the previously initiated object and connect the temporary audio file into that. Then it will check the sample rate and that information will be sent to python library I created.

Then the predicted emotions which are in the csv format will be written into a text file and stored in the given location.

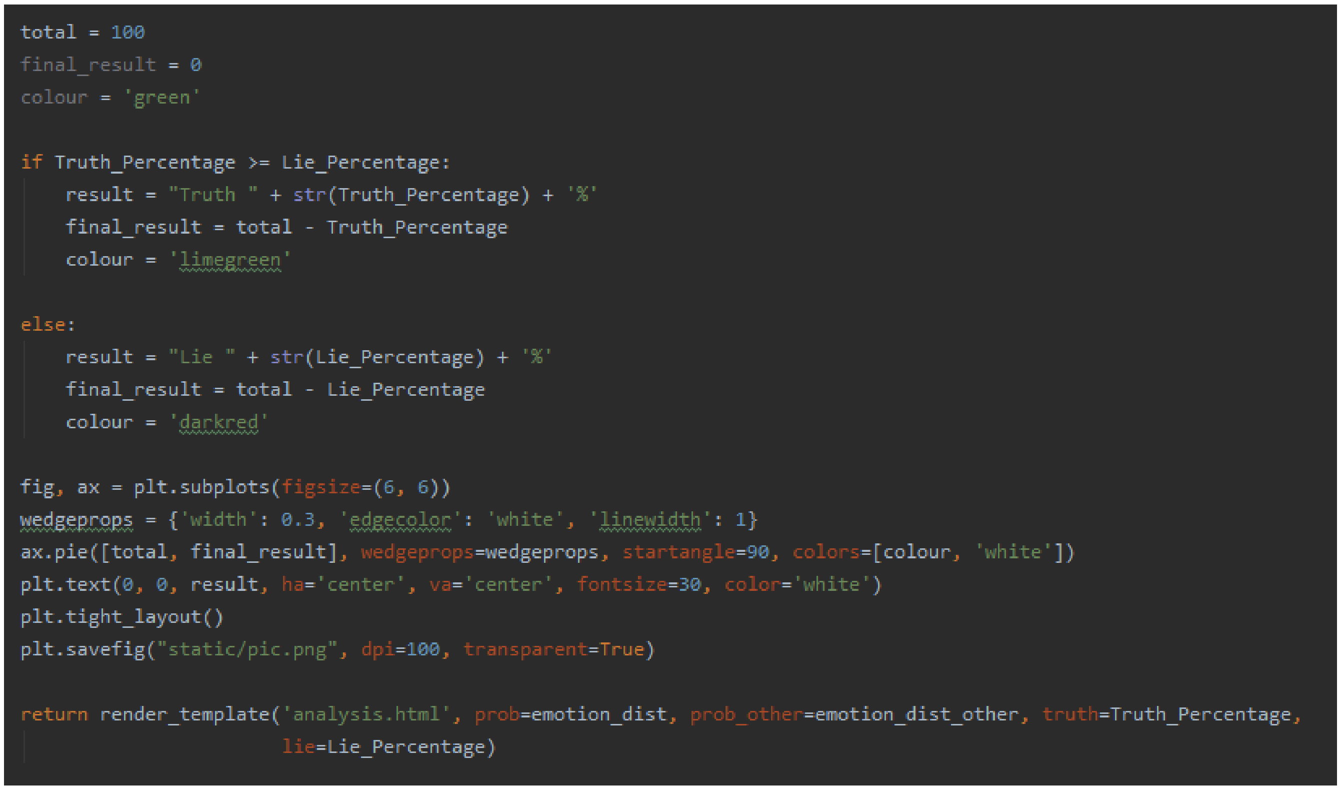


```python
# Audio Emotion Analysis
@app.route('/audio_dash', methods=("POST", "GET"))
def audio_dash():
    Truth_Percentage = 0
    Lie_Percentage = 0

    # Sub dir to speech emotion recognition model
    model_sub_dir = os.path.join('Models', 'audio.hdf5')

    # Instantiate new SpeechEmotionRecognition object
    SER = speechEmotionRecognition(model_sub_dir)

    # Store Voice Record
    rec_sub_dir = os.path.join('tmp', 'a.wav')

    # Predict emotions
    step = 1  # in sec
    sample_rate = 44100  # in kHz
    emotions, timestamp = SER.predict_emotion_from_file(rec_sub_dir, chunk_step=step * sample_rate)

    # Export predicted emotions to .txt format
    SER.prediction_to_csv(emotions, os.path.join("static/js/db", "audio_emotions.txt"), mode='w')
    SER.prediction_to_csv(emotions, os.path.join("static/js/db", "audio_emotions_other.txt"), mode='a')
```

Figure 5.2.1.20 Loading the model and getting results into a text file

Since the results are in decimal values, those are multiplied by one hundred and that csv will be converted into a dataframe. Next, both emotions and values are gotten into the same text file, and I used it to create the pie chart and other visualization methods.

The truth percentage and lie percentage will be shown in a pie chart with the colors indicating green for "truth" and red for "lie". Analysis.html page will be loaded with these results.



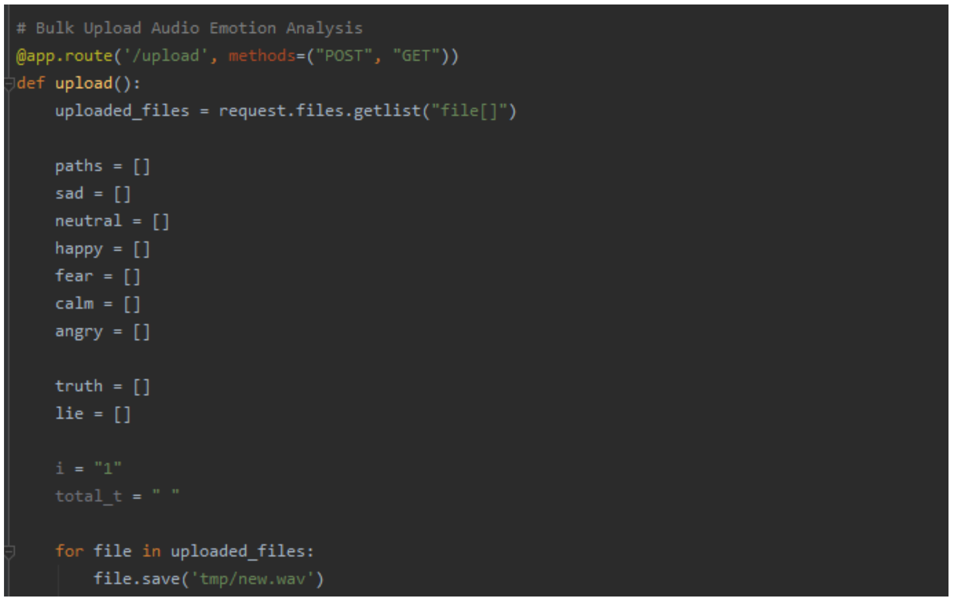
```python
total = 100
final_result = 0
colour = 'green'

if Truth_Percentage >= Lie_Percentage:
    result = "Truth " + str(Truth_Percentage) + '%'
    final_result = total - Truth_Percentage
    colour = 'limegreen'

else:
    result = "Lie " + str(Lie_Percentage) + '%'
    final_result = total - Lie_Percentage
    colour = 'darkred'

fig, ax = plt.subplots(figsize=(6, 6))
wedgeprops = {'width': 0.3, 'edgecolor': 'white', 'linewidth': 1}
ax.pie([total, final_result], wedgeprops=wedgeprops, startangle=90, colors=[colour, 'white'])
plt.text(0, 0, result, ha='center', va='center', fontsize=30, color='white')
plt.tight_layout()
plt.savefig("static/pic.png", dpi=100, transparent=True)

return render_template('analysis.html', prob=emotion_dist, prob_other=emotion_dist_other, truth=Truth_Percentage,
                       lie=Lie_Percentage)
```

Figure 5.2.1.21 Visualizing the result (python code)

Let's consider the audio upload functions and how it works. When we browse the file in the upload interface and press upload those files will be sent to the server and I have declared variable named uploaded_files to store those in a list. Also, I have initialized variables to store the paths of the audio files with its name and the emotions. This helped me to visualize the results in the html file.

From the uploaded_file list the audios are analyzed one by one using a for loop and each result will be recorded in an array.

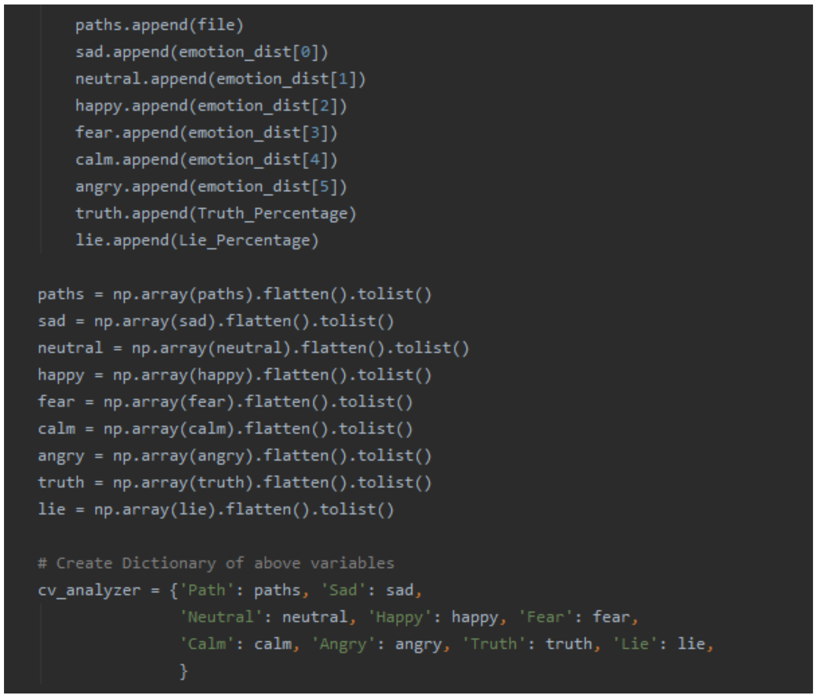
The analysis and getting the results will be happen in the same way as a recorded audio was analyzed. I used the same code because from the for loop the audios are analyzing one by one not altogether. So, code was reusable at that point and the result was temporarily stored in a text file.



```
# Bulk Upload Audio Emotion Analysis
@app.route('/upload', methods=("POST", "GET"))
def upload():
    uploaded_files = request.files.getlist("file[]")

    paths = []
    sad = []
    neutral = []
    happy = []
    fear = []
    calm = []
    angry = []

    truth = []
    lie = []

    i = "1"
    total_t = " "

    for file in uploaded_files:
        file.save('tmp/new.wav')
```

Figure 5.2.1.22 creating empty lists

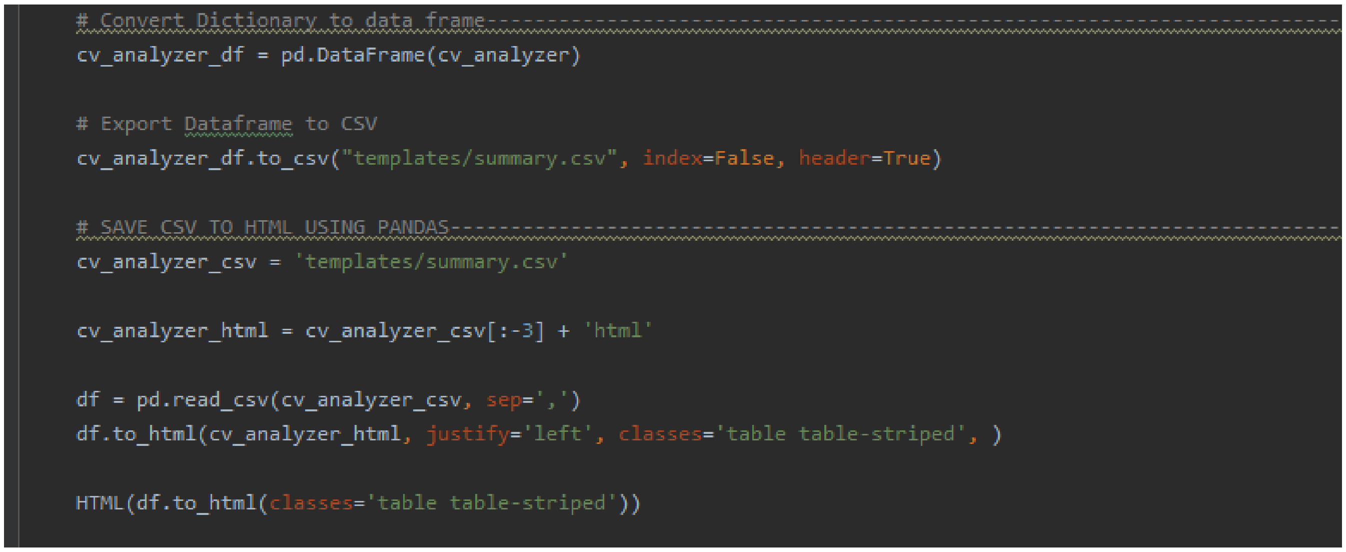
Now let us see how the data stored in an array and how it will convert into the result we want. In the following code I have appended all the results into the lists I created before and then I converted those into NumPy array so that can create a dictionary.



```
        paths.append(file)
        sad.append(emotion_dist[0])
        neutral.append(emotion_dist[1])
        happy.append(emotion_dist[2])
        fear.append(emotion_dist[3])
        calm.append(emotion_dist[4])
        angry.append(emotion_dist[5])
        truth.append(Truth_Percentage)
        lie.append(Lie_Percentage)

    paths = np.array(paths).flatten().tolist()
    sad = np.array(sad).flatten().tolist()
    neutral = np.array(neutral).flatten().tolist()
    happy = np.array(happy).flatten().tolist()
    fear = np.array(fear).flatten().tolist()
    calm = np.array(calm).flatten().tolist()
    angry = np.array(angry).flatten().tolist()
    truth = np.array(truth).flatten().tolist()
    lie = np.array(lie).flatten().tolist()

    # Create Dictionary of above variables
    cv_analyzer = {'Path': paths, 'Sad': sad,
                   'Neutral': neutral, 'Happy': happy, 'Fear': fear,
                   'Calm': calm, 'Angry': angry, 'Truth': truth, 'Lie': lie,
                   }
```

Figure 5.2.1.23 Append results into the lists

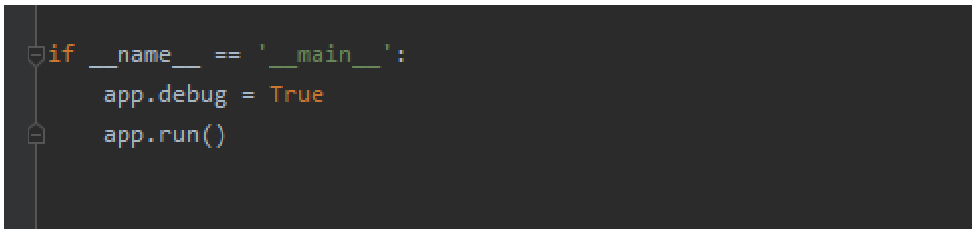
This is how I created a dataframe using that dictionary and convert it to a csv. I creted a csv because it allowed me to create a html file through it and load the results on to that page. Funtion df.to_html load the csv table into the created html file.



```python
# Convert Dictionary to data frame--------------------------------------------------------------------
cv_analyzer_df = pd.DataFrame(cv_analyzer)

# Export Dataframe to CSV
cv_analyzer_df.to_csv("templates/summary.csv", index=False, header=True)

# SAVE CSV TO HTML USING PANDAS-----------------------------------------------------------------------
cv_analyzer_csv = 'templates/summary.csv'

cv_analyzer_html = cv_analyzer_csv[:-3] + 'html'

df = pd.read_csv(cv_analyzer_csv, sep=',')
df.to_html(cv_analyzer_html, justify='left', classes='table table-striped', )

HTML(df.to_html(classes='table table-striped'))
```

Figure 5.2.1.24 Convert CSV into a HTML file

Finally I wrote the code to run the python program on the server.Here you can define whatever the host address you want. By defalut its using port 5000 and adress is http://127.0.0.1:5000.

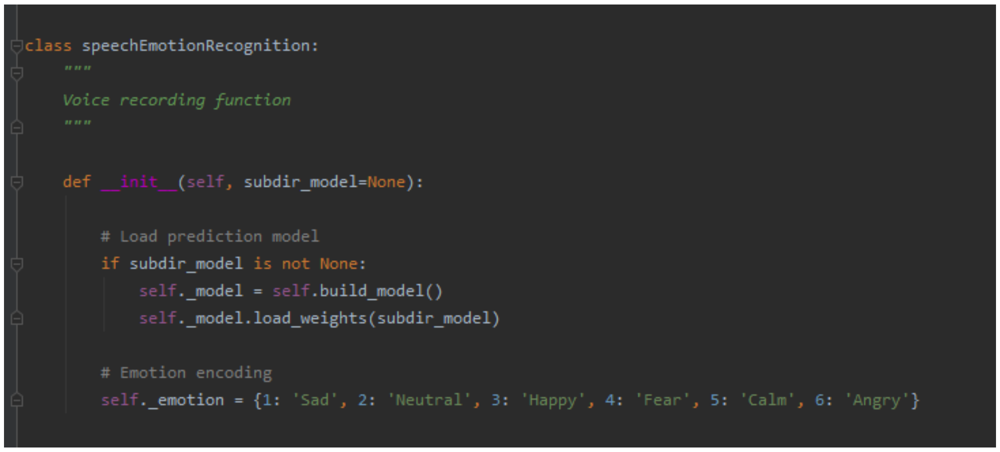


```python
if __name__ == '__main__':
    app.debug = True
    app.run()
```

Figure 5.2.1.25 Running application code

5.2.2. speech_emotion_recognition.py

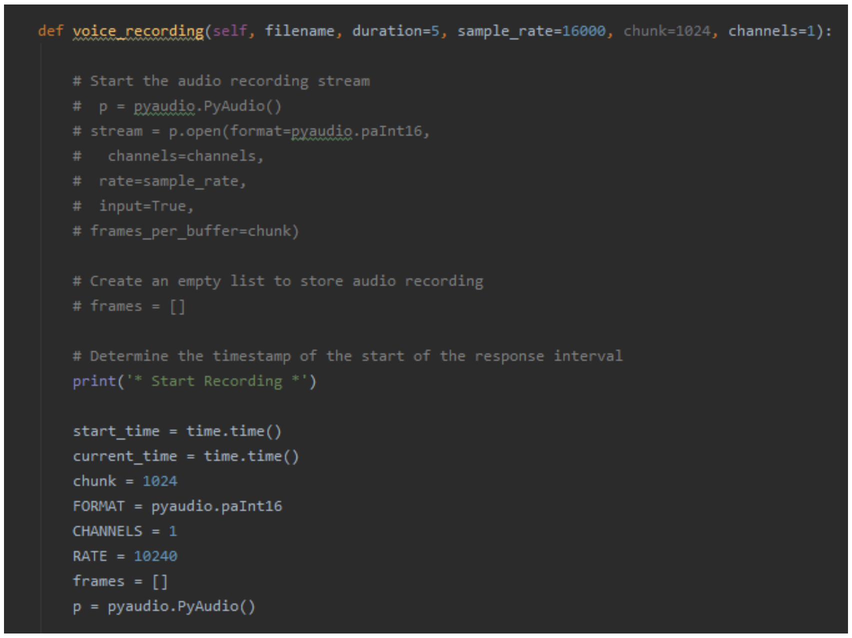
This library is designed to do the analysis of the audios that will input the system. Here I load the model and gave meaning to the numbers which will give as the output of that model. The sequence declared here is the same sequence I used to train the model.



```python
class speechEmotionRecognition:
    """
    Voice recording function
    """

    def __init__(self, subdir_model=None):

        # Load prediction model
        if subdir_model is not None:
            self._model = self.build_model()
            self._model.load_weights(subdir_model)

        # Emotion encoding
        self._emotion = {1: 'Sad', 2: 'Neutral', 3: 'Happy', 4: 'Fear', 5: 'Calm', 6: 'Angry'}
```

Figure 5.2.2.26 Initializing the model outputs

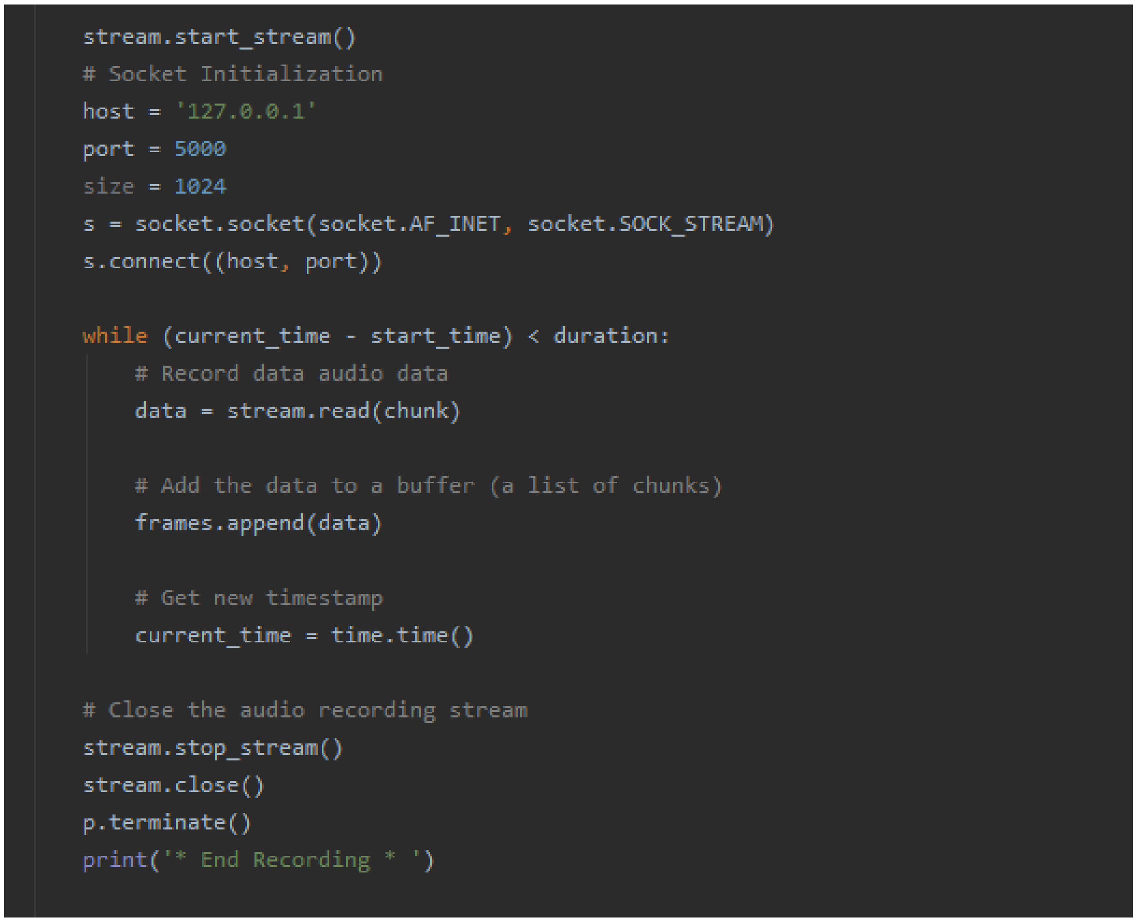
All the recordings are happening through the pyaudio library and when the recording in process I can start recording printed on the terminal. It will record on the given sample rate and here I'm using 16000.



```python
def voice_recording(self, filename, duration=5, sample_rate=16000, chunk=1024, channels=1):

    # Start the audio recording stream
    #  p = pyaudio.PyAudio()
    # stream = p.open(format=pyaudio.paInt16,
    #   channels=channels,
    #  rate=sample_rate,
    #  input=True,
    # frames_per_buffer=chunk)

    # Create an empty list to store audio recording
    # frames = []

    # Determine the timestamp of the start of the response interval
    print('* Start Recording *')

    start_time = time.time()
    current_time = time.time()
    chunk = 1024
    FORMAT = pyaudio.paInt16
    CHANNELS = 1
    RATE = 10240
    frames = []
    p = pyaudio.PyAudio()
```

Figure 5.2.2.27 Voice recording code

Since the program will be running on port five thousand, I had to give it here as variable including the host. When the time is up audio recording stream is closing its own.

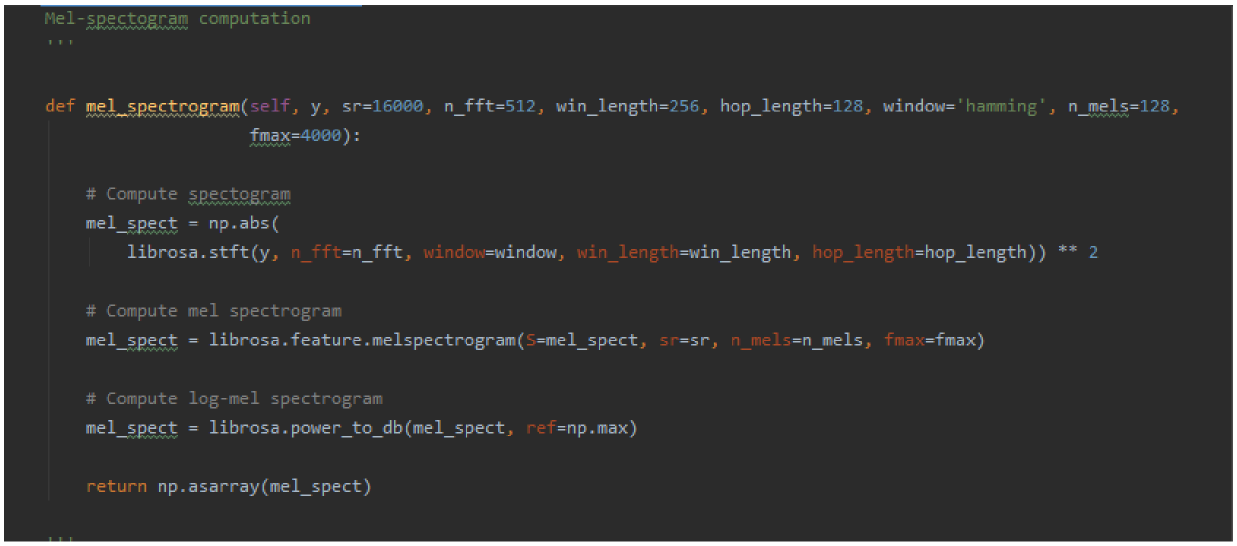


```python
stream.start_stream()
# Socket Initialization
host = '127.0.0.1'
port = 5000
size = 1024
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

while (current_time - start_time) < duration:
    # Record data audio data
    data = stream.read(chunk)

    # Add the data to a buffer (a list of chunks)
    frames.append(data)

    # Get new timestamp
    current_time = time.time()

# Close the audio recording stream
stream.stop_stream()
stream.close()
p.terminate()
print('* End Recording * ')
```

Figure 5.2.2.28 Close the audio recording stream

As already mentioned in the previous chapters I chose Mel spectrogram and we are dealing with time series data. That is why I have used librosa stft for further analysis of the audios. Stft is Short-Time Fourier Transform that represents a signal in the domain of time frequency. It computes the Discrete Fourier Transforms without overlapping the windows.

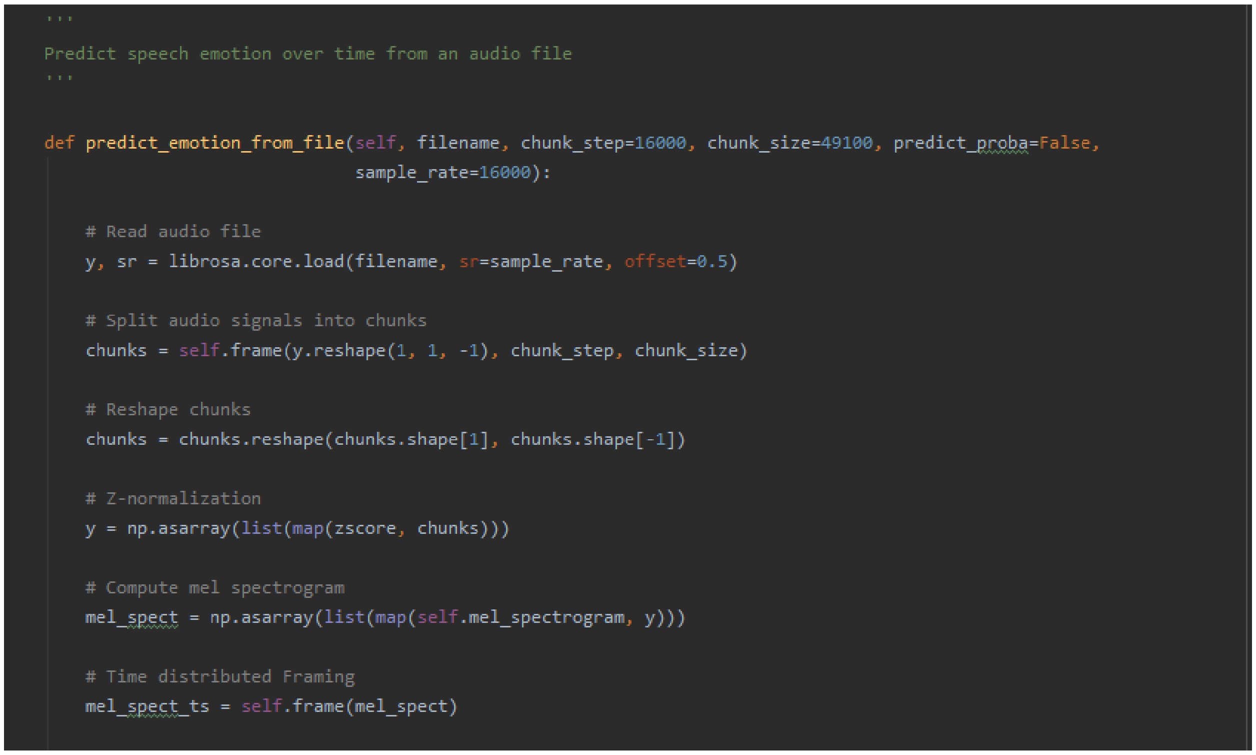
The parameters are y which reprensts the input signal, n_fft which is an integer greater than zero and represents the length of the signal after padding with 0s. In speech recognition or processing the recommended value for n_fft is 512. Another parameter is hop_length is the no of audio samples between the STFT adjacent columns.



```
Mel-spectogram computation
'''

def mel_spectrogram(self, y, sr=16000, n_fft=512, win_length=256, hop_length=128, window='hamming', n_mels=128,
                    fmax=4000):

    # Compute spectogram
    mel_spect = np.abs(
        librosa.stft(y, n_fft=n_fft, window=window, win_length=win_length, hop_length=hop_length)) ** 2

    # Compute mel spectrogram
    mel_spect = librosa.feature.melspectrogram(S=mel_spect, sr=sr, n_mels=n_mels, fmax=fmax)

    # Compute log-mel spectrogram
    mel_spect = librosa.power_to_db(mel_spect, ref=np.max)

    return np.asarray(mel_spect)

'''
```

Figure 5.2.2.29 Using librosa.stft

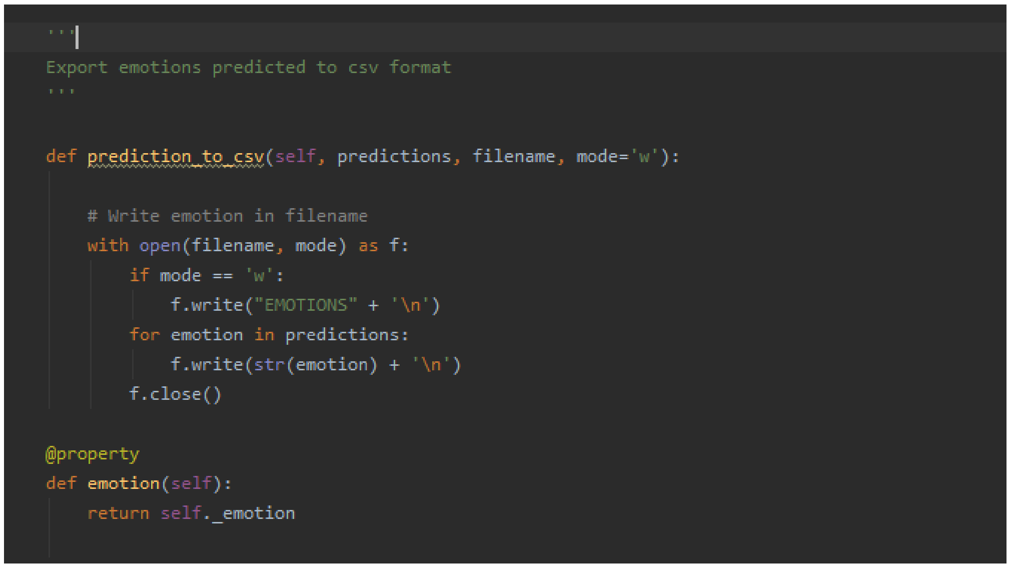
Then it reads the audio file that we stored in a temporary location and checks the sample rate. If the audio has the predifined sample rate it will show an error, otherwise it will do the analysis and give the result.



```
'''
Predict speech emotion over time from an audio file
'''

def predict_emotion_from_file(self, filename, chunk_step=16000, chunk_size=49100, predict_proba=False,
                              sample_rate=16000):

    # Read audio file
    y, sr = librosa.core.load(filename, sr=sample_rate, offset=0.5)

    # Split audio signals into chunks
    chunks = self.frame(y.reshape(1, 1, -1), chunk_step, chunk_size)

    # Reshape chunks
    chunks = chunks.reshape(chunks.shape[1], chunks.shape[-1])

    # Z-normalization
    y = np.asarray(list(map(zscore, chunks)))

    # Compute mel spectrogram
    mel_spect = np.asarray(list(map(self.mel_spectrogram, y)))

    # Time distributed Framing
    mel_spect_ts = self.frame(mel_spect)
```

Figure 5.2.2.30 Predicting results

Finally, the predicted emotions will be into a csv and that will be accessed from main.py.



```
'''
Export emotions predicted to csv format
'''

def prediction_to_csv(self, predictions, filename, mode='w'):

    # Write emotion in filename
    with open(filename, mode) as f:
        if mode == 'w':
            f.write("EMOTIONS" + '\n')
        for emotion in predictions:
            f.write(str(emotion) + '\n')
        f.close()

@property
def emotion(self):
    return self._emotion
```

Figure 5.2.2.31 Write predicted results into a CSV

So, the technology stack used in the application is,

Programming (Python, HTML, CSS, JavaScript)

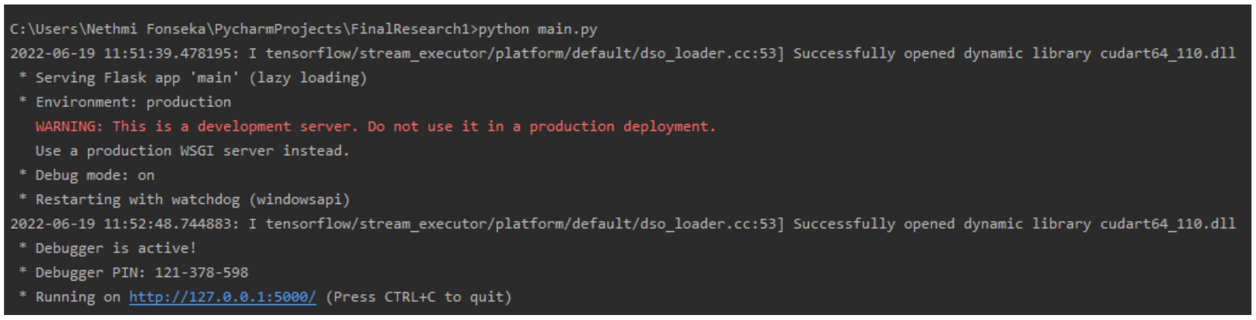
Frameworks (Flask, TensorFlow)

Environments (Anaconda, Google Colab)

and as for deployment, we can use Microsoft Azure or Amazon Web Services and for the database development, we can use MySQL, Firebase Databases, etc.

5.3. Run the program

To run the program either I can use command prompt or terminal of the IDE. After getting into the project directory just have run the command python main.py. Then it will show some information and finally the link of the running application. Either you can copy paste it into the browser or you can just click on the link, and it will open on the browser.



```
C:\Users\Nethmi Fonseka\PycharmProjects\FinalResearch1>python main.py
2022-06-19 11:51:39.478195: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library cudart64_110.dll
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with watchdog (windowsapi)
2022-06-19 11:52:48.744883: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library cudart64_110.dll
 * Debugger is active!
 * Debugger PIN: 121-378-598
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 5.2.3.32 Way the program is running

Lets consider the results pages after completing the implentation of the program.

This is result page user can see when a audio is recorded through the application and click on the button “Get Emotion Analysis”. Here user can get an idea about the emotions of the user and also the truth and lie percentage.

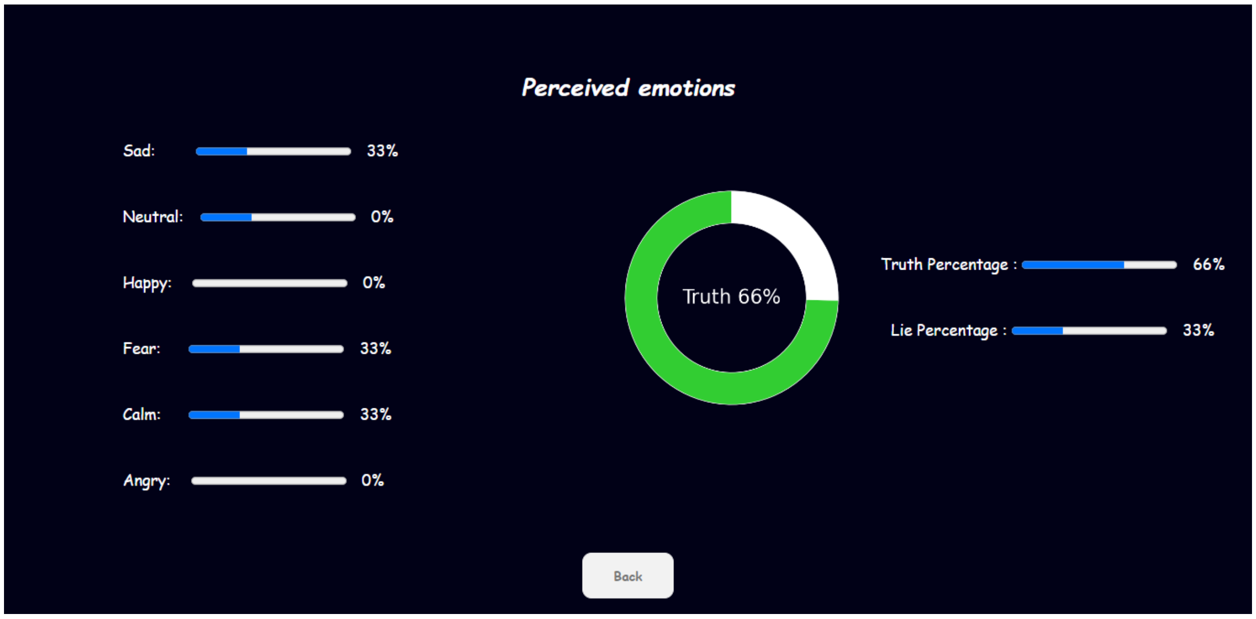
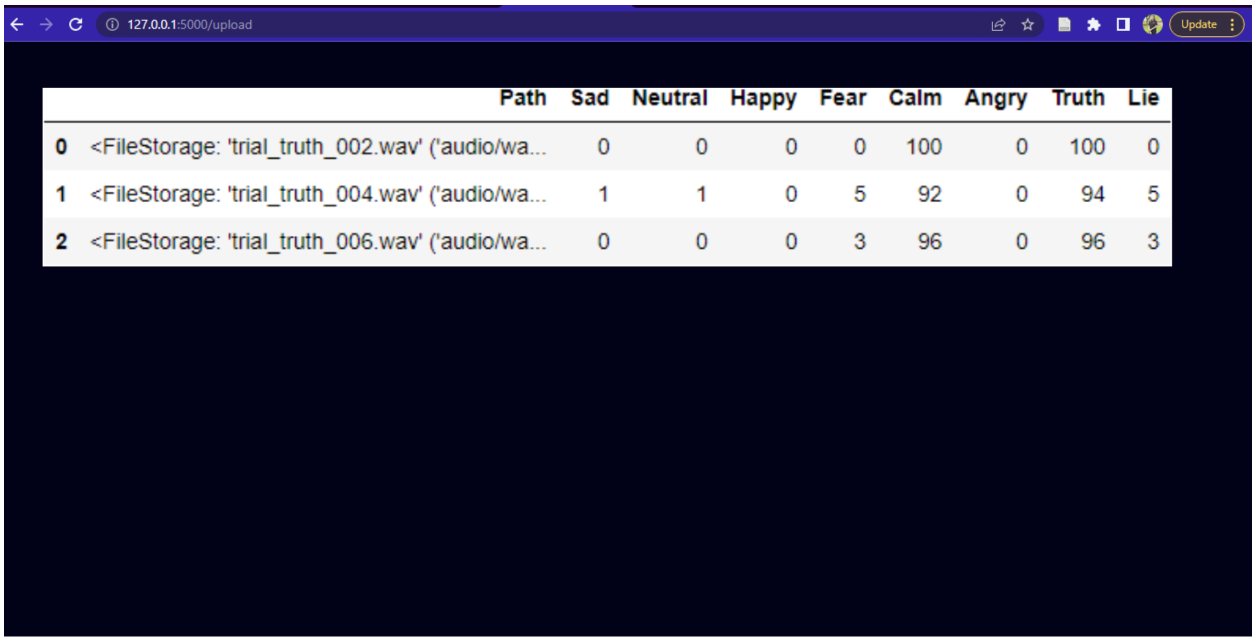


Figure 5.2.3.33 Audio recording result page

Samething happens when user goes to the upload files page. It analyze all the files one by one but gives the result alltogether. Here also user can get idea about the emotions and truth , lie percentages according the audios gives as a input.



	Path	Sad	Neutral	Happy	Fear	Calm	Angry	Truth	Lie
0	<FileStorage: 'trial_truth_002.wav' ('audio/wa...	0	0	0	0	100	0	100	0
1	<FileStorage: 'trial_truth_004.wav' ('audio/wa...	1	1	0	5	92	0	94	5
2	<FileStorage: 'trial_truth_006.wav' ('audio/wa...	0	0	0	3	96	0	96	3

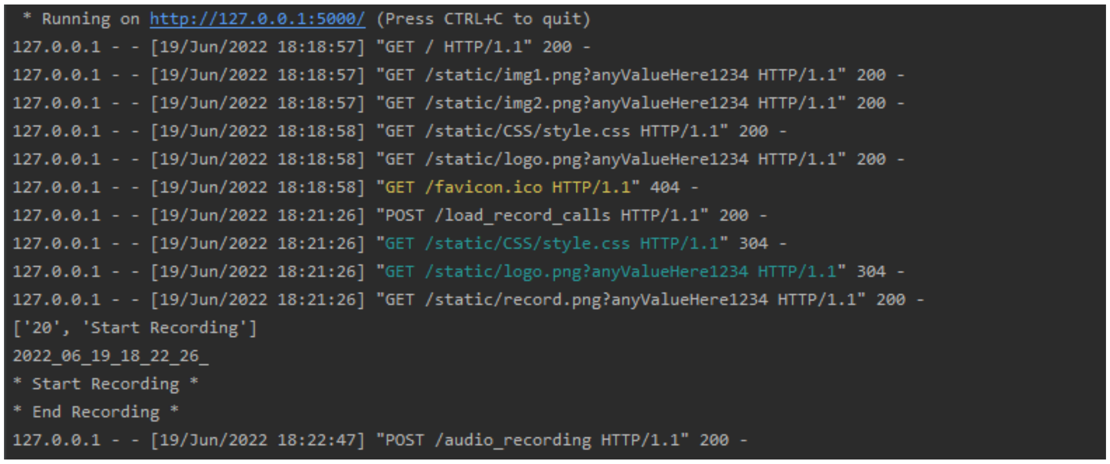
Figure 5.2.3.34 Audio uploading result page

6. Testing

The optimized system was validated using available data with different techniques. I mainly focused on static and dynamic testing for validation. Under the static validation, we have done an informal review, code walkthrough, inspection, technical review, and static code review. Initially tested the system with model accuracy which can be done by the python libraries and the testing data set that was partitioned from the data collected. Secondly, after implementing the application I was able to test the application with audios I collected from a GitHub project which included some truth audios and lie audios. I tested the application with those audios, and it also gave better results, and the performance of the application is also could be measured from that. The rest of the testing approaches can be decided with the development of the system, e.g., Regression testing, Performance testing, Sanity testing, etc.

For an example lets say we are recording a audio using the application. Now we can see all the logs which were successfully loaded and which were not. Here HTTP 200 indicates OK or success.

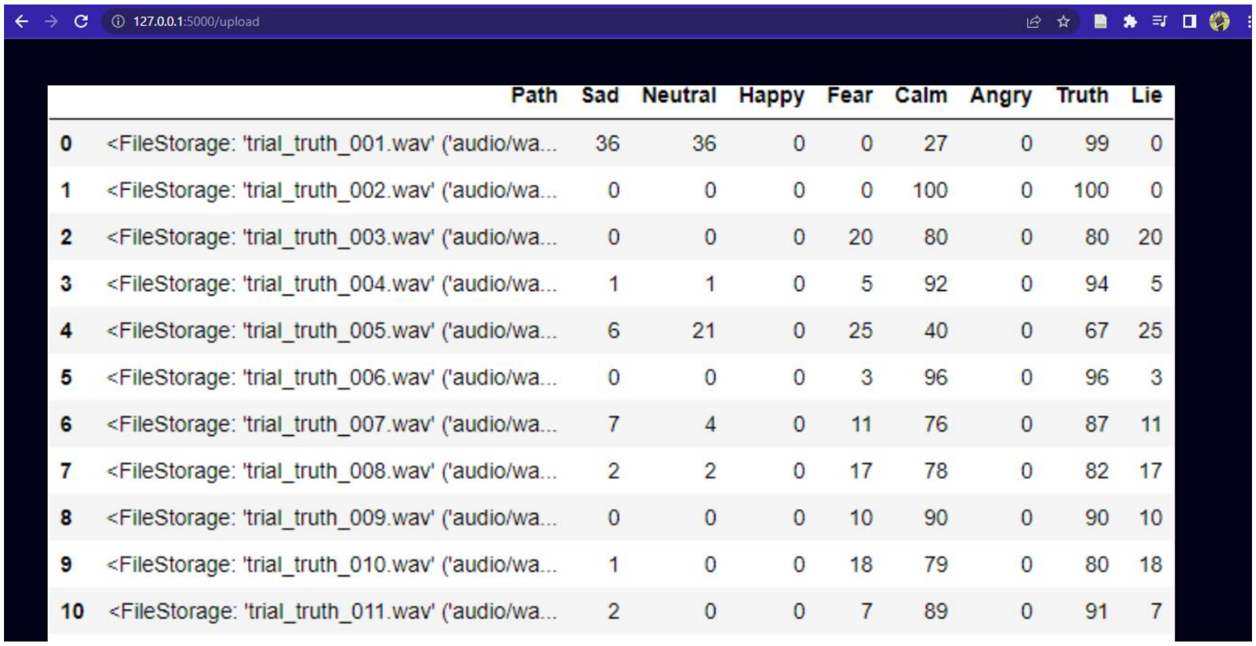
HTTP 304 means request hasn’t been updated from the last time. Like this I tested the application to find any error.



```
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [19/Jun/2022 18:18:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2022 18:18:57] "GET /static/img1.png?anyValueHere1234 HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2022 18:18:57] "GET /static/img2.png?anyValueHere1234 HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2022 18:18:58] "GET /static/CSS/style.css HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2022 18:18:58] "GET /static/logo.png?anyValueHere1234 HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2022 18:18:58] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [19/Jun/2022 18:21:26] "POST /load_record_calls HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2022 18:21:26] "GET /static/CSS/style.css HTTP/1.1" 304 -
127.0.0.1 - - [19/Jun/2022 18:21:26] "GET /static/logo.png?anyValueHere1234 HTTP/1.1" 304 -
127.0.0.1 - - [19/Jun/2022 18:21:26] "GET /static/record.png?anyValueHere1234 HTTP/1.1" 200 -
['20', 'Start Recording']
2022_06_19_18_22_26_
* Start Recording *
* End Recording *
127.0.0.1 - - [19/Jun/2022 18:22:47] "POST /audio_recording HTTP/1.1" 200 -
```

Figure 6.35 Using terminal to find the errors

Another way I tested the application by using the data I found from a github project. It has these audios that people are telling truth and lies. So I got those data and converted the into 16000 sample rate and then upload and checked whether application is working accordignly.



127.0.0.1:5000/upload

	Path	Sad	Neutral	Happy	Fear	Calm	Angry	Truth	Lie
0	<FileStorage: 'trial_truth_001.wav' ('audio/wa...	36	36	0	0	27	0	99	0
1	<FileStorage: 'trial_truth_002.wav' ('audio/wa...	0	0	0	0	100	0	100	0
2	<FileStorage: 'trial_truth_003.wav' ('audio/wa...	0	0	0	20	80	0	80	20
3	<FileStorage: 'trial_truth_004.wav' ('audio/wa...	1	1	0	5	92	0	94	5
4	<FileStorage: 'trial_truth_005.wav' ('audio/wa...	6	21	0	25	40	0	67	25
5	<FileStorage: 'trial_truth_006.wav' ('audio/wa...	0	0	0	3	96	0	96	3
6	<FileStorage: 'trial_truth_007.wav' ('audio/wa...	7	4	0	11	76	0	87	11
7	<FileStorage: 'trial_truth_008.wav' ('audio/wa...	2	2	0	17	78	0	82	17
8	<FileStorage: 'trial_truth_009.wav' ('audio/wa...	0	0	0	10	90	0	90	10
9	<FileStorage: 'trial_truth_010.wav' ('audio/wa...	1	0	0	18	79	0	80	18
10	<FileStorage: 'trial_truth_011.wav' ('audio/wa...	2	0	0	7	89	0	91	7

Figure 6.36 Trial audio results

The best approach is to test the application in the industry with real-time cases and the data that is already classified by the insurance companies. Since they already have the data records of their solved cases, using this system we can test and validate the accuracy and decide how reliable and efficient it is.

7. Deployment

We have many options to deploy the Flask Web application and then there are a few most commonly used methods.

Currently using limited resources proposed system can be deployed as a flask web application in the local environment. To do that we need to ensure that all the dependencies are installed and for better results, it's good to work on an Anaconda environment that has support from a CUDA-enabled graphical processing unit.

But to scale up the solution we can use either amazon web services or Microsoft Azure cloud services. To use both services there should a subscription and after deploying we can use it anywhere anytime.

7.1.Deploying Flask App on Heroku

Basic software requirements for this include Python, pip, Heroku CLI, and Git. After installing these the first step is to create a virtual environment with pipenv and install flask and Gunicorn. Then Create a Procfile and runtime.txt. Then after login into the Heroku CLI, we have to push the code from the local to the Heroku remote.

7.2.Deploying Flask App on AWS or Azure

The second option is to use AWS or Azure. Here is the explanation of deploying on AWS.

First, we need an AWS account and must install Docker, AWS Command Line Interface (CLI) tool, and LightSail Control (LightSailctl) plugin on the system as prerequisites. Then we have to create the flask application which is already we have in the system. Then we must create requirements.txt to specify the python packages required for the application. Then we have to create a new file named Dockerfile. At this point, the director contains app.py, Dockerfile, and requirements.txt.

The next step is to build and test the container image locally. To do that we have to build the container using Docker. Once the container build is done, test the Flask application locally by running the container. Then we must create the Lightsail container service using the AWS CLI, and after that push your local container image to the new container service using the Lightsail control (lightsailctl) plugin. The scale parameters and power indicate the capacity of the container service. Small capacity is required for a minimal flask app.

- Using the create-container-service command we can create the Lightsail container service. The output of that command indicates "PENDING" as the state of the new service.
- To monitor the state of the container as it is being created, we can use the get-container-services command. Then before going to the next step, wait until the container service state changes to "ACTIVE". After a few minutes container service should become active.
- Then push the application container to Lightsail with the push-container-image command.

The next step is to deploy the container. To do that we must complete the following steps to create public endpoint configuration JSON files and then deploy the container image to the container service.

- First create a new file as containers.json edit the file to replace X in:flask-service.flask-container.X and save it. (Settings of the containers that will be launched on the container service are described by the containers.json file.)

- Then create another new file named public-endpoint.json. Edit that file and save it. (This file describes the settings of the container service's public endpoints. These settings are only required for the services that require public access.)

After creating these two files, the project directory should have 5 files including app.py, cintainers.json, Dockerfile, public-endpoint.json, and requirements.txt.

- Then we can use AWS CLI create-container-service-deployment command to deploy the container to the container service. The output of this command indicates that the state of the container service as "DEPLOYING".
- Next use the get-container-services command to return the endpoint URL and to monitor the state of the container.
- After a few minutes container service state changes to the "RUNNING" state and now redirect to the URL and confirms whether the service running properly.

In this manner, we can successfully deploy a containerized Flask application using Amazon Lightsail containers.

8. Project Management

The first task was to decide on a topic that is more related to Data Science and also along with the machine learning concepts. Since python is more familiar to me, I research the topics and went through articles and tutorials that are dealing with a larger amount of data. As a development project I choose this topic because of the experience I gained from my internship and some projects I was involved in during my academics.

After getting a brief idea of the concepts I started working on the proposal and I sent a draft to the lectures. After they approved my proposal, I started working on the data collection procedure and did the preprocessing steps. The supervisor guided me throughout the process and gave some tasks to complete biweekly.

Then training the model with different parameters took me about a month and I was able to observe differences in the accuracy. First, there was lesser accuracy but later I was able to get better accuracy in the process.

Then the implementation of the application took me about three months with all the features I mentioned in my proposal. It took another couple of weeks for the UI implementation and along with all these works I started working on the thesis as well. Finally, I was able to deploy this application on a local server.

Table 1 Time Frame

Tasks	2021							2022					
	June	July	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun
Deciding a topic and Reading Research papers	■												
preliminary discussions with one of the supervisors		■											
Project planning	■	■											
Preparing the project proposal and submission	■	■											
Preparing proposal defending presentation			■										
Data Collection		■	■										
Data Preprocessing and feature extraction			■										
Training and testing the model				■									
Creating the main functionalities of the application				■	■	■							
Testing the completed work						■			■	■			

Creating web applications and designing the user interface						■	■						
Adding some more features to the program							■						
Preparing the thesis									■	■		■	■
Complete the Web Application							■	■			■		
Deployment									■	■			■

9. Evaluation - feedback from users/experts' reviews

To evaluate the system, first, we can use the model accuracy. Python library provides the capability of finding the accuracy of a model as a percentage so we can get an idea about the quality of the system.

In another case, if it is possible to connect with an insurance company, they have data on the solved cases. For example, they might be having some data related to the solved cases and prove that as fraud. So, from those audios, we can test the system and get an evaluation.

10. Conclusion

Since the voice can be recorded from the system there can be privacy issues. So, to avoid this normally in a call center they are informing the customer that the call is being recorded and then there will be no issue. Also, some insurance companies mention this in their agreements. If the customer continues with that insurance, then they will automatically be agreed to the conditions which include these privacy concerns.

References

[1] ‘Machine Learning Method for Detecting and Analysis of Fraud Phone Calls Datasets’, *IJRTE*, vol. 8, no. 6, pp. 3806–3810, Mar. 2020, doi: 10.35940/ijrte.F8751.038620.

[2] Z. Tariq, S. Shah, and Y. Lee, ‘Speech Emotion Detection using IoT based Deep Learning for Health Care’, Dec. 2019. doi: 10.1109/BigData47090.2019.9005638.

[3] J. E. Loy, H. Rohde, and M. Corley, ‘Cues to Lying May be Deceptive: Speaker and Listener Behaviour in an Interactive Game of Deception’, *Journal of Cognition*, vol. 1, no. 1, Art. no. 1, Sep. 2018, doi: 10.5334/joc.46.

[4] ‘Background on: Insurance fraud | III’. https://www.iii.org/article/background-on-insurance-fraud (accessed Jun. 28, 2021).

[5] ‘Fraud detection in the Insurance sector’. https://www.sia-partners.com/en/news-and-publications/from-our-experts/fraud-detection-insurance-sector (accessed Jun. 28, 2021).

[6] ‘Official languages in Sri Lanka | Mai Globe Travels’. https://www.maiglobetravels.com/language-in-sri-lanka (accessed Jun. 28, 2021).

[7] T. K. Singh, ‘The True Cost of Insurance Fraud’, *Subex Limited*, Mar. 02, 2021. https://www.subex.com/blog/the-true-cost-of-insurance-fraud/ (accessed Jun. 29, 2021).

[8] K. Han, D. Yu, and I. Tashev, ‘Speech Emotion Recognition Using Deep Neural Network and Extreme Learning Machine’, p. 5.

[9] ‘arXiv Fulltext PDF’. Accessed: Jun. 27, 2021. [Online]. Available: https://arxiv.org/pdf/2102.01813v1.pdf

[10] ‘Han et al. - Speech Emotion Recognition Using Deep Neural Netwo.pdf’. Accessed: Jun. 29, 2021. [Online]. Available: https://www.nematilab.info/bmijc/assets/160518_paper.pdf

[11] M. Piórkowska and M. Wrobel, ‘Basic Emotions’, 2017. doi: 10.1007/978-3-319-28099-8_495-1.

[12] A. Adigwe, N. Tits, K. E. Haddad, S. Ostadabbas, and T. Dutoit, ‘The Emotional Voices Database: Towards Controlling the Emotion Dimension in Voice Generation Systems’,

arXiv:1806.09514 [cs, eess], Jun. 2018, Accessed: Jun. 27, 2021. [Online]. Available: http://arxiv.org/abs/1806.09514

[13] H. Cao, D. Cooper, M. Keutmann, R. Gur, A. Nenkova, and R. Verma, 'CREMA-D: Crowd-sourced emotional multimodal actors dataset', *IEEE transactions on affective computing*, vol. 5, pp. 377–390, Oct. 2014, doi: 10.1109/TAFFC.2014.2336244.

[14] C. Busso *et al.*, 'IEMOCAP: interactive emotional dyadic motion capture database', *Lang Resources & Evaluation*, vol. 42, no. 4, pp. 335–359, Dec. 2008, doi: 10.1007/s10579-008-9076-6.

[15] S. Yoon, S. Byun, and K. Jung, 'Multimodal Speech Emotion Recognition Using Audio and Text', *arXiv:1810.04635 [cs]*, Oct. 2018, Accessed: Jun. 27, 2021. [Online]. Available: http://arxiv.org/abs/1810.04635

[16] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, 'Support vector machines using GMM supervectors for speaker verification', *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 308–311, May 2006, doi: 10.1109/LSP.2006.870086.

[17] '(PDF) Joint Factor Analysis Versus Eigenchannels in Speaker Recognition'. https://www.researchgate.net/publication/3457787_Joint_Factor_Analysis_Versus_Eigenchannels_in_Speaker_Recognition (accessed Jun. 18, 2022).

[18] R. Connaughton, A. Sgroi, K. W. Bowyer, and P. J. Flynn, 'A cross-sensor evaluation of three commercial iris cameras for iris biometrics', presented at the CVPR Workshops, Jan. 2011. Accessed: Jun. 18, 2022. [Online]. Available: https://openreview.net/forum?id=H1E0aT-OWS

[19] 'Contrasting the Effects of Different Frequency Bands on Speaker and Accent Identification | Request PDF'. https://www.researchgate.net/publication/257117919_Contrasting_the_Effects_of_Different_Frequency_Bands_on_Speaker_and_Accent_Identification (accessed Jun. 18, 2022).

[20] S. Safavi, H. Gan, I. Mporas, and R. Sotudeh, 'Fraud Detection in Voice-Based Identity Authentication Applications and Services', *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 2016, Accessed: Jun. 18, 2022. [Online]. Available:

https://www.academia.edu/30857507/Fraud_Detection_in_Voice_based_Identity_Authentication_Applications_and_Services

[21] S. Safavi and I. Mporas, ‘Improving Performance of Speaker Identification Systems Using Score Level Fusion of Two Modes of Operation’, in *Speech and Computer*, vol. 10458, A. Karpov, R. Potapova, and I. Mporas, Eds. Cham: Springer International Publishing, 2017, pp. 438–444. doi: 10.1007/978-3-319-66429-3_43.

[22] Department of Computer Science, Sri Krishna arts and Science, Coimbatore, Tamilnadu, India., S. Sandhya*, N. Karthikeyan, Department of Computer Science, Sri Krishna arts and Science, Coimbatore, Tamilnadu, India., R. Sruthi, and Department of Computer Science, Sri Krishna arts and Science, Coimbatore, Tamilnadu, India., ‘Machine Learning Method for Detecting and Analysis of Fraud Phone Calls Datasets’, *IJRTE*, vol. 8, no. 6, pp. 3806–3810, Mar. 2020, doi: 10.35940/ijrte.F8751.038620.

[23] ‘(PDF) OpenEAR - Introducing the Munich open-source emotion and affect recognition toolkit’. https://www.researchgate.net/publication/224088060_OpenEAR_-_Introducing_the_Munich_open-source_emotion_and_affect_recognition_toolkit (accessed Jun. 18, 2022).

[24] C.-C. Lee, E. Mower, C. Busso, S. Lee, and S. Narayanan, ‘Emotion recognition using a hierarchical binary decision tree approach’, *Speech Communication*, vol. 53, no. 9–10, pp. 1162–1171, Nov. 2011, doi: 10.1016/j.specom.2011.06.004.

[25] ‘(PDF) Audio-Visual Emotion Forecasting: Characterizing and Predicting Future Emotion Using Deep Learning’. https://www.researchgate.net/publication/334422333_Audio-Visual_Emotion_Forecasting_Characterizing_and_Predicting_Future_Emotion_Using_Deep_Learning (accessed Jun. 18, 2022).

[26] ‘[PDF] Credit card fraud detection: A fusion approach using Dempster-Shafer theory and Bayesian learning | Semantic Scholar’. https://www.semanticscholar.org/paper/Credit-card-fraud-detection%3A-A-fusion-approach-and-Panigrahi-Kundu/c749796328e8b20ab31665d3cd52aba2665488ac (accessed Jun. 18, 2022).

[27] ‘Detection of mobile phone fraud using supervised neural networks: A first prototype | SpringerLink’. https://link.springer.com/chapter/10.1007/BFb0020294 (accessed Jun. 18, 2022).

[28] ‘Isolating and analyzing fraud activities in a large cellular network via voice call graph analysis | Proceedings of the 10th international conference on Mobile systems, applications, and services’. https://dl.acm.org/doi/abs/10.1145/2307636.2307660 (accessed Jun. 18, 2022).

[29] ‘Audio Deep Learning Made Simple: Sound Classification, Step-by-Step | by Ketan Doshi | Towards Data Science’. https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5 (accessed Jun. 18, 2022).

[30] ‘Visualizing Sounds Using Librosa - Analytics Vidhya’. https://www.analyticsvidhya.com/blog/2021/06/visualizing-sounds-librosa/ (accessed Jun. 18, 2022).

[31] ‘Playing and Recording Sound in Python – Real Python’. https://realpython.com/playing-and-recording-sound-python/ (accessed Jun. 18, 2022).

[32] ‘sklearn.neural_network.MLPClassifier — scikit-learn 0.24.2 documentation’. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (accessed Jul. 03, 2021).

Appendices